Sociedade Educacional Três de Maio - SETREM
Tecnologia em Redes de Computadores

CARLOS ALBERTO FRANCO MARON

AVALIAÇÃO E COMPARAÇÃO DA COMPUTAÇÃO DE ALTO DESEMPENHO EM FERRAMENTAS OPENSOURCE DE ADMINISTRAÇÃO DE NUVEM USANDO ESTAÇÕES DE TRABALHO

TRÊS DE MAIO – RS
2014

CARLOS ALBERTO FRANCO MARON

AVALIAÇÃO E COMPARAÇÃO DA COMPUTAÇÃO DE ALTO DESEMPENHO EM FERRAMENTAS OPENSOURCE DE ADMINISTRAÇÃO DE NUVEM USANDO ESTAÇÕES DE TRABALHO

Estágio Supervisionado
Sociedade Educacional Três de Maio – SETREM
Faculdade Três de Maio
Tecnologia em Redes de Computadores

Professor Orientador:
Doutorando M. Sc. Dalvan Griebler

Três de Maio
2014

TERMO DE APROVAÇÃO

CARLOS ALBERTO FRANCO MARON

AVALIAÇÃO E COMPARAÇÃO DA COMPUTAÇÃO DE ALTO DESEMPENHO EM FERRAMENTAS OPENSOURCE DE ADMINISTRAÇÃO DE NUVEM USANDO ESTAÇÕES DE TRABALHO

Relatório aprovado como requisito parcial para obtenção do título de **Tecnólogo em Redes de Computadores** concedido pela Faculdade de Tecnologia em Redes de Computadores da Sociedade Educacional Três de Maio, pela seguinte Banca examinadora:

Orientador: Prof. Dalvan Jair Griebler, Drndo.
PUCRS-RS
Faculdade de Tecnologia em Redes de Computadores da SETREM

Prof. Cláudio Schepke, Dr.
Universidade Federal do Pampa – UNIPAMPA – Alegrete - RS

Prof. Denis Valdir Benatti, Esp.
Faculdade de Tecnologia em Redes de Computadores da SETREM

Prof. Vinicius da Silveira Serafim, M.Sc.
Faculdade de Tecnologia em Redes de Computadores da SETREM

Prof Guilherme Damásio Goulart, M.Sc.
Faculdade de Tecnologia em Redes de Computadores da SETREM

Prof. Vera Lúcia Lorenset Benedetti, M.Sc.
Coordenação do Curso Superior de Tecnologia em Redes de Computadores

Faculdade de Tecnologia em Redes de Computadores da SETREM

Três de Maio, 08 de agosto de 2014

AGRADECIMENTOS

Dalvan Jair Griebler, pela confiança imposta à mim durante esta trajetória. Seus concelhos e ensinamentos ficaram para vida toda. E pelo melhor resultado desse trabalho, poder hoje te chamar de amigo.

Mãe Jussara, e pai Aldair (Maronzinho), se cheguei até aqui, foi porque vocês me indicaram o caminho, me apoiando em todas as decisões.

Eloísa Suzana Borin, por cada momento vivido. E ter acreditado que o final seria muito melhor que o início.

Coordenação e demais professores do curso de Tecnologia em Redes de Computadores, o apoio de cada um sempre foi importante no decorrer deste trabalho, e em minha graduação.

Equipe de colaboradores da SETREM, que direta e indiretamente colaboraram com esta pesquisa, disponibilizando equipamentos importantes para alcançar os resultados aqui demostrados.

Ildo Corso, por acreditar no meu conhecimento, e incentivar a continuidade deste trabalho.

RESUMO

A computação em nuvem está se tornando cada vez mais presente nas infraestruturas empresarias e diversas ferramentas estão sendo criadas para auxiliar na administração dos recursos computacionais, que lidam diretamente com tecnologias de virtualização. No entanto, com tantas opções para a tomada de decisão, a escolha se torna difícil, devido a falta de informação disponível sobre o desempenho delas. Sendo assim, o objetivo deste trabalho é estudar, implantar e comparar os ambientes das ferramentas de administração de computação em nuvem (OpenStack e OpenNebula), analisando o desempenho de aplicações paralelas e da infraestrutura (usando benchmarks). Além disso, o trabalho buscou identificar se existem diferenças significativas no desempenho na implantação das ferramentas em relação ao ambiente nativo, e também entre elas. Para isto, foram executados testes de avaliação do desempenho da infraestrutura (Memória, disco, rede, e processador) e das aplicações de alto desempenho de cada ambiente Nativo e Virtual, com o proposito de mostrar um comparativo entre os resultados e analisar estatisticamente as diferenças. Os resultados obtidos indicaram que o OpenNebula foi melhor com os testes das aplicações paralelas e na maior parte dos testes de infraestrutura. O OpenStack obteve vantagem somente nos testes de disco, pois o seu modelo de implantação foi favorecido neste cenário específico.

Palavras-Chave — Redes de Computadores, Computação em Nuvem, Alto Desempenho.

ABSTRACT

Cloud computing is becoming increasingly present in the business infrastructure, and several tools are being developed to assist in the management of computational resources, which are dealing directly with virtualization technologies. However, with so many options to make a decision, it becomes difficult to choose, due to the lack of available information about their performance. Thus, the objective of this work is to study, implement and compare the environments of cloud computing administration tools (OpenStack and OpenNebula), analyzing the performance of parallel applications and the infrastructure (using benchmarks). Moreover, the work seeks to identify whether there are significant differences in performance in the implementation of this tools compared to the native environment, and also between them. For this, we performed tests to evaluate the performance of infrastructure (memory, disk, network, and processor) and high-performance applications of each native and virtual environment, in order to show a comparison between the results, and statistically analyze the differences. The results indicated that the OpenNebula was better in parallel application tests and most of the infrastructure tests. OpenStack only had the advantage in disk tests, because its deployment model was favored in this particular scenario.

Keyword: Network Computing, Cloud Computing, High Performance.

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE SIGLAS

AMD – *Advanced Micro Devices*

API – *Application Interface Programming*

BIOS - *Basic Input/Output System*

COMA - *Cache-Only Memory Architecture*

CPD – Centro de Processamento De Dados

CPUs – *Central Processing Unit*

EBS - *Elastic Block Storage*

FLOPS – *Floating-point Operations Per Second*

GRE - *Generic Routing Encapsulation*

HPC - *High-performance computing*

I/O – *Input/Output*

IaaS – *Infraestructure as a Service*

ISO – *International Organization for Standardization*

KVM – *Kernel-based Virtual Machine*

LAN – *Local Area Network*

MAC - *Media Access Control*

Mb/s – Mega bits por Segundo

MIMD - *Multiple Instruction Multiple Data*

MISD - *Multiple Instruction Single Data*

MPI - *Message Passing Interface*

MR – *MapReduce*

MTU – *Maximum Transmission Unit*

NORMA - *NO-Remote Memory Access*

NUMA - *Non-Uniform Memory Access*

PaaS – *Plataform as a Service*

POSIX - Portable Operating System Interface

PUCRS – Pontifícia Universidade Católica do Rio Grande do Sul

RAM – *Random Access Memory*

RPC – *Remote Procedure Call*

SaaS – *Software as a Service*

SIMD - *Single Instruction Multiple Data*

SISD - *Sigle Instruction Single Data)*

SMP - *Symmetric Multiprocessor*

TI – Tecnologia da Informação

UFPEL – Universidade Federal de Pelotas

UFRGS – Universidade Federal do Rio Grande do Sul

UFSM – Universidade Federal de Santa Maria

UMA - *Uniform Memory Access*

VLAN – Virtual LAN

VM – *Virtual Machine*

Vmdk - *Virtual Machine Disk*

Vmx - *Virtual Machine eXtensions*

VPN – *Virtual Private Network*

VSWP - *Virtual Machine Swap File*

WAN – *Wide Area Network*

XML - *eXtensible Markup Language*

SUMÁRIO

INTRODUÇÃO

Na atualidade, existe uma grande expectativa na aceitação e na utilização da computação em nuvem. Pois dentre as inúmeras vantagens que existem, PaaS (*Plataform as a Service)* oferece serviços voltados às plataformas específicas para implantações ou desenvolvimento de *softwares* na nuvem. SaaS (*Software as a Service*) visa oferecer serviços de *softwares* na nuvem. IaaS (*Infrastructure as a Service*) disponibiliza serviços a fim utilizar uma infraestrutura (máquinas virtuais, servidores, *firewalls*) para processamentos em nuvem. Todos esses modelos de serviços, são disponibilizados em estruturas que muitas vezes a localização física é desconhecida, porém, todo o processamento é entregue ao usuário através da rede, podendo ser a internet.

Uma pesquisa realizada por THOMÉ, et al., (2013) aonde procurou estudar e avaliar as principais ferramentas de computação em nuvem existentes no mercado para o modelo IaaS. A pesquisa teve como objetivo testar as funcionalidades operacionais (interface, instalação, configuração) de duas ferramentas (OpenStack e OpenNebula) escolhidas a partir de uma avaliação teórica. Este que foi realizado no laboratório de Redes da SETREM, usando estações de trabalho. Assim, puderam comparar se os conceitos de operacionalização descritos na teoria, se refletiam na prática.

O trabalho realizado por THOMÉ, et al., (2013) não contemplou a avaliação aprofundada do desempenho das ferramentas que fora usado na pesquisa. Diante disto, o proposito deste trabalho é avaliar o desempenho, com ênfase em aplicações paralelas e da infraestrutura virtual das ferramentas OpenStack e OpenNebula, dando

continuidade na pesquisa e avaliando estas duas ferramentas de computação em nuvem.

Portanto, o presente trabalho propõe a implantação das ferramentas de computação em nuvem IaaS em *clusters* homogêneos, para montar uma mesma estrutura de máquinas virtuais na nuvem com o objetivo de comparar o desempenho de cada uma das ferramentas, tendo como referência os resultados de execuções em ambientes Nativos. Para isto, a proposta é usar *benchmarks* de aplicações de alto desempenho e de infraestrutura, e assim comparar os resultados entre as ferramentas.

Partindo então de uma ideia da utilização de ferramentas de computação em nuvem, a utilização de infraestrutura com estações de trabalho, juntamente com pesquisas voltadas à avaliação de desempenho de computação em nuvem, virtualização e aplicações paralelas para alto desempenho, este trabalho busca obter resultados que possam contribuir para o meio acadêmico e ainda compreender a real diferença de desempenho em cada ferramenta avaliada. Além destes objetivos, a pesquisa ainda busca realizar um estudo em trabalhos relacionados à área de pesquisa, buscando encontrar formas de contribuir com a comunidade científica.

Este trabalho está divido em 3 capítulos. O primeiro, aborda tema, delimitação do tema, hipóteses, variáveis, justificativa, objetivos, cronograma e orçamento. O segundo se encontra todo o referencial teórico necessário para a pesquisa. Nesta seção são abordados assuntos como virtualização, conceitos de computação em nuvem, avaliação de desempenho, entre outros. E o terceiro e último capítulo, são apresentados todos os resultados obtidos no decorrer da a pesquisa.

.

CAPÍTULO 1: **PROJETO DE PESQUISA**

1.1 TEMA

Avaliação e comparação de desempenho em uma nuvem utilizando estações de trabalho.

1.2 DELIMITAÇÃO DO TEMA

Avaliação do desempenho da infraestrutura e de algumas aplicações paralelas em uma nuvem composta de estações de trabalhos usando ferramentas *open source* de administração do modelo IaaS.

O projeto foi desenvolvido pelo acadêmico do curso de Redes de Computadores Carlos Alberto Franco Maron como trabalho de conclusão de curso, com orientação do Doutorando Dalvan Jair Griebler e foi realizado na Sociedade Educacional Três de Maio – SETREM, durante o período de novembro de 2013 à Julho de 2014.

1.3 FORMULAÇÃO DO PROBLEMA

Segundo Matteson (2013), é esperado para até 2020 um aumento em processamentos realizados em nuvem. Levando esse conceito adiante uma pesquisa realizada por Jacquet, et al. (2012) mostra que entre as empresas que migraram sua estrutura para a nuvem particulares ou de grandes provedores, tiveram o principal objetivo à diminuição de custos, e aumento da disponibilidade dos serviços e desempenho.

Portanto, para este fim, um ambiente de nuvem (virtualizado) deve manter semelhanças a um ambiente clássico. Com estes conceitos, e ao longo do entendimento sobre o assunto, uma infraestrutura de nuvem (IaaS) composta por estações de trabalho apresenta um desempenho compatível com um ambiente Nativo em aplicações e na infraestrutura virtual (rede, memória, disco, processador)?

1.4 HIPÓTESES

1) O desempenho da infraestrutura (disco, rede, memória e processamento) virtual é significativamente afetado em relação ao ambiente real[1].

    a) **Validação:** Para validar esta hipótese será necessário executar um benchmark que testa a infraestrutura nos dois ambientes e verificar através de teste estatístico se o desempenho é significativamente afetado.

2) O desempenho de aplicações paralelas (*speed-up,* eficiência, tempo de execução) em um ambiente de nuvem não é significativamente afetado em relação ao ambiente real[1].

    a) **Validação:** Para validar esta hipótese será necessário executar um benchmark que possui um conjunto de aplicações paralelas nos dois ambientes e verificar através de um teste estatístico se o desempenho é significativamente afetado.

3) O desempenho obtido nas aplicações paralelas e na infraestrutura não é significativamente diferente entre as ferramentas OpenStack e OpenNebula na nuvem.

[1] Modelos de serviços IaaS trabalha com ferramentas de virtualização. Neste contexto, o ambiente real é composto por memória, disco, rede, e processamento sem a camada de virtualização implantada por ferramentas para este propósito. Sendo de forma genérica, o ambiente real é composto somente por camada física e camada de *software (S.O).*

a) **Validação**: para validar esta hipótese serão comparados os resultados obtidos no ambiente virtual em ambas as nuvens sobre o desempenho de aplicações paralelas e da infraestrutura.

1.5 VARIÁVEIS

- Desempenho
- *Throughput*
- Speed-up
- Tempo de execução

1.6 OBJETIVOS

1.6.1 **Objetivo Geral**

Implantação, testes e avaliação do desempenho da infraestrutura e aplicações paralelas usando ferramentas de computação em nuvem *Open Source* em um ambiente de estações de trabalho.

1.6.2 **Objetivos Específicos**

- Estudar tecnologias de virtualização.
- Estudar as principais aplicações de computação em nuvem.
- Estudar como avaliar o desempenho.
- Estudar e Implantar as ferramentas *Open Source* OpenStack e OpenNebula.
- Avaliar o desempenho de aplicações paralelas e da infraestrutura em nuvem formada com estações de trabalho.

1.7 JUSTIFICATIVA

Sendo cada vez mais acessível a utilização de computadores e a facilidade de conexão com a internet, criou-se uma demanda contínua para acesso às informações de qualquer lugar, com a computação em nuvem entrando fortemente

neste conceito. Isso porque, de acordo com Dell, (2011), é importante pensar na nuvem como um facilitador de resultados empresarias, pois conduz ao aumento da produtividade de colaboradores garantindo uma vantagem competitiva, transformando verdadeiros valores de crescimento e prosperidade.

Sendo a computação em nuvem uma tecnologia em ascensão, pesquisas afirmam que para até 2018 são esperados que 30,2% das cargas de trabalho serão executadas em ambientes na nuvem (MATTESON, 2013). Isto mostra que cada vez mais pessoas buscarão esses tipos de serviços, por motivos que vão desde a economia de investimentos até escalabilidade de recursos a qualquer momento, motivando cada vez mais o uso de ferramentas voltadas para computação em nuvem.

Nos serviços de computação em nuvem desempenho e disponibilidade do serviço são itens que proporcionam a qualidade de serviços em infraestruturas. Para implantação de ambientes como *datacenters* (espaços rigorosamente planejados para alocação de computadores de alto desempenho) onde são comumente encontradas ferramentas de computação em nuvem. Isso requer um alto custo de *hardware* e manutenção, onde estima-se que para manter um número equivalente de 100.000 servidores, o custo de manutenção pode alcançar a casa dos milhões de dólares (VERDI, 2010).

Como mostrou a pesquisa sobre o uso de tecnologias de informação e comunicação nas empresas, no ano de 2010. Dentre as 2,8 milhões de empresas com mais de um colaborador, 80,8% (equivalente a 2,1 milhões) utilizaram computador para trabalho (IBGE, 2010). Com base nestas informações é visível que empresas estão cada vez mais informatizando seus processos e destinando computadores aos seus colaboradores.

Computação em nuvem pode ser encarada como um alto investimento financeiro quando partindo para a implantação de uma nuvem privada. Visando o ponto comercial do produto “Serviço de Computação em Nuvem”, é necessário e importante um alto investimento das empresas que comercializarão este tipo de serviços em nuvens públicas. Todavia empresas que querem adotar o serviço de computação em nuvem em sua estrutura de TI, poderão economizar expressivamente

investimentos no setor de TI, alocando seus serviços em uma nuvem pública, alugando uma estrutura e pagando somente pelo uso, como é hoje a política comercial destes serviços.

No entanto, resultados de aplicações de alto desempenho em ambientes utilizando ferramentas *open sorurce* de administração de nuvem se tornou questionável e por isso este trabalho teve como objetivo investigar o desempenho nestes ambientes, a fim de analisar a infraestrutura de *hardware* virtualizado, o comportamento das ferramentas *open source* do modelo IaaS e das aplicações.

A pesquisa buscou encontrar evidências sobre a diferença de desempenho entre as ferramentas *open source* de administração de nuvem executando aplicações para computação de alto desempenho. No resultado final, com conhecimento adquirido durante as pesquisas e testes experimentais, foi possível confirmar a implantação de ferramentas nestes cenários.

1.8 METODOLOGIA

Nesta seção serão descritos os métodos que serão usados para se alcançar e tratar os resultados da pesquisa.

1.8.1 **Método de Abordagem**

Pode-se pensar em duas dimensões para os métodos de abordagem, a primeira diz respeito ao tipo de raciocínio que é utilizado para se inferir a conclusão e a segunda é relacionada com a utilização, ou não de números ou estatísticas (LOVATO, 2013, p.29).

A pesquisa partiu do método hipotético dedutivo, que a partir do conhecimento teórico, foi aplicado as ferramentas o*pen source* e avaliação de seu desempenho em ambientes programados para analisar se foi praticável o uso de computação em nuvem utilizando estações de trabalho. E método quantitativo para que todas as informações coletadas no decorrer da pesquisa fossem classificadas e avaliadas.

1.8.2 **Método de Procedimento**

- Coleta de dados – Os resultados de testes e experimentos, foram devidamente coletados para análise.
- Análise – os dados coletados serão analisados e interpretados com base nos conhecimentos e estudos adquiridos com a teoria.

1.8.3 **Técnicas**

Como técnica, foi usada a experimental, esta foi efetuada simulando ambientes com o uso de ferramentas específicas, de forma que foi induzido variáveis para manipulação do ambiente para a obtenção dos dados, com o propósito de classifica-los levando em conta o nível de processamento alcançados nos testes.

1.9 DEFINIÇÃO DE TERMOS

1.9.1 **Redes WAN**

Segundo Tanenbaum (2003), as redes WANs, são redes geograficamente distribuídas que abrangem uma grande área geográfica, um país ou até mesmo um continente. Este tipo de rede possui um conjunto de máquinas, que tem a finalidade de executar os programas dos usuários, e que estão conectados a uma sub-rede. Estas sub-redes transportam as mensagens de um *host* para outro.

Na maioria das WAN's, as redes possuem muitas linhas de transmissão que estão conectados a uma porção de roteadores. WAN consiste na comunicação com outras maquinas fora do arranjo físico da rede LAN.

1.9.2 **Redes LAN**

Segundo Tanenbaum (2003), são redes contidas em um prédio ou em uma universidade que tem alguns quilômetros de extensão. Elas são amplamente usadas para conectar computadores pessoais e estações de trabalho em escritório e instalações industriais, permitindo os compartilhamentos de recursos e a troca de informações.

As redes LAN's funcionam em várias velocidades, podem variar de 10/100/1000 Mb/s.

1.9.3 **Modelo de referência OSI**

Modelo proposto pela ISO (*International Standard Organization)*, tratando da interconexão de sistema abertos de comunicação com outros sistemas – significou o primeiro passo para padronização da internet – sendo este modelo dividido em 7 (sete) camadas: Físico, enlace, rede, transporte, sessão, apresentação, aplicação (TANENBAUM, 2003).

O modo de como as camadas são distribuídas, faz com que a pacote que será transmitido pela rede, ultrapasse todos os níveis de camadas até chegar ao seu destino real na pilha de comunicação, assim cada camada tem um método de comunicação com as camadas adjacentes, que interpretam de forma adequada as informações trocadas entre as camadas. Cada pacote de rede no momento de seus encaminhamentos, carrega consigo as informações que garantirão a interpretação correta na pilha de camadas, e junto com elas, informações que serão importantes para o transporte destes pacotes na rede.

Este modelo de referência busca orientar a forma de como os protocolos de rede e equipamentos devem se comunicar através da rede. Sendo um tanto abstrato, alguns equipamentos e protocolos não seguem à risca este modelo de implantação.

A seguir as descrições das camadas pertencentes a este modelo de referência.

1.9.4 **Camada Física**

De acordo com Tanenbaum (2003), a camada física tem a missão de permitir a comunicação através do meio físico de transmissão, onde nesta camada ocorre a emissão do bit através dos sinais elétricos para que o destinatário possa receber a mensagem correta. Neste caso, usa-se uma voltagem específica para definir cada bit que será transmitido no meu físico.

1.9.5 **Enlace de Rede**

Esta camada faz a conversão de bits lógicos para frames. A camada de enlace recebe os bits da camada de rede e os transforma em frames, após esta conversão, é verificado se estes dados estão íntegros ou esparsos e após isso, envia um frame de confirmação para o remetente. Esta camada também e responsável pelo controle de fluxo, isso ocorre quando se tem a comunicação realizada por um emissor muito rápido e o receptor é mais lento (TANENBAUM, 2003).

1.9.6 **Camada de Rede**

A camada de rede, é responsável pela operação da sub-rede, determinando como são roteados os pacotes da origem até o destino. Para ocorrer o roteamento entre redes, é necessária a utilização de rotas, sendo elas estáticas ou altamente dinâmicas, sendo designado para cada pacote (TANENBAUM, 2003).

1.9.7 **Camada de Transporte**

A função básica da camada de transporte é aceitar dados da camada acima dela, fragmentá-los se necessário, passar para adjacente, e garantir que todos os fragmentos chegarão ao seu destino. Esta camada também é responsável em designar o tipo de conexão para a camada de sessão e aos usuários da rede. A camada de transporte é a única camada fim a fim que interliga o emissor ao destino (TANENBAUM, 2003).

1.9.8 **Camada de Apresentação**

A camada de aplicação, diferenciando das mais baixas camadas mostradas até aqui, a camada de apresentação, como Tanenbaum (2003) define, é a camada relacionada à sintaxe e a semântica das informações transmitidas pelas outras camadas. Para tornar a comunicação possível entre computadores com representações distintas de dados abstratos, como a criptografia das informações, a camada de apresentação gerencia essas estruturas.

1.9.9 **Camada de Aplicação**

Esta camada é responsável em fazer a comunicação entre as solicitações de serviços e as aplicações. Ao abrir uma página na internet, por exemplo, a aplicação responsável usará um pacote específico, que neste caso pode ser o HTTP. Assim a partir desse ponto a solicitação será transformada em um pacote de rede, percorrerá físico e lógico até seu caminho até o servidor WEB, e ele irá transmitir a página de volta (TANENBAUM, 2003).

1.9.10 **Modelo de referência TCP/IP**

Esse modelo recebe os nomes dos primeiros protocolos firmados neste modelo de referência. Este modelo se encontra dividido em 4 (quatro) camadas: *Host*-rede, inter-redes, transporte e aplicação.

Inicialmente, o modelo de referência TCP/IP foi desenvolvido com para uso em atividades militares para a criação de redes pertencentes ao departamento de defesa dos Estado Unidos, denominada ARPANET.

O motivo da utilização de somente 4 camadas no modelo de referência TCP/IP foi pelo motivo de ser um primeiro passo para um modelo organizado da estrutura da comunicação em redes. Posteriormente acabou sendo criado os modelos de referência OSI, com o objetivo de corrigir erros que o modelo TCP/IP possuía.

A seguir será descrito as camadas pertencentes ao modelo de referência.

1.9.10.1 Camada de Aplicação

No modelo TCP/IP não é encontrado as camadas de sessão e apresentação, pois poucas aplicações as utilizam. Essa camada possui todos os protocolos de nível mais alto da pilha do modelo de referência, podemos citar o FTP, SMTP, DNS, HTTP, e entre outros. Essa camada se comunica com a camada que de transporte, que por sua vez é a que está logo abaixo da camada de aplicação (TANENBAUM, 2003).

As portas têm um número padrão correlacionando serviços aos protocolos da camada mais alta, por exemplo: para acessar uma página na internet usamos o protocolo HTTP ou HTTPS que utilizam a porta 80 ou 443, assim o protocolo TCP/IP sabe o conteúdo dos pacotes que trafegam na rede. Comparando o modelo de referência TCP/IP ao OSI, a camada de aplicação corresponde a camada de sessão, apresentação e aplicação do modelo OSI (TANENBAUM, 2003).

1.9.10.2 Camada de Transporte

Esta camada está localizada acima da camada de inter-redes, onde sua finalidade é que entidades pares dos *hosts* de origem e destino mantenham uma conversação.

Dois protocolos se situam nesta camada, o TCP (*Transmission Control Protocol*) e o UDP (*User Datagram Protocol*), o TCP é orientado a conexão permitindo a entrega de pacotes sem erros entre a origem e destino. O UDP é um protocolo que não é orientado a conexão, onde não possui um controle de erros e nem ordem na entrega de pacotes, porém é mais rápido.

A recepção dos dados é feita a partir da camada de internet, que pega os dados e entrega a camada de transporte. Caso seja necessário, ela organiza os pacotes em ordem e envia para a camada de aplicação, assim chegando de fato até a aplicação (TANENBAUM, 2003).

1.9.10.3 Camada Inter-Redes

Com a necessidade da comunicação entre *hosts* geograficamente distantes, foi escolhida uma camada que pudesse transmitir pacotes entre redes e que eles não se perdessem durante o trajeto garantindo a entrega dos mesmos.

Os pacotes enviados por um *host* A que está em uma rede para outro *host* B que está em outra rede distante, ele passa por diversos roteadores, e poderá ocorrer dos pacotes chegar foram de ordem. Então as camadas superiores fazem o reordenamento dos pacotes recebidos, e sem algum pacote não tenha sido recebido, estes mesmo pacotes são solicitados novamente. Todo esse processo é transparente

para o usuário, ou seja, ele não percebe todo o processo que acontece (TANENBAUM, 2003)

1.9.10.4 Camada Host-rede

O modelo de referência TCP/IP não especifica muito o que acontece nesta camada, apenas que o *host* tem que se conectar na rede para enviar os pacotes. Isso depende de *host* para *host* e rede para rede. Essa camada da pilha TCP/IP é equivalente às camadas física e enlace do modelo OSI (TANENBAUM, 2003).

1.9.11 **Equipamentos de redes**

Representam os equipamentos que estão em contato direto com o tráfego das informações na rede.

- **Roteador:** é um equipamento munido de um *software* que controla os caminhos que cada pacote de rede deve seguir.

> É um *software* que utiliza o cabeçalho de pacote para escolher uma linha de saída. Quando um pacote entra em um roteador, o cabeçalho de quadro e o final são retirados, e o pacote localizado no campo de carga útil do quadro, para assim encaminhá-lo ao destino da estação (TANENBAUM, 2003, pág. 256).

- **Switches:** São equipamentos que trabalham com chaveamento de portas, onde são conectados os dispositivos da rede.

> Switches são dispositivos que filtram e encaminham pacotes entre segmentos de redes locais, operando na camada de enlace (camada 2) do modelo RM-OSI. São responsáveis por organizar logicamente as estações para que os respectivos pacotes cheguem ao seu destino (TANENBAUM, 2003, pág. 258).

1.9.12 ***Open Souce*** **(Código Aberto)**

Termo criado pela OSI (*Open Source Iniciative)* para definir critérios que caracterizam um *Software* como sendo de código aberto. Porém o termo *open source*, não define somente o acesso ao código fonte, outros critérios devem ser considerados para que o *software* seja *open source*, os critério são: Distribuição livre, código aberto, trabalhos derivados, integridade do autor do código fonte, não discriminação contra

pessoas ou grupos, não discriminação contra áreas de atuação, distribuição da licença, licença não específica à um produto, licença não restrinja outros programas e licença neutra em relação a tecnologia (SOFTWARE LIVRE BRASIL, 2013).

1.10 CRONOGRAMA

O Quadro 1 (um) apresenta as atividades que foram desempenhadas no andamento da pesquisa durante os meses previstos para cada uma delas.

A primeira coluna faz referência às atividades que foram planejadas para serem realizadas com o objetivo de alcançar os resultados da pesquisa. Nas colunas seguintes, encontra-se os meses que em que cada atividade foi realizada. Para etapas planejadas, o quadro está preenchido com o sombreado, indicando o item planejado e o mês que pretendia ser desenvolvido. Para etapas cumpridas, o quadro terá um símbolo de um círculo (O), identificando o mês e a etapa aplicada.

ATIVIDADE	Dezembro	Janeiro	Fevereiro	Março	Abril	Maio	Junho	Julho
Projeto de Pesquisa	O							
Estudar e fundamentar os conceitos sobre computação em nuvem.	O	O						
Estudar aplicação de computação em nuvem.	O	O						
Estudar ferramentas de virtualização			O					
Estudar ferramentas as ferramentas OpenStack e OpenNebula.	O	O	O	O	O	O	O	
Estudar métodos de avaliação de desempenho em computação em nuvem.			O					
Pesquisar e ponderar trabalhos relacionados à avaliação de desempenho em IaaS.		O	O	O	O			
Estudar ferramentas de avaliação de desempenho de computação em nuvem.				O	O	O	O	
Implantar os dois ambientes de nuvem com OpenNebula e OpeNStack		O	O	O	O	O		
Monitoramento de coleta dos resultados					O	O	O	O
Formulação do relatório sobre as análises						O	O	O
Escrita de artigos sobre a pesquisa	O	O	O					

Fonte: Maron, Griebler, 2013.

Quadro 1: Cronograma das atividades

1.11 RECURSOS

Para andamento da pesquisa e auxílio para alcançar os objetivos, foram necessários o uso de recursos que serão descritos a seguir.

1.11.1 **Recursos Humanos**

Os recursos humanos foram: professor orientador, professores, acadêmicos do curso de Tecnologia em Redes de Computadores, funcionários da Faculdade.

1.11.2 **Recursos Materiais**

Os recursos materiais foram: equipamentos de informática, folhas, livros.

1.11.3 **Recursos Institucionais**

Os recursos institucionais são: laboratórios de informática, internet, biblioteca, central de cópias SETREM, coordenação do curso de Redes de Computadores.

1.12 ORÇAMENTO

Para realização do projeto foi previsto os itens descritos no Quadro 2 (dois), que está disposto da seguinte maneira:

- A primeira coluna refere-se à identificação dos itens que serão necessários.
- A segunda coluna descreve a quantidade desses itens.
- A terceira coluna demonstra o valor unitário de cada item em moeda correte nacional.
- A quarta exibe o somatório da coluna dos valores de cada item em moeda correte nacional.
- E a última um valor total de todos os itens em moeda correte nacional.

REFERÊNCIA	QUANTIDADE	VALOR UNITÁRIO	VALOR TOTAL
Impressões	1000	0,15	150,00
Encadernação espiral	5	3,00	15,00
Encadernação capa dura	2	50,00	100,00
Combustível	300	3,09	927,00
Horas trabalhadas	500	45,00	16.000,00
		TOTAL	17.253,25

Fonte: Maron, Griebler, 2013.

Quadro 2: Orçamento

CAPÍTULO 2: REFERENCIAL TEÓRICO

Esta seção detalha os principais assuntos necessários para o entendimento da pesquisa que fora detalhada no capítulo anterior. Dentre os vários assuntos estão: Arquiteturas paralelas, recursos de virtualização, computação em nuvem, desempenho de computação e ferramentas de desempenho, entre outros. Cada assunto será dividido em seções abordando os assuntos e algumas características e entendimentos peculiares sobre o mesmo.

2.1 ARQUITETURAS PARALELAS

Um servidor hoje ocupa um papel muito importante nas atividades em que é destinado. Ele deve ser capaz atender as ações que foram designadas à ele para serem cumpridas, tanto em processamento, como armazenamento de informações.

Um servidor tem o objetivo importante de cumprir estes quesitos. Mesmo buscando a simplicidade nestes tipos de arquiteturas, os desafios atuais tendem a obter um aumento constante do processamento, não deixando de lado os baixos custos financeiros. Portanto a facilidade no uso, arquiteturas compactas, alto desempenho, baixo custo e baixo consumo de energia devem ser alguns dos objetivos principais em arquiteturas paralelas. (PARHAMI, 2002).

De forma um tanto genérica, as arquiteturas de computação paralela se definem na combinação de processadores/servidores, trafegando e trocando informações entre si, em uma alta largura de banda, de forma simples e rápida (PARHAMI, 2002).

Portanto, a utilização de arquiteturas paralelas é para prover um alto poder computacional. Com objetivo de ser usadas aplicações que exijam precisão e rapidez nos seus processos, onde não atingiriam o mesmo desempenho e eficiência em arquiteturas convencionais não paralelas. Algumas áreas que fazem utilização constante de processamento paralelo, devido a intensa massa de informações que precisam ser processadas, são as pesquisas espaciais, meteorologia, engenharia, medicina, que necessitam de grande poder computacional para o processamento dos dados. Sendo que a TI tem uma preocupação constante em aumentar esse nível computacional, não importando os esforços que serão empregados (PARHAMI, 2002).

Assim sendo, as motivações que movem o aumento do processamento paralelo é a busca de maior velocidade, obtendo soluções de problemas computacionais com maior rapidez. Sendo também, uma maior produção, resultando em maior número de resoluções de caso em menos tempo, tornando-se um ponto importante na execução de tarefas (processos) semelhantes. Além da velocidade, é instigante o aumento do poder computacional que atualmente é encontrado em valores de TFLOPS. Esse aumento traria um maior processamento em menos tempo. Sendo assim, as motivações de busca de maior velocidade, maior produção, e maior aumento computacional, movem a ambição da computação paralela em chegar nos valores PFLOPS de processamento (PARHAMI, 2002).

Hoje a computação paralela pode ser encontrada em duas classes distintas. O SMP *(Symmetric Multiprocessor)*, também conhecido por sistema fortemente acoplado. Que consiste em um sistema em que diversos processadores, compartilham uma mesma memória física, aonde o gerenciamento é feito somente por um único sistema operacional. (CHEVANCE, 2005).

De acordo com Chevance (2005), outra é a classe das arquiteturas flexíveis, em que são constituídos através de redes de alta performance, com um determinado número de sistemas independentes, todos com seus próprios recursos de processamento e com os sistemas operacionais individuais.

Porém, ainda existe os sistemas massivamente paralelos, que se enquadram em sistema de baixo acoplamento. Esta classe caracteriza-se pelo fato que não há um compartilhamento de *hardware,* apenas no processamento e na interconexão (CHEVANCE, 2005).

Dentre os 3 (três) tipos de arquiteturas abordadas pelo autor, SMP pode ser constituída por processadores convencionais, usados em estações de trabalho, servidores e computadores portáteis, porém em uma infraestrutura que permita o paralelismo no processamento. Já as arquiteturas flexíveis, são mais complexas, exigem uma programação mais detalhadas da maioria dos componentes físicos e lógicos. E na arquitetura de baixo acoplamento, o processamento é independente em cada nodo da estrutura, necessitando uma rede de comunicação eficiente.

Arquiteturas paralelas são alternativas de infraestruturas para se alcançar um alto poder de processamento. Conforme a necessidade das áreas de pesquisas, a computação paralela pode se tornar um aliado forte, na obtenção de resultados importantes, como na medicina, geologia, entre outros.

Com toda essa capacidade de processamento que o conjunto de arquiteturas paralelas comportam, a computação em nuvem pode ser privilegiada com este poder computacional. Porém, a necessidade de um estudo para implementação desse conjunto é importante, e toda a arquitetura de processamento por trás disso, deve ser levado em conta no momento da implantação de uma nuvem computacional, não importando o modelo de implantação.

Nas seções seguintes, serão abordados alguns exemplos de arquiteturas paralelas.

2.1.1 **Exemplos de arquiteturas paralelas**

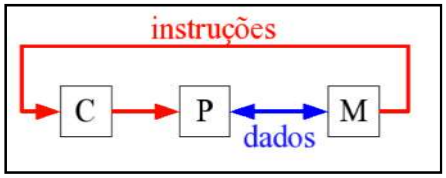
As arquiteturas paralelas podem ser classificadas pelo acesso a memória RAM, e também pelo fluxo de instruções e dados enviados ao processador, como: SISD, SIMD, MIMD, MISD, chamada de taxonomia de Flynn. As arquiteturas UMA e NUMA (sendo COMA uma subdivisão da NUMA) pertencem ao grupo de memória

compartilhadas (Multiprocessadores). NORMA pertence ao grupo de memória não compartilhada (Multicomputadores). Todos esses modelos serão abordados nas seções seguintes.

2.1.1.1 SISD (Sigle Instruction Single Data)

De acordo com Chevance (2005), este é o método simples e tradicional de processamento, executado por processadores nas arquiteturas SISD.

Na figura 1 o fluxo de instruções e de processamento segue uma fila contínua e singular, onde os dados recebidos da memória e as instruções, são executadas unicamente no processador e em ordem. É o exemplo de processadores de estações de trabalho.

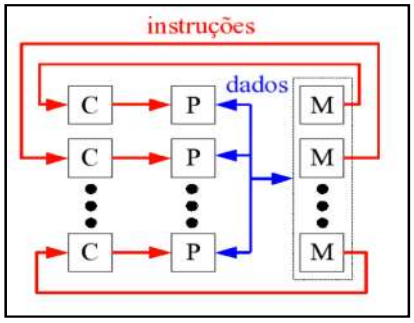


Adaptado de Chevance, 2005.

Figura 1: Exemplo do processamento da arquitetura SISD.

2.1.1.2 SIMD (Single Instruction Multiple Data)

De acordo com Chevance (2005), uma única instrução é responsável em processar vários dados de entrada no processador, seguindo ainda um fluxo sequencial de processamento.



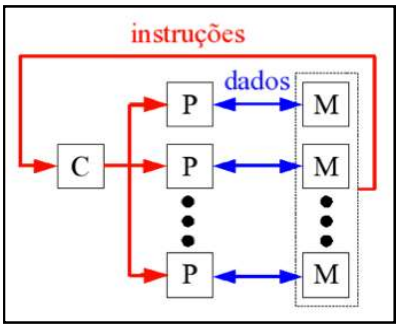
Adaptado de Chevance, 2005.

Figura 2: Exemplo do processamento da arquitetura SIMD.

Na Figura 2, é possível perceber que existe mais de um processador, e apenas um único fluxo de dados entre a memória e o processador. A utilização deste tipo de arquitetura se faz com processadores gráficos (GPUs), arquiteturas Array (Matrizes, imagens, vetores, etc).

2.1.1.3 MISD (Multiple Instruction Single Data)

Segundo Chevance (2005), define este tipo de arquitetura como sendo o processamento de um único dado por diversas instruções, onde somente é permitido uma execução por vez.



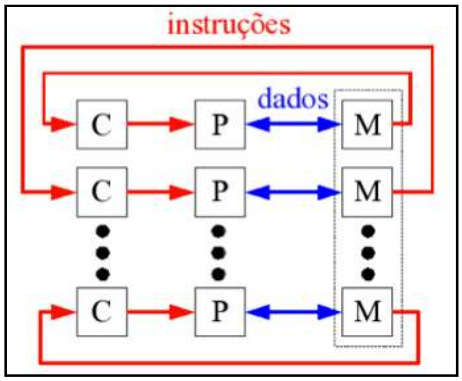
Adaptado de Chevance, 2005.

Figura 3: Exemplo do processamento da arquitetura MISD.

Na Figura 3 mostra os vários processadores e memórias, onde o mesmo fluxo de instruções aplicadas aos diversos processadores, processam um único dado. Ainda não existe uma implementação real desta arquitetura, permanecendo somente na classe teórica.

2.1.1.4 MIMD (Multiple Instruction Multiple Data)

Segundo Chevance (2005), são fluxos de instruções diferentes, aplicados a diferentes dados de um processamento.



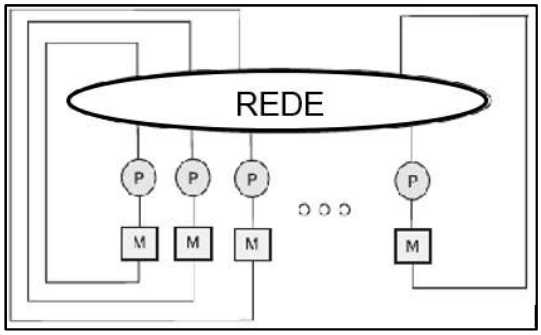
Adaptado de Chevance, 2005.

Figura 4: Exemplo do processamento da arquitetura MISD.

Na Figura 4, os fluxos de instruções e dados independente em cada execução. Conforme Chevance (2005), este tipo de arquitetura é conceito no processamento paralelo, pois neste caso, vários programas fazem uso de vários dados para obter os resultados do processamento. Este tipo de arquitetura ocorre em infraestruturas mais modernas, servidores multiprocessados.

2.1.1.5 NUMA (Non-Uniform Memory Access)

As arquiteturas de *hardware* NUMA, é caracterizada por possuir grupos de processadores. Estes grupos são compostos por memórias e dispositivos de I/O específicos para estes grupos. Para otimização dos recursos e agilidade nos processos. Estes grupos podem acessar memórias pertencentes a outros grupos de processadores (CHEVANCE, 2005).



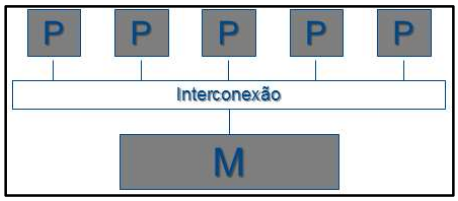
Adaptado de Ozdogan, 2006

Figura 5: Exemplo arquitetura NUMA

Na Figura 5 a letra “P” representa os processadores. Já a letra “M” representa a memória RAM. Ilustrando desta forma o grupo de processadores, aonde todos possuem memória dedicada, mas sendo estas memórias acessíveis pela rede aos outros processadores.

2.1.1.6 UMA (Uniform Memory Access)

De acordo com Chevance (2005), neste tipo de arquitetura de compartilhamento de memória, o canal de I/O, de acesso a memória e o tempo para o mesmo são únicos para todos os processadores da infraestrutura.



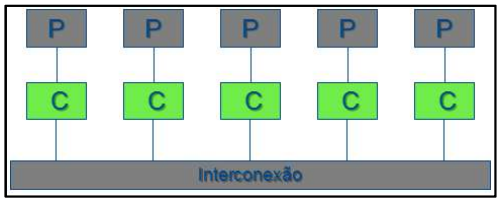
Adaptado de Chevance, 2005

Figura 6: Exemplo de uma arquitetura UMA.

No exemplo da Figura 6, existe o grupo de processadores, e para acesso à memória disponível, é somente por um canal de interconexão. Em algumas implementações a memória cache do processador podem abstrair o tempo de latência, mas neste caso, pode se tonar um gargalo no processamento (CHEVANCE, 2005).

2.1.1.7 COMA (Cache-Only Memory Architecture)

Neste tipo de implementação, conforme Chevance (2005), cada processador da infraestrutura possui uma memória cache, e memória RAM dedica. Porém, cada memória cache da infraestrutura é compartilhada com todos os processadores, permitindo que um processador acesse endereços de uma outra memória cache da mesma infraestrutura.



Adaptado de Chevance, 2005.

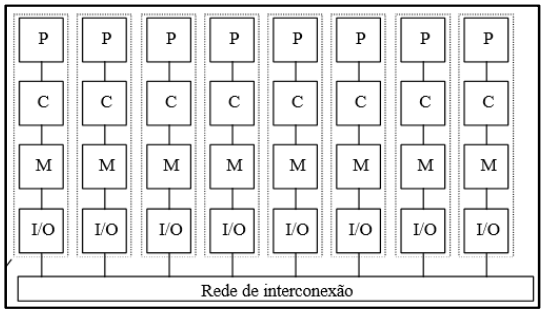
Figura 7: Exemplo de uma arquitetura COMA.

No exemplo da Figura 7, há a representação de uma interconexão na infraestrutura que permite que cada memória cache seja acessível aos outros processadores da infraestrutura.

2.1.1.8 NORMA (NO-Remote Memory Access)

Segundo Chevance (2005), nesta arquitetura cada processador visualiza sua própria memória. Porém toda a comunicação é feita através de requisições de mensagens pela rede. É o exemplo de *clusters* de estações de trabalho, ou servidores.

Na Figura 8, é exemplificada a arquitetura NORMA, aonde possui a existência de processadores, memória cache e RAM, dispositivos de I/O. Todos estes individuais na estrutura, porem interligados por uma rede de interconexão.



Adaptado de Chevance, 2005.

Figura 8: Exemplo de uma arquitetura NORMA

2.2 SISTEMAS DISTRIBUÍDOS

Sistemas distribuídos são computadores interligados através de uma rede de comunicação aonde todos são gerenciados através de um *middleware,* oferecendo

recursos a uma aplicação única. Essa composição de um sistema distribuído complementa uma estrutura computacional de uma nuvem (MARINESCU, 2013).

Algumas características de um sistema distribuído são compartilhadas para uma infraestrutura de computação em nuvem. Contudo, em regra, o sistema distribuído é composto por componentes autônomos, como a programação, a gestão de recursos e de segurança, isso tudo sendo implementado por sistemas independentes (MARINESCU, 2013).

Em geral, o conjunto desses componentes serão convertidos no sistema distribuído, aonde neste caso, sendo importante e necessário pontos de controle em todos os recursos, pois poderão ocorrer pontos de falha. E é por este motivo que o sistema distribuído possuí uma alta escalabilidade, sendo possível adicionar recursos a todo instante, com o objetivo de manter a disponibilidade dos recursos para toda a estrutura (MARINESCU, 2013).

De acordo com Marinescu (2013), o *middleware* responsável em intermediar a comunicação entre outras aplicações em um sistema distribuído, se torna desejável que este *middleware* atenda alguns requisitos, de acordo com o autor são eles:

Transparência no acesso – todos os locais e objetos de acessos são realizados por chamadas de operações idênticas.

Transparência na localização – os objetos remotos devem ser acessados sem o conhecimento de sua localização.

Transparência na concorrência – processos que são executados em um sistema distribuído devem ter informações compartilhadas entre si, sem que sejam prejudicados.

Transparência de replicação – é importante que várias instâncias de informações sejam usadas para aumentar a confiabilidade da aplicação, fazendo com que um único objeto seja identificado pelo usuário ou aplicação.

Transparência em falhas – falhas que ocorrem no sistema devem ser imperceptíveis aos usuários e para as aplicações que rodam na infraestrutura.

Transparência em Migração – objetos disponíveis para as aplicações podem ser movidos, sem afetar o processamento dos processos.

2.2.1 **Tipos de sistemas distribuídos**

Sendo um conjunto de recursos que envolve um nível alto de complexidade, pode existir alguns modelos peculiares de sistemas distribuídos. Alguns serão apresentados nas seguintes seções.

2.2.1.1 Cluster

É um sistema interligado por uma mesma rede de comunicação, composto por unidades de processamento, com o objetivo de unir processamento para que cargas de trabalho sejam paralelizadas e distribuídas em sua estrutura computacional (ALECRIM, 2013).

As unidades de processamento que compõe um cluster são denominadas cluster. Em uma implantação, a quantidade dessas unidades de processamento é limitada somente pelo espaço físico, porém é uma arquitetura altamente escalável, sendo totalmente abstrato ao usuário, se apresentando para o usuário como uma única unidade de processamento (ALECRIM, 2013).

Um *cluster* pode conter equipamentos específicos de uma estrutura profissional, como servidores, mas também é possível utilizar equipamentos convencionais como estações de trabalho. A transparência de toda a arquitetura física é mantida pelo uso de um *middleware* que está intensamente ligado com o sistema operacional hospedado na estrutura. E a comunicação interna na infraestrutura pode ser definida por bibliotecas como a MPI (*Message Passing Interface)* (ALECRIM, 2013).

Como uma estrutura de cluster depende muito de um alto fluxo de comunicação, ela se dá entre os nodos da estrutura através de tecnologias de rede

local, como os padrões *gigabit ethernet, fast ethernet*, que possuem um custo de implementação mais em conta. Mas existe a possibilidade de utilização de conexões físicas específicas para *clusters*, como as conexões *InfiniBand* e *Myrinet* que tem um desempenho superior aos padrões mais básicos de comunicação (ALECRIM, 2013).

Um *cluster* apresenta alta portabilidade na alocação de serviços em sua estrutura, isto se resume em utilizar toda a sua capacidade em serviços para a computação em nuvem. E além disso, é possível tornar uma estrutura de *cluster* em um ambiente voltado para virtualização de ambientes e sistemas operacionais. Hoje estes grandes serviços vêm sendo largamente utilizado na área da tecnologia da informação.

Existem diversos tipos e aplicabilidades de *cluster*, sendo os mais vistos: *cluster* de alto desempenho, *cluster* de alta disponibilidade, e *cluster* de balanceamento de carga. A utilização destes tipos de *cluster* pode abranger grande parte dos serviços ligados a tecnologia da informação. Estes três modelos serão particularizados nas seções seguintes (ALECRIM, 2013).

- Cluster de alto desempenho

A utilização de *cluster* de alto desempenho é para finalidade de altos níveis de processamento. Todos os recursos disponíveis devem ser direcionados para um processamento de determinada aplicação, com o propósito de buscar resultados em um curto intervalo de tempo e com precisões nos resultados (ALECRIM, 2013).

Um exemplo prático da utilização de *cluster* de alto processamento é no ramo da meteorologia, pois é necessário um alto processamento para se alcançar em tempo os dados climáticos de uma determinada região.

- Cluster de alta disponibilidade

O princípio da utilização de *cluster* de alta disponibilidade é manter serviços sempre estáveis e ativos. Sua estrutura computacional é projetada para alcançar níveis próximos a 100% de disponibilidade durante o ano (ALECRIM, 2013).

Os níveis de disponibilidade são mantidos através de serviços de monitoramento e gerenciamento da estrutura do *cluster*. Esses mecanismos devem ser capazes de identificar problemas na estrutura e tentar contorna-los através de gatilhos de controle. Além do controle de serviços e nodos, há um grande nível de replicação destes componentes. Todos esses recursos trabalham em conjunto para garantir a disponibilidade da estrutura, porém, se entende que é possível aparentar um grau de perda de desempenho (ALECRIM, 2013).

- Cluster para balanceamento de carga

O objetivo básico de um *cluster* de balanceamento de carga é que todas as tarefas aplicadas à infraestrutura sejam uniformemente divididas entre os nodos que compõem a estrutura (ALECRIM, 2013).

Para garantir uniformidade na distribuição de processamento, um *cluster* deste tipo conta com algoritmos especializados que gerenciam toda essa distribuição, buscando manter um equilíbrio aceitável para a aplicação e não somente para a estrutura de processamento (ALECRIM, 2013).

A utilização de *clusters* com a finalidade de balancear a carga da estrutura é comumente visto em hospedagens de sites de internet, onde é grande a demanda de acessos e requisições de conteúdo em um curto intervalo de tempo (ALECRIM, 2013).

2.2.1.2 Grid

Uma *grid* parte da ideia básica de um *cluster*, que são vários computadores compartilhando recursos entre tarefas para busca de um resultado único. No entanto, este tipo de processamento vai além de ambientes locais.

A infraestrutura de uma *grid* tem o propósito de unir recursos de qualquer tipo de arquitetura, indiferentemente da sua localização, com a finalidade de oferecer poder computacional em grande escala. Essa capacidade de processamento é distribuída e interligada através de redes de comunicação, sendo WAN, MAN, ou LAN.

Além dos recursos físicos, em uma *grid* é necessário um *middleware* para gerência e integração dos recursos. Neste caso, o *middleware* é responsável em negociar os recursos da estrutura e prover a heterogeneidade da estrutura.

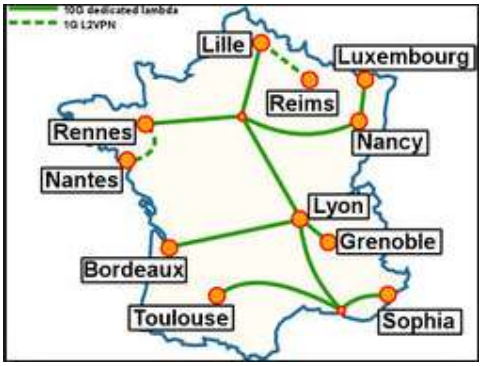
Alguns exemplos de grids:

- Grid5000

É uma infraestrutura computacional distribuída em 11 (onze) sites[2] diferentes em toda a França, e Porto Alegre/RS no Brasil está se tornando a primeira estrutura fora da França.

Os 11 (onze) grandes centros de processamento da França se uniram para montar uma estrutura computacional com uma capacidade de 1926 processadores da linha Intel Xeon e AMD Opteron, totalizando em 7782 *cores* de processamento. A estrutura montada visa fornecer uma estrutura de processamento em larga escala para estudar, simular, emular, e experimentar grandes sistemas distribuídos.

A Figura 09 mostra um mapa da Grid 5000 contendo de forma gráfica a interligação dos centros de processamento.



FONTE: Grid 5000

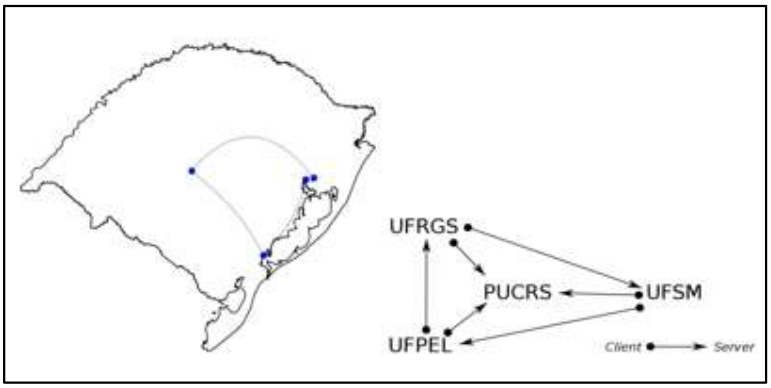
Figura 9: Mapa da estrutura da *grid* 5000

[2] Sítios de máquinas. *Clusters.*

- GridRS

É um projeto que desenvolveu uma *grid* envolvendo estruturas computacionais das universidades federais do Rio Grande do Sul – UFPEL, UFRGS, PUCRS e UFSM. Toda sua estrutura abrangente tem como objetivo fortalecer a comunidade de alto desempenho rio-grandense, e ainda fornecer um instrumento científico para pesquisas em uma plataforma de computação heterogênea composta por 136 CPUs,

A Figura 10 apresenta um mapa com a estrutura de interligação das instituições.



FONTE: GridRS
Figura 10: Mapa da GridRS

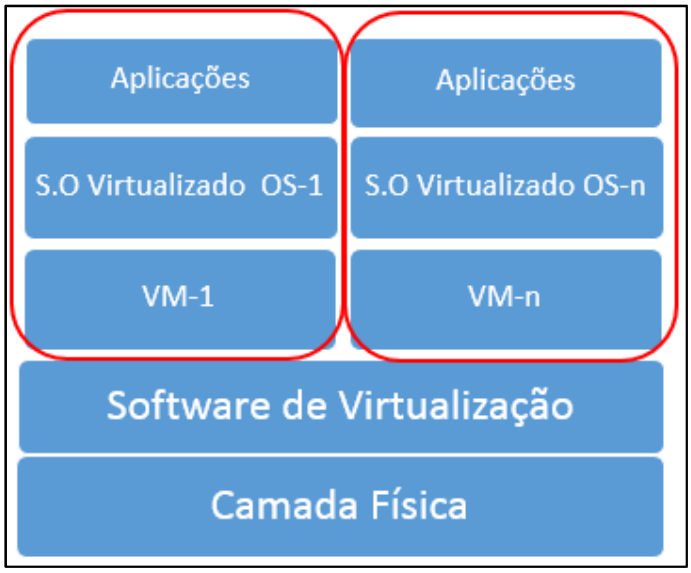
2.3 VIRTUALIZAÇÃO

Os recursos de virtualização que hoje são empregados na computação moderna permitem que vários sistemas operativos sejam implantados em um único *hardware*. Isso é possível respeitando claramente a capacidade física do equipamento.

Ao serem implantados sistemas operacionais em ambientes virtuais, os sistemas de monitoramento de máquina virtual (*hypervisor),* como são chamados, simulam um ambiente físico de tal maneira que o sistema operacional hospedado

consiga comunicar se com o *hardware* do equipamento. O monitor de máquina virtual cria de maneira eficiente uma camada de abstração entre o sistema operacional e a camada física do equipamento, fazendo com que o sistema operacional hospedado trate o ambiente virtual como se fosse o físico. Essa camada também controla o ambiente de cada hospedeiro, delimitando os recursos físicos destinado a cada hospedeiro (MATHEWS, et al., 2009).

Na Figura 11, é mostrado um exemplo genérico de virtualização, com a figura, podemos observar a primeira camada como sendo a de *hardware*, e logo em seguida a camada de virtualização criada pelo monitor de máquina virtual e, posteriormente os sistemas operacionais já virtualizados dentro do círculo vermelho.



Adaptado de MATHEWS, et al., 2009.

Figura 11: Exemplo genérico de virtualização.

2.3.1 **Benfícios do uso da virtualização**

Sendo a virtualização um recurso de alocação de sistemas em um único equipamento físico, isto pode trazer vários benefícios computacionais para os utilizadores deste recurso. A seguir serão relatados alguns benefícios sobre ela.

Para um desenvolvedor que necessite de um ambiente para testar suas aplicações em desenvolvimento, um ambiente virtual pode trazer resultados

importantes sem que o sistema seja aplicado à um ambiente de produção impactando em transtornos desnecessários (MATHEWS, 2009).

Ambientes rodando maquinas virtuais de qualquer natureza podem se recuperar rapidamente de um desastre, pois diferentemente do ambiente físico, o virtualizado periodicamente pode ser feito cópias de todo seu conjunto, onde em um casual sinistro que comprometa a disponibilidade dos serviços, esta, pode rapidamente ser retornada ao seu estado funcional (SOSINSKY, 2011)

A virtualização além de consolidar vários serviços e sistemas operacionais que podem ser administrados independentemente em um único *hardware*, por sua vez, podem se tornar altamente disponíveis, já que uma máquina virtual pode ser remanejada de maneira que os seus serviços não sejam interrompidos (MATHEWS, 2009).

Além destes fatores citados anteriormente, o objetivo inicial de se virtualizar sistemas operacionais, é para obter um melhor aproveitamento dos recursos que o *hardware* disponibiliza. Pois em regra, grande parte dos sistemas operacionais não virtualizados, não chegam a ocupar todo o potencial disponível de um processador, deixando ocioso um grande poder de processamento que poderia ser reaproveitado em conjunto com outros sistemas (SOSINSKY, 2011).

2.3.2 **Tipos de virtualização**

Na implantação da virtualização de ambiente existem tipos de plataformas que podem ser aplicadas dependendo da necessidade, aqui serão relatadas 4 (quatro) delas: Emuladores, virtualização completa, paravirtualização e virtualização a nível de sistema operacional. Todas elas serão particularizadas a seguir.

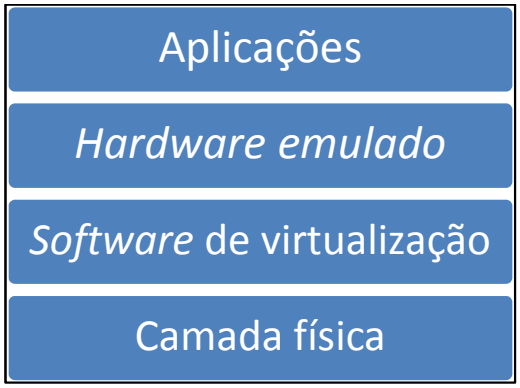
2.3.2.1 Emuladores

De acordo com Mathews (2009), neste tipo de plataforma o conjunto de *hardware* virtual é totalmente divergente da arquitetura física, a plataforma é inteiramente simulada pelo virtualizador, de forma que o hóspede, onde comumente

rodará uma aplicação específica, execute todas suas funções sem a necessidade de alterações.

Este tipo de virtualização é usado quando há necessidade de testes de algum determinado *software* em desenvolvimento em um *hardware* específico, o emulador neste caso, fará com que seja reproduzido um ambiente levando em conta os requisitos que determina a aplicação, este ambiente muitas vezes é totalmente distinto da arquitetura real do computador.

Na Figura 12, podemos perceber na base da imagem a camada física da infraestrutura. Logo a seguir a camada do *software* virtualizador, e posteriormente o *hardware* emulado pelo *software* e as aplicações do usuário.



Adaptado de Mathews. 2009

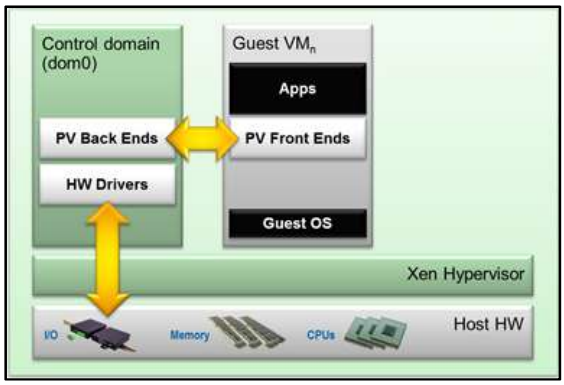
Figura 12: Exemplo de uma virtualização por emulação

2.3.2.2 Virtualização completa

Segundo Mathews (2009), este modelo de virtualização, é muito semelhante ao modelo Emulador, com a diferença de que os sistemas operacionais hospedados da máquina virtual, são projetados para rodar na mesma arquitetura do ambiente físico do servidor hospedeiro. Sendo assim, o ambiente virtual estando compatível com a arquitetura física, o *hypervisor* permite que a máquina virtual execute algumas instruções diretamente no *hardware*.

2.3.2.3 Paravirtualização

Neste modelo de virtualização é necessária uma modificação do sistema que será virtualizado Neste caso, consiste em que o *hypervisor* exporta uma versão modificada do *hardware* físico para o sistema virtualizado, para facilitar e acelerar atributos básicos do sistema (SOSINSKY, 2011).



Fonte: Xen Project.2012

Figura 13: Exemplo da paravirtualização

2.3.2.4 Virtualização ao nível do sistema operacional

Essa plataforma de virtualização pode ser encarada como um método de "quase virtualização". Nesta modalidade é dispensado o uso de monitores de máquinas virtuais, toda a virtualização é aplicada sobre o próprio sistema operacional (MATHEWS, 2009).

Ao ser aplicado a virtualização por container (como também é conhecida), o sistema operacional que hospedará as máquinas virtuais compartilhará recursos diretamente com cada sistema virtualizado. Cada sistema será conhecido como um único ambiente, mas com a vantagem muito específica da não duplicação de recursos do sistema operacional, dispensando a necessidade de um grande volume de memória de armazenamento. Também é importante ressaltar que a forma de virtualização faz com que arquivos binários e bibliotecas do sistema operacional que hospeda as VM's (*Virtual Machine)* seja compartilhado entre ambiente virtual e não virtual (MATHEWS, 2009).

Um dos principais objetivos na virtualização de sistemas, é ter a possibilidade de uso de diferentes sistemas operacionais um mesmo *hardware*, sendo assim, a virtualização por container não permitindo este recurso.

Outro empecilho neste uso de virtualização, é que o mesmo não apresenta um isolamento apropriado e disponibilidade real dos recursos de *hardware*. Se cada sistema apresentar uma configuração específica de recursos de *hardware* e com o decorrer do uso poderá vir a ocorrer um alto uso de processamento em um algum determinado sistema, o restante poderá ser comprometido, podendo acarretar na perda de desempenho dos demais (MATHEWS, 2009).

2.3.3 **Ferramentas de Virtualização**

As ferramentas de virtualização são usadas para implantar e gerenciar os sistemas virtualizados utilizando os tipos de virtualização escolhidos de acordo com a necessidade de cada ambiente. Estão disponíveis várias ferramentas de virtualização, onde algumas serão abordadas nesta seção.

2.3.3.1 Xen

Criado através de um projeto de pesquisa da universidade de Cambridge, ao final dos anos 90, Keir Fraser e Ian Pratt desenvolveram o primeiro código *open source* do *hypervisior* Xen (XEN Project, 2013).

O Xen basicamente se estabelece acima da camada física, e desta forma, para que não haja modificações nos sistemas operacionais que serão virtualizados, cria um ambiente virtual de modo que o sistema hospedado perceba um *hardware* real. O princípio de virtualização do Xen, é através de domínios de máquinas virtuais, sua principal função é abstrair a capacidade total do *hardware*, e em parcelas imparciais/uniformes ceder para os ambientes virtuais, fatias do ambiente físico para processamento nos ambientes virtuais (MATHEWS, 2009).

Todo sistema operacional ao ser instalado em uma máquina física deve conhecer corretamente todos os dispositivos que estarão disponíveis para uso do sistema e do próprio usuário. Assim, o sistema deve ter acesso as especificações

técnicas dos dispositivos. No ambiente virtualizado pelo Xen, esses dispositivos muitas vezes não ficam em contato direto com o sistema hospedado, e o *hypervisior* entrega a este sistema um dispositivo identificado como genérico, mas que mantêm as características técnicas físicas do dispositivo físico. Como discos de armazenamento ou adaptadores de rede, aparecem no sistema hospedado como dispositivos genéricos, isto tudo para tornar o ambiente virtual o mais compatível possível com outras arquiteturas físicas (MATHEWS, 2009).

Sistemas operacionais por sua vez, executam processos com diversos níveis de privilégios. “Na verdade a arquitetura x86 possui 4 níveis de privilégios chamados de Anéis de Proteção.” (Mathews, 2009, pág 52).

Mathews (2009) explica que nas arquiteturas x86 existem 4 níveis de privilégios de execução de instruções, sendo elas a 0 com maior grau de privilégio e respectivamente 1, 2 e a 3 com menor grau de privilégio. Em sistemas operacionais não virtualizados, é usado somente o nível 0 para instruções do sistema, e o 3 para processos de usuário.

Usando os anéis de privilégio designados para elevar os níveis de tolerância a falhas, o *hypervisior* cria um sistema de multicamada usando os 4 (quatro) níveis para administrar as requisições que cada hospede fará à camada física. O *hypervisior* fica em contato direto com a camada física, executando as instruções no grau máximo de privilégio. Sistemas operacionais alocados no sistema de virtualização executam no nível 1, e por seguinte, programas de usuários destes ambientes virtuais executam na camada 3 do nível.

Particularmente, o *hypervisior* do Xen não administra o seus hospedes sozinho. Com o intuito de criar uma proteção adicional, e ainda ter privilégios mais elevados na execução de processos, o Xen cria o *Domain0,* onde auxilia o *hypervisior* a controlar os ambientes virtuais para uma melhor eficiência na comunicação entre dispositivos de armazenamento e dispositivos de rede. Não bastando somente o *Domain0*, o *hypervisior* injeta para dentro de cada sistema hospedado um *driver* que é responsável em tornar possível a comunicação entre os dispositivos de *hardware*, e com isto ele realizará de forma adequada cada requisição do sistema virtualizado

aumentando muito o grau de desempenho e estabilidade do sistema operacional. Toda a comunicação feita entre o *driver*, e a camada física e outros domínios, é feita pelo transporte assíncrono de memória compartilhada (MATHEWS, 2009).

Além de *hypervisior,* o Xen apresentada outros componentes e *daemons* importantes que auxiliam no funcionamento e gerenciamento do sistema de virtualização, onde serão abordados a seguir:

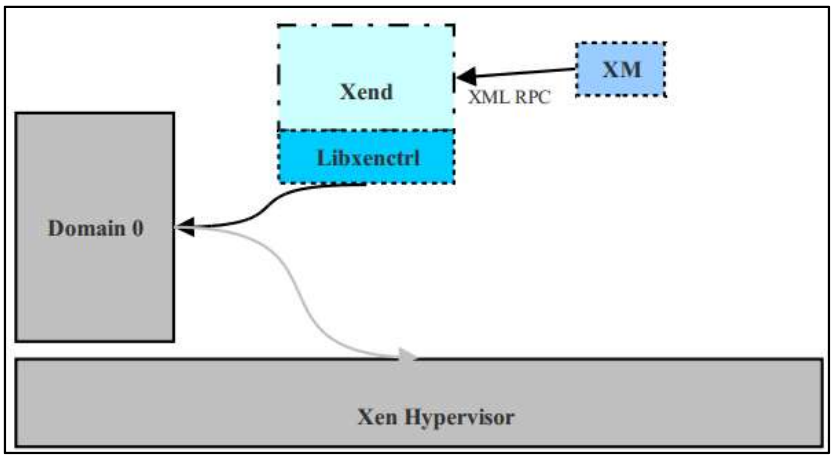
***Domain* 0** – de acordo com Xen Project (2009), é o primeiro ambiente de um servidor que hospeda a virtualização dos sistemas. Sendo um *Kernel* de um sistema Unix, o *domain0* se faz necessário para execução dos diversos sistemas que serão virtualizados no servidor, este sendo o primeiro modulo a iniciar no momento que um servidor de virtualização inicia. Dois componentes atrelados ao *domain0* que são importantes em apoiar o gerenciamento de rede e do disco local, são os denominados *drivers*, por exercer a função de gerenciar a comunicação com os dispositivos de rede e processamento dos dados do disco, requisitando leitura e gravação dos setores do discos disponibilizando as informações ao sistema.

***Domain U* –** este é o componente representado pelas máquinas virtuais que o Xen irá virtualizar, diferentemente do *domain*0, o *domain*U não tem acesso direto ao *hardware* do equipamento. Existem duas representações que caracterizam os hóspedes em uma plataforma controlada pelo Xen, o *domain*U PV – que são máquinas virtuais rodando em formato de paravirtualização e que seus sistemas operacionais são modificados conforme rege a regra da plataforma de paravirtualização. Por sua vez, reconhecem a existência de uma camada de abstração física e que não tem permissões suficientes para terem acesso direto ao *hardware,* e reconhecendo que os recursos físicos são compartilhados com outros hospedeiros. O *domain*U HVM – são hóspedes que rodam seu sistema operacional inalterado, respeitando as regras da plataforma de virtualização completa, rodando em um ambiente onde o sistema operacional não reconhece a existência de outros hospedeiros no mesmo *hardware* e interpreta o ambiente como sendo dedicado ao seu processamento (Xen Project, 2009).

***Xend* –** de acordo com Xen Project (2009) o *deamon* Xend auxilia o *hypervisior* a gerenciar as requisições do hospedeiro para a camada física.

> "O daemon Xend é uma aplicação python que é considerado o gestor do sistema para o ambiente Xen. Ele aproveita a biblioteca libxenctrl para fazer solicitações do hypervisor Xen. Todos os pedidos processados pelo Xend são entregues a ele através de uma interface RPC XML pelo Xm ferramenta." (Xen Project, 2009, pág 6. **Tradução nossa**).

Conforme a Figura 14 é demonstrada como o *deamon* comunica-se com seus componentes.



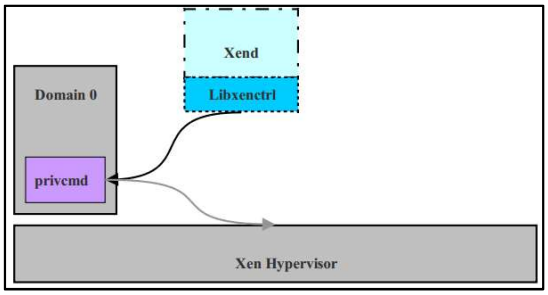
Fonte: Xen Project, 2009

Figura 14: Hierarquia de comunicação de *Deamos*.

Xm – Utilitário de linha de comando que permite o administrador gerenciar a ferramenta de acessos externos (Xen Project, 2009).

Xenstored – mantém como forma de registros informações como eventos de canais entre o *Domain*0 e os domínios hospedeiros.

Libxenctrl - biblioteca responsável em fornecer comunicação com ajuda de um *driver,* entre o *hypervisior* através do *domain0* (Xen Project, 2009).



Fonte: Xen Project, 2009

Figura 15: Hierarquia de comunicação Libxenctrl.

Quemu-dm – *deamon* que trata todas as requisições de rede e disco solicitadas pelas máquinas virtuais totalmente virtualizadas (Xen Project, 2009).

Xen Virtual *Firmware* - corresponde a BIOS de um ambiente não virtualizado, Xen Virtual *Firmware* é inserido em cada sistema para controle das instruções de inicialização do equipamento (Xen Project, 2009).

2.3.3.2 Vmware

Fundada em 1998 por um grupo de profissionais, a empresa desde o início esteve atuando no mercado de ferramentas de virtualização de ambientes computacionais, mas a procura em acompanhar a evolução do mercado de TI fez a empresa buscar novas oportunidades de investimentos e pesquisas e, ultimamente vem investindo fortemente em serviços voltados para computação em nuvem.

Com vários produtos lançados no mercado, o *hypervisor* que a empresa comercializa atualmente é o ESXi e o ESX. Basicamente, a diferença que existe entre as versões é o que ESXi não permite nos servidores físicos a administração, monitoramento, backups ou um gerenciamento mais avançado dos ambientes virtualizados, tudo deve ser feito através de um software remoto. Todos os produtos são protegidos por direitos autorais, possuindo licenciamentos que são voltados para pequenas até grandes empresas. No entanto, existe uma alternativa para o licenciamento livre de alguns softwares mas que acabam tornando as características e funcionalidades bem limitadas do produto (VMWARE, 2009).

*Hypervisor (*ESXi/ESX) – possui componentes que se encontram divididos em 2 grupos, a camada da virtualização do ambiente, e as VM's. Em cada um dos grupos existem componentes adicionais que garantem a eficiência e a funcionalidade do virtualizador (MAILLÉ, et al., 2013).

Na camada de virtualização dos virtualizadores ESXi e ESX, existe o componente VMKernel, ele é responsável em criar a abstração do *hardware* para que as máquinas virtuais consigam executar suas funções. Isso faz com que os virtualizadores em questão, garantem ao sistema virtual os recursos necessários para garantir seu desempenho neste ambiente, controlando tempo e espaço dos recursos de disco, memória, rede e processador da arquitetura física. Além disso, ao mesmo tempo garantindo os limites entre outros ambientes virtualizados na mesma arquitetura. Na Figura 16 é demonstrada a arquitetura dos componentes do virtualizador (MAILLÉ, et al., 2013).

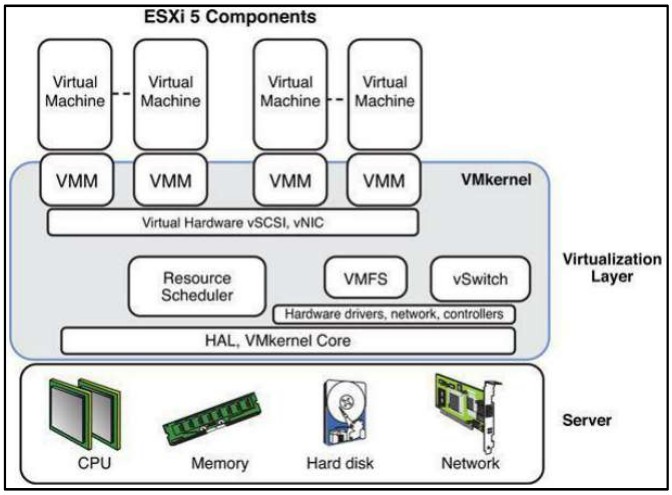


Figura 16: Exemplo da arquitetura dos componentes do ESXi 5.
Fonte: MAILLÉ, et al., 2013.

VM (*Virtual Machine)* – consiste na estrutura composta pelos sistemas operacionais virtualizados na arquitetura física. Em um servidor que se encontra algum dos virtualizadores, a quantidade de VM's somente se restringe a capacidade operacional do *hardware* hospedeiro, e a licença de uso que determinará o quanto o virtualizador tornará disponível os componentes físicos aos ambientes virtuais.

Na camada das VM's, existe um aglomerado de arquivos que são necessários para a execução dos hospedes virtuais na plataforma do virtualizador. Muitos destes arquivos são considerados arquivos de configuração dos hospedeiros, como o vmx, e já alguns são tratados como um banco de dados, como flat-vmdk. Conforme Maillé (2013), alguns dos principais componentes serão descritos a seguir:

- **VMDK:** é o arquivo de configuração que define os parâmetros do disco virtual do sistema virtualizado.
- **FLAT-VMDK:** através de um formato específico, este é o arquivo que ficam armazenados todos os dados do ambiente virtual fazendo a função de um disco rígido em condições não virtualizadas.
- **VMX:** é o arquivo que parametriza as características de uma máquina virtual. Nele estão contidas as informações como quantidade de memória, tamanho de disco rígido, endereço MAC da placa de rede. É o primeiro arquivo criado no momento da criação de uma máquina virtual.
- **NVRAM:** tem as características da BIOS de um ambiente não virtual.
- **VMSS:** uma máquina virtual no momento em que ela é suspensa, este arquivo recebe todos os dados que estavam alocados em sua memória RAM alocada ao sistema hospedeiro. No momento em que a máquina virtual volta ao seu estado de operação normal, todas as informações contidas neste arquivo voltam a memória RAM do sistema operacional virtual.
- **VSWP:** arquivo definido como memória SWAP da máquina virtual.

Os produtos proprietários da empresa VMware – ESX e ESXi – buscam atender as diversas necessidades na utilização da virtualização de recursos físicos usando servidores profissionais.

Em sua literatura, é possível encontrar uma grande variedade de recursos e funções que podem ser aplicadas e configuradas em seus *softwares* para virtualização. Estes podem ser destinados ao gerenciamento e manutenção dos ambientes virtuais e, definições de formas de utilização dos recursos físicos que decidirão os tipos de processamento exercido pelos ambientes virtuais.

Cada um dos recursos do VMware busca atender um cenário específico, uma arquitetura específica, e uma necessidade específica do usuário que utiliza os *softwares* ESX e ESXI.

No que diz respeito ao uso dos recursos físicos, fatores como o armazenamento, o uso do processador e uso da memória RAM farão a diferença no desempenho, se administrados corretamente

2.3.3.3 KVM (Kernel-based Virtual Machine)

Surgiu em outubro de 2006, sendo introduzido junto ao *kernel* 2.6.20, seu desenvolvimento partiu de Avi Kivity, o Linux KVM acabou se transformando em um novo conceito de virtualização. Seu método aplicado se diferenciava das ferramentas existentes no mercado, hoje sendo mantido pela Red Hat Interprise Linux. A notável diferença da aplicação, é devido ao local em que o KVM opera seus serviços de virtualização. Ao aplicar o Linux KVM, a ferramenta é capaz de ocupar pouco espaço do disco do servidor, pois existe uma grande reutilização de recursos do sistema operacional (Linux KVM, 2013).

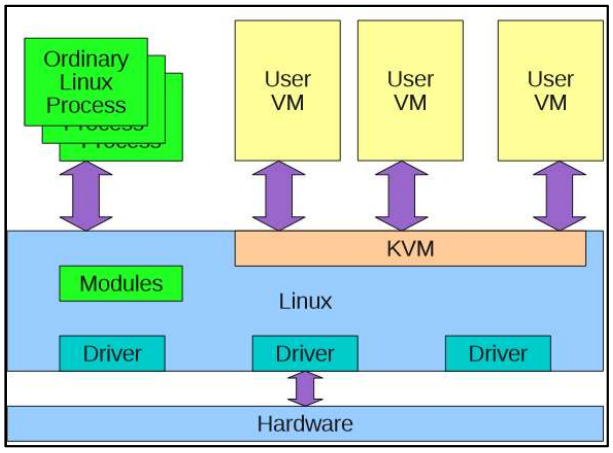
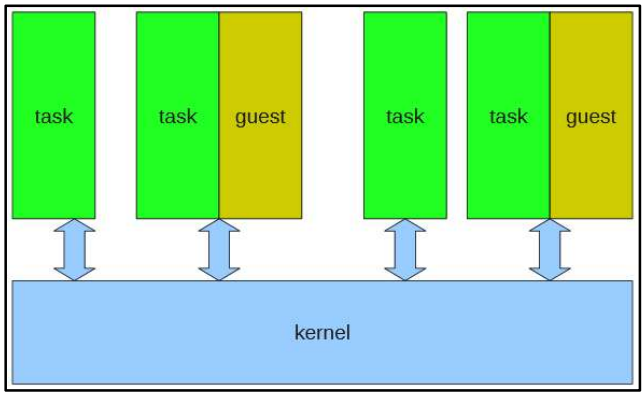


Figura 17: Arquitetura Geral da estrutura do KVM.
Fonte: BURNS, 2010

Como a Figura 17 está demonstrando, a estrutura de virtualização usando o *hypervisor* KVM se assemelha muito a um sistema não virtualizado. Isso é pela maneira com que o KVM está inserido ao sistema, sendo em conjunto com o *kernel*

do Linux, fazendo com que o virtualizador utilize dos módulos do próprio sistema operacional para operar as máquinas virtuais.



Fonte: BURNS, 2010.

Figura 18: Arquitetura geral de um sistema operacional.

Na figura 18, é visualizado a arquitetura de um sistema operacional não virtualizado, onde é possível visualizar de forma simples, a lógica de comunicação dos processos com o *kernel* do sistema operacional.

Em comparação com as duas imagens acima, é notável que um sistema virtualizador, possui módulos de controle para as operações dos sistemas virtualizados, e a grande preocupação é a garantia de desempenho que um virtualizador deve oferecer ao sistema hospedado. No KVM, existe componentes que trabalham em conjunto para prover as operações ao sistema operacional virtualizado. A seguir os principais componentes e funções são descritos:

Virt-manager: é uma interface de gerenciamento de maquinas virtuais. O virt-manager é utilizado em conjunto com o KMV, consegue criar e gerenciar máquinas virtuais através de uma interface gráfica, evitando a complexidade de comandos extensos (VIRT-MANAGER, 2013).

Libvirt: trabalha em conjunto com os módulos responsáveis pela virtualização dos sistemas operacionais. O libvirt possui um conjunto de *softwares* e funções que possibilitam o gerenciamento das máquinas virtuais. Suas principais funcionalidades exercidas para o ambiente é o gerenciamento de dispositivos funções de I/O como o

armazenamento e a rede destes ambientes. Com ele é possível aplicar nas máquinas virtuais operações do tipo *start, stop, pause, salve, restore, migrate* (LIBVIRT, 2013).

Qemu: o qemu é o modulo responsável pela virtualização do ambiente, onde emula um ambiente físico de maneira que o torne imperceptível para o sistema virtual, tornando possível a execução dos sistemas operacionais com o mesmo propósito de um ambiente físico comum. De acordo com a necessidade do ambiente, com o qemu é possível até mesmo virtualizar arquiteturas específicas para uma determinada aplicação, as quais são totalmente divergentes do servidor físico que aloca a máquina virtual (QEMU, 2013).

KVM Kernel Module: é um modulo privado que gerencia entradas e saídas das máquinas virtuais (GOTO, 2011).

Linux Kernel: necessário para que os módulos iniciados no momento de uma virtualização sejam administrados pelo Linux Kernel (GOTO,2011).

Extended Page Table (EPT): é um recurso desenvolvido pela Intel que contribuiu para um ganho de desempenho em ambientes virtualizados. Ele se resume na tabela de endereçamento de memória de uma máquina virtual. Este método garante a sincronia de endereços lógicos endereçados pelo virtualizador a pedido do sistema virtual. Desta forma, evita a sobrecarga de processamento do *hardware*, pois o próprio sistema virtual irá consultar a sua tabela de endereços diretamente no *hardware* (GOWDA, 2009)

VT-d: é uma solução usada pelos virtualizadores para garantir de forma eficiente o endereçamento de dispositivos físicos para o ambiente virtual. Neste recurso, os endereços podem ser cedidos diretamente para o sistema virtual (GOTO, 2011).

Virtio: na aplicação da virtualização completa em sistemas operacionais, um dos grandes problemas para o *hipervisor* é o baixo desempenho da comunicação de dispositivos físicos, como a rede e o armazenamento. O virtio entra neste processo

de comunicação para melhorar o desempenho de rede e disco nos ambientes virtuais, criando uma interface onde a comunicação é direta com o meio físico (LIBVIRT, 2014).

Kernel SamePage Merging (KSM): permite o compartilhamento de endereços de memória idênticos entre outros sistemas virtualizados, afim de economizar recursos e aumentar o desempenho do processamento (GOTO,2011).

2.3.3.4 OPENVZ

A ferramenta OpenVZ é umas das que permitem a virtualização baseada em container. Sendo ela implementada em sistemas operacionais Linux, permitindo a criação de máquinas virtuais no próprio sistema operacional. O kernel do sistema OpenVZ, é um kernel Linux modificado que acrescenta as funcionalidades de virtualização (KOLYSHKIN, 2006).

Neste tipo de implementação usando a ferramenta OpenVZ, os sistemas criados e gerenciados pela ferramenta, são chamados de VE (*Virtual Environment)*, são ambientes virtuais que aonde o sistema Linux é virtualizado, possuindo as mesmas características de um ambiente não virtualizado, com as mesmas estruturas de arquivos, e com sua própria estrutura de processos (KOLYSHKIN, 2006).

Devido ao sistema de virtualização por container compartilhar recursos de *software* do sistema operacional hospedeiro com sistemas operacionais hospedado, o OpenVZ é composto por funções que auxiliam o controle destes recursos compartilhados, De acordo com Kolyshkin (2006) alguns destas funcionalidades são:

- *Two-level disk quota* – permite ao administrador do servidor controlar as quotas de disco para os sistemas virtualizados.
- *Fair CPU Scheduler* – é o um escalonador de máquinas virtuais da ferramenta, mas agindo no controle da fatia de tempo que cada máquina virtual usará o processador.
- *User Bean Conuters* – este é um controlador dos limites que cada ambiente virtual deve ter, sendo de memória e até mesmo de alguns objetos do kernel.

A grande parte de aplicativos que existem para sistemas operacionais Linux são compatíveis nos ambientes virtualizados pelo OpenVZ, até mesmo como aplicações de grandes portes como um gerenciador de banco de dados. Uma das funcionalidades que OpenVZ ainda carrega consigo é a alta escalabilidade que ele possui, é possível aumentar a capacidade de um ambiente virtual em produção, não importando a quantidade de recursos que serão direcionados para tal ambiente virtual (KOLYSHKIN, 2006).

2.4 COMPUTAÇÃO EM NUVEM

A computação em nuvem abrangendo toda sua estrutura física, pode ser entendida como uma imensa computação em ambientes de processamento com servidores físicos ou lógicos, onde todo o resultado deste processamento é entregue ao usuário através da internet (TAURION, 2009).

A estrutura da computação em nuvem baseia-se na entrega de serviços como infraestrutura, ou softwares para usuário que não possuem esses recursos em suas dependências, ou necessitam temporariamente. A computação em nuvem, financeiramente se apresenta aos usuários com valores mais acessíveis, na sua modalidade de locação sendo utilizada nuvens públicas para oferecer os recursos (infraestrutura, *software*, ...) necessário para a utilização do usuário. Pois esses valores contrapondo com um custo total de investimento do que se é necessário, em alguns casos acaba sendo financeiramente inviável. É este um dos principais motivos para o uso destes serviços, o baixo investimento financeiro em alugar serviços deste porte, aonde as modalidades de cobrança de provedores, baseiam-se no *pay-per-use*, ou seja, pago pelo uso.

Alguns detalhes podem ser característicos destes ambientes computacionais, sendo repassados a seguir conforme Taurion (2009) nos apresenta:

- A computação em nuvem cria uma ilusão da disponibilidade de recursos infinitos, acessíveis sobre demanda.
- A computação em nuvem, pela visão do usuário, elimina a necessidade de adquirir e provisionar recursos antecipadamente.

- A computação em nuvem oferece elasticidade, tornando os recursos de processamento escaláveis aos usuários.
- O pagamento dos serviços em nuvem é pela quantidade de recursos utilizados (*pay-per-use).*

2.4.1 **Modelos de Implantação**

Nesta seção é abordado aos tipos básicos de nuvem, sendo eles: Nuvens públicas, nuvens privadas, nuvens híbridas e nuvens comunitárias.

2.4.1.1 Nuvens públicas

As nuvens públicas são voltadas para um público mais geral. Empresas provedoras de serviços voltados para a computação em nuvem são responsáveis em disponibilizar toda a infraestrutura necessárias para a utilização de seus clientes. Neste caso, não somente um cliente faz o uso de seus serviços, mas sim, diversas outras organizações utilizam os serviços de computação em nuvem compartilhando a mesma estrutura (TAURION, 2009)

São vistos como grandes exemplos de nuvens públicas as empresas Google e Amazon, que disponibilizam seus serviços de computação em nuvem para uma grande variedade de clientes através da internet. Neste contexto, as empresas investem em ambientes e equipamentos profissionais para garantir as necessidades dos clientes usando uma infraestrutura única. Ainda que as empresas provedoras de serviços se preocupem com a segurança nestes ambientes, ela é o principal fator que torna decisivo a utilização da nuvem pública por uma empresa que tende adotar este tipo de serviço.

2.4.1.2 Nuvens privadas

Neste modelo de implantação, as nuvens privadas são uma alternativa para organizações que desejam ter seus *softwares* centralizados e em sua propriedade. Isso torna este modelo de implantação exclusiva para determinados usuários e para determinadas aplicações de uma organização. As organizações que adotam as

nuvens privadas, são responsáveis por toda a implantação e manutenção da infraestrutura da nuvem e das aplicações que rodam neste ambiente. Em alguns casos, elas terceirizam esta manutenção (MARINESCU, 2013).

2.4.1.3 Nuvens híbridas

O modelo de nuvens híbridas parte do mesmo princípio dos outros tipos de nuvem com a diferença que a infraestrutura física compartilhará entre vários modelos de implantação, como nuvens privadas, públicas ou comunitárias (MARINESCU, 2013).

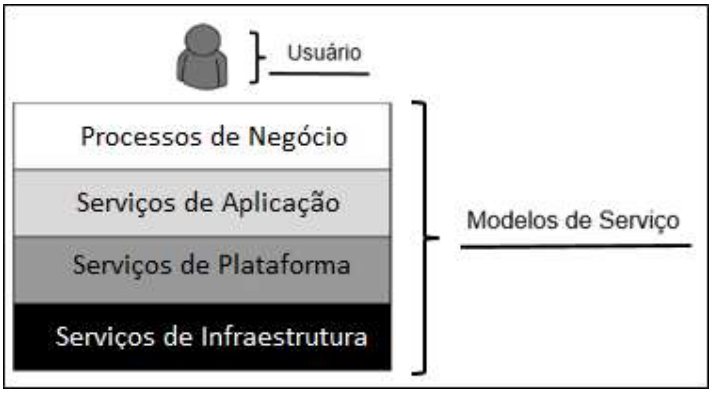
2.4.1.4 Nuvens comunitárias

As nuvens comunitárias seguem a mesma linha dos demais tipos de nuvem, porém a principal função de uma nuvem comunitária é de compartilhar os recursos em informações com outros usuários que possuem as mesmas preocupações, como segurança e política (MARINESCU, 2013).

2.4.2 **Modelos de serviço**

Na construção de serviços para computação em nuvem, é possível formar ecossistemas que determinaram o modelo de serviços da nuvem. Nas camadas de aplicação, os serviços de computação em nuvem formam o modelo de *Software* como Serviço - SaaS (*Software as a Service*); já a camada de plataforma, disponibiliza o modelo de serviço Plataforma como Serviço (PaaS); e o modelo de Infraestrutura como serviço (IaaS), onde toda a infraestrutura necessária para implantação de servidores físicos é disponibilizada pelos serviços de computação em nuvem (SOSINSKY, 2011).

A Figura 19 mostra a forma de como está organizado um ecossistema de computação em nuvem.



Adaptado de Sosinsky, 2011.

Figura 19: Ecossistemas de computação em nuvem.

2.4.2.1 Modelos IaaS

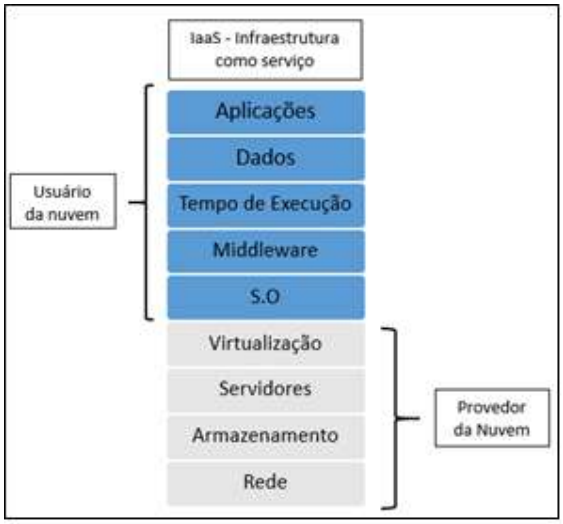
O modelo de computação do tipo IaaS – *Infrastructure as a Service* (Infraestrutura como serviço) – disponibiliza para os usuários toda a infraestrutura necessária para um servidor. No caso de provedores de computação em nuvem, é disponibilizado uma estrutura de computação que comporte vários servidores virtuais atendendo diversos clientes.

Neste modelo, o usuário dispensa a aquisição e implantação de uma infraestrutura computacional. O fornecedor fica encarregado em disponibilizar os serviços e administrar os equipamentos como: servidores, armazenamento e infraestrutura de rede. O usuário que adquire este tipo de serviço na maioria das vezes não tem o conhecimento da localidade dos equipamentos, recebendo apenas o processamento necessário para o seu uso (SOSINSKY, 2011).

O usuário ao utilizar os serviços de computação em nuvem do modelo IaaS tem a possiblidade de criar seus servidores de forma virtual usando a estrutura do provedor. Desta forma, aloca-se as aplicações nestes servidores virtuais criados pelos usuários, onde os recursos virtuais do servidor serão repassados para servidores físicos.

Na Figura 20 é demonstrado em forma de pilha cada camada de uma aplicação de computação em nuvem. Como é possível visualizar na figura, os 4 (quatro) primeiros blocos em ordem crescente são as camadas em que o provedor

que oferece o serviço de nuvem é responsável em manter ao usuário. Os 5 (cinco) blocos (Aplicações, dados, Tempo de execução, *Middleware*, S.O), demarcados com a cor azul são as camadas que ficam disponíveis para o usuário, ficando por conta dele toda a alteração e manutenção.



FONTE: SAVILL, 2012. Adaptador por Maron, Griebler. 2014

Figura 20: Demonstração das áreas do provedor e do usuário do modelo IaaS de computação em nuvem.

Dentre vários outros, podemos citar um exemplo onde empresa Amazon, que disponibiliza aos seus clientes um serviço chamado EC2 (*Elastic Cloud Compute*), que consiste em um serviço *web* para o provisionamento, gerenciamento de servidores virtuais em sua estrutura de *datacenter*. Deste modo é possível o usuário acessar estes serviços e instanciar máquinas virtuais em sua estrutura através de uma imagem pré-definida pela empresa, permitindo a execução de um sistema operacional específico em uma máquina virtual (REESE, 2009)

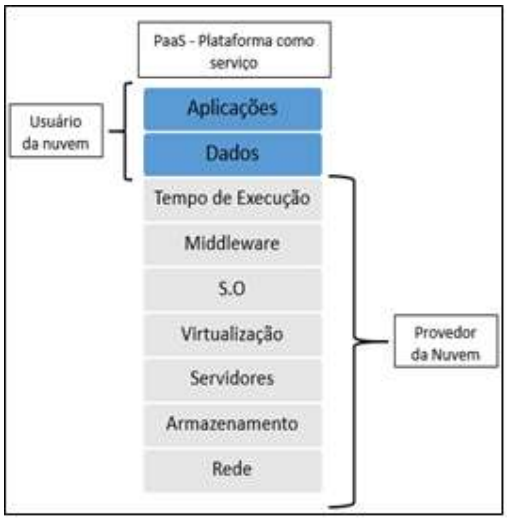
2.4.2.2 Modelo PaaS

No modelo *Plataform as a Service (PaaS)* (plataforma como serviço, na tradução literal) é a forma de oferecer uma plataforma específica para um determinado ambiente de testes ou produção, estando totalmente alocada na nuvem.

Neste modelo, é criado um ambiente personalizado dentro dos requisitos necessários para uma determinada aplicação. Sendo assim, a modalidade PaaS

oferece ferramentas e ambiente de desenvolvimento de aplicativos com plataforma de linguagens de desenvolvimento e *frameworks*. O cliente neste caso, fica responsável apenas em interagir com os *Softwares* que estão alocados nesta nuvem, ficando por conta do fornecedor todo o tipo de manutenção no ambiente (SOSINSKY, 2011).

A Figura 21 mostra em forma de pilha cada camada de uma aplicação de computação em nuvem. Os 7 (sete) primeiros blocos na ordem crescente são as camadas em que o provedor que oferece o serviço de nuvem é responsável em manter ao usuário. Os 2 (dois) blocos acima destes 7 (sete) (Aplicações e dados), demarcados com a cor azul, são as camadas que ficam disponível para o usuário, ficando por conta dele toda a alteração e manutenção.



FONTE: SAVILL, 2012. Adaptador por Maron, Griebler. 2014

Figura 21: Demonstração das áreas do provedor e do usuário do modelo PaaS de computação em nuvem.

Como exemplo para uma nuvem computacional PaaS é o Windows Azure da empresa Microsoft. Ele permite criar e executar aplicativos em uma plataforma específica usando quaisquer linguagens ou estrutura ou ferramenta para criar as aplicações, tudo hospedado em nuvem (MICROSOFT , 2013)

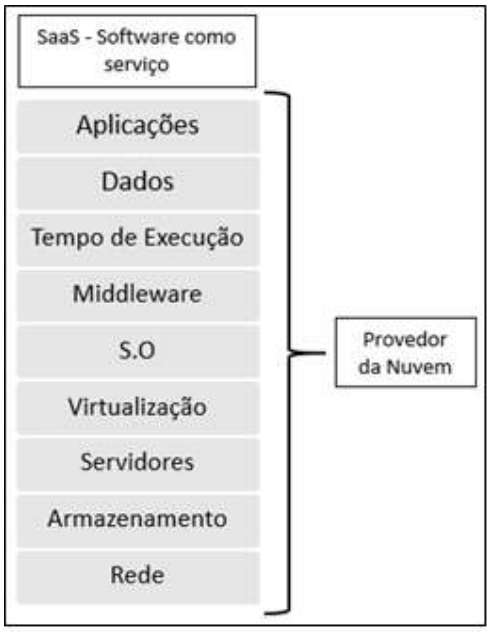
2.4.2.3 Modelo SaaS

Nesta categoria, a infraestrutura da nuvem sendo *Software* como um serviço. O fornecedor da nuvem é responsável em prover a infraestrutura necessária para

comportar o ambiente e, o *Software* que estará rodando neste ambiente também será do mesmo fornecedor, onde este modelo emprega um pouco do modelo IaaS.

Os fornecedores de serviços de computação em nuvem do modelo SaaS tem uma maior preocupação em fornecer um determinado *Software* como um serviço. Sendo assim, existe um único código que abrange toda a aplicação para todos os usuários sendo acessível de qualquer lugar e com uso de navegadores. Nesta aplicação existem módulos de personalização, aonde cada cliente consegue modificar o *software* de acordo com a sua necessidade, (LANDIS, et al., 2013).

A Figura 22 mostra em forma de pilha cada camada de uma aplicação de computação em nuvem, todas as camadas são de responsabilidade do provedor dos serviços da nuvem.



FONTE: SAVILL, 2012. Adaptador por Maron, Griebler. 2014

Figura 22: Demonstração das áreas do provedor e do usuário do modelo PaaS de computação em nuvem.

2.4.3 **Estudo de caso de computação em nuvem**

Diferentemente de uma pesquisa sobre trabalhos relacionados, sobre pesquisas aplicadas em uma área específica, o estudo de caso traz em questão

situações sobre um olhar do cliente/usuário, ou experiências vivenciadas perante a utilização de determinados serviços. Mas não deve ser visto como *marketing*, mas como uma nova perspectiva para novas ideias, e direciona-las para que de fato em algum momento possam ser implementadas, ou puramente para trazer um melhor entendimento um assunto.

Nas seções seguintes serão relatados alguns estudos de casos na utilização da computação em nuvem em um contexto geral de modelos de serviços, mas em algumas áreas específicas de atuação

2.4.3.1 Computação em nuvem em uma visão comercial

A empresa Google tem uma expressiva estrutura de computação para comportar todos os seus serviços mais populares, e em conjunto a isto, consegue disponibilizar fatias de sua estrutura para empresas alocarem softwares e servidores.

No que diz respeito a plataforma como serviço, a Google apresenta alguns estudos de caso de empresas que usam as suas estruturas para alocarem seus serviços usando uma plataforma específica. Um exemplo disto é a empresa Rovio, criadora do famoso jogo “Angry Birds”, utilizou a plataforma Google App Engine para adaptar e hospedar seus aplicativos para que suportassem a execução em navegadores Web e conter a demanda de utilização dos aplicativos (GOOGLE, 2013).

De acordo com as funcionalidades do Google App Engine, ele oferece plataforma capaz de hospedar códigos de programas e ainda possui a capacidade de executar esses códigos entregando o resultado através da internet. Essa plataforma da Google tem suporte a ferramentas de grande utilização no ramo da programação como as linguagens Python, Java, PHP, Go e MySQL (GOOGLE, 2012)

Nos modelos de serviços envolvendo o modelo SaaS, a empresa Microsoft possui vários aplicativos que antes só era possível a execução em arquiteturas físicas. Um exemplo disto é a linha de produtos *Office* (*Word, Outlook, PowerPoint, Excel, OneNote*), pois com uma conta de identificação é possível acessar estas ferramentas e através do seu navegar *web* criar/editar documentos destinados a estas aplicações.

Além da empresa Google oferecer uma plataforma de desenvolvimento e execução de códigos de programação na nuvem, ela ainda possibilita a provisionamento de máquinas virtuais na sua estrutura de datacenter, com o serviço *Compute Engine*. Essas máquinas virtuais chamadas de instâncias são alocadas utilizando um sistema operacional Linux fornecido pelo Google, com a possibilidade de escolher uma determinada configuração destas instâncias.

Na utilização da infraestrutura para o processamento dos dados, a importância de se alocar dados para o alto processamento nas nuvens, é devido ao serviço oferecido pelos provedores apresentar mais eficiência por motivo da capacidade computacional. Estes serviços comumente são cobrados pelas horas de processamento, onde cada hora possui um valor conforme a capacidade computacional das maquinas instanciadas.

O Google também oferece seus serviços de infraestrutura para fins de processamento, um estudo de caso é a utilização de uma infraestrutura de processamento por um Instituto sem fins lucrativos para sistemas de biologia, que atua fortemente em pesquisas de mapeamento de mudanças genéticas de 20 tipos de câncer, buscando acompanhar dados complexos destas doenças, é necessária uma busca rápida de novas formas de tratamentos das doenças estudadas. O principal objetivo do instituto de pesquisas é a possibilidade de alocar uma maior quantidade de dados para serem processados com um tempo menor, diminuindo processamentos que poderiam levar semanas (GOOGLE, 2012).

2.4.3.2 Computação em Nuvem em uma visão educacional

Sabendo do grande avanço das tecnologias de informações e telecomunicações, todas as facilidades encontradas na área, de alguma maneira podem ser direcionadas para a educação, e a computação em nuvem pode tornar importante essa transformação da educação.

Um trabalho realizado pela Intel Word Ahead, mostra alguns dos importantes avanços na educação após o uso da computação em nuvem em um estudo de caso. De acordo com a Intel (2010), um dos exemplos da utilização de computação em

nuvem voltado para a educação é o serviço “e-Arquivos de Alunos”. A ideia deste serviço, é que sejam centralizadas todas as informações e trabalhos de alunos de diversas escolas, onde podem ficar disponíveis para órgãos de educação poderem avaliar fatores sobre o ensino dos alunos, e até mesmo professores em suas tarefas de avaliação.

Entende-se que ao utilizar o serviço “e-Arquivos de Alunos”, professores poderão centralizar o livro de notas, de presença, plano de aulas em um único lugar. Com isso ainda, os alunos poderão utilizar estes serviços para realizarem as tarefas solicitadas pelos professores, tornando ágil o envio de exercícios.

O serviço “e-Arquivos Alunos” envolve toda uma infraestrutura de modelos de serviços voltados a computação em nuvem. Pois de acordo com suas funcionalidades, ele necessita de grande armazenamento de arquivos e informações, já que irá armazenar informações de diversas escolas. Além do armazenamento, é necessário processamento, comunicação em larga escala. Por ser um *software*, é necessário um modelo de serviço voltado a atender a demanda do programa. Sendo assim, envolve todos os modelos de serviços de infraestrutura, plataforma e serviço.

Continuando nesta área educacional, percebe-se um grande avanço na aceitação de tecnologias de computação em nuvem aqui no Brasil, onde uma instituição pública de ensino migra seus serviços para a nuvem, colocando-a em um ponto privilegiado tanto no setor educacional quanto no mercado brasileiro.

Este caso aconteceu na Universidade de São Paulo - USP, onde grande parte dos sistemas da Universidade foram destinados à nuvem, e ainda permitiu que a infraestrutura cedesse uma fatia para serviços do tipo IaaS (GALISTEU, 2013).

A busca pela adesão de uma infraestrutura de computação em nuvem, aconteceu devido a alguns problemas que a Universidade vinha enfrentando na falta de agilidade e flexibilidade, gastos desnecessários com infraestrutura, ausência e falhas no alinhamento entre as atualizações necessárias em *softwares* do parque de máquinas, e a inexistência de escalabilidade. Mas segundo Galisteu (2013), talvez um dos principais problemas e necessidades da USP, seria na implementação de projetos

científicos que a academia desenvolve. Isto devido a falta de orçamento necessário e equipamentos que comportassem toda a necessidade computacional para utilização em determinados projetos.

O projeto que levou 7 meses para entrar em atividade, e com um custo estimado em 200 milhões, serviram para montar toda a infraestrutura de computação em um *datacenter* e, ainda para atualização de *desktops* e infraestrutura de redes. A universidade teve apoio de grandes empresas para a implementação de toda esta estrutura (GALISTEU, 2013).

Com a implementação da nuvem, pode-se centralizar grandes serviços da Universidade, disponibilizando recursos para 3 campus da USP, e mais para alguns CPDs, e ainda englobando serviços internos da própria universidade. Toda a nova infraestrutura permite ainda para a USP a criação de políticas de *home office*, e ainda operando como modelos de serviços IaaS, onde potencializará o desenvolvimento de pesquisas na instituição (GALISTEU, 2013).

Visto que todo esse trabalho acabou sendo um grande avanço tecnológico para a USP, onde a questão muitas vezes não está somente em como ele foi aplicado, mas sim, nas oportunidades que todo esse empreendimento trará para a Universidade e os acadêmicos.

2.5 FERRAMENTAS DE ADMINISTRAÇÃO DE NUVEM

Hoje a computação em nuvem consegue oferecer recursos de maneira prática e eficiente para os usuários. Com os modelos de serviços SaaS – *Software* como um serviço. PaaS – Plataforma como um serviço. IaaS – Infraestrutura como um serviço.

Mas para isso, são necessários que cada uma das ferramentas cumpra aos requisitos para a utilização de qualquer um destes modelos. Estes requisitos são para tornar a utilização da computação em nuvem, em uma ferramenta funcional, mantendo seu princípio básico que é a entrega de serviços através da internet.

Cada ferramenta usada para a implantação de serviços de computação em nuvem possui particularidades que são definidas antes mesmo da sua criação. Como

por exemplo: definição de linguagens para a programação da ferramenta. Compatibilidade com virtualizadores. Tipos de serviços de redes. O conjunto de ferramentas, e serviços que permitem a funcionalidade de uma nuvem. Isso se torna importante, em um exemplo básico, em ferramentas que são do modelo de serviços IaaS, a escolha do virtualizador, fará com o usuário opte por determinada ferramenta de administração de uma nuvem, para instalação em um ambiente privado.

Nesta seção serão particularizadas algumas ferramentas para administração uso de nuvens do tipo IaaS.

2.5.1 **OpenStack**

O *software* OpenStack teve seu desenvolvimento através de licenças de código aberto. Nascido de um projeto da Rackspace e da NASA, hoje o OpenStack é usado para construção de nuvens IaaS públicas e privadas (JACKSON, 2012).

O OpenStack permite o gerenciamento de recursos computacionais sob demanda, através do provisionamento e gerenciamento de máquinas virtuais e redes de grande porte. Com ele, o gerenciamento é feito pelo administradores e desenvolvedores através das tecnologias como APIs, interfaces WEB. Alguns dos componentes que compõe sua estrutura, são implementadas pelos próprios desenvolvedores, mas alguns, são utilizados de terceiros, e juntos incrementam a funcionalidade da ferramenta (OPENSTACK [A], 2014).

Em nuvens do tipo IaaS, o uso de armazenamento é inevitável. O método de para o uso deste recurso pode ser tanto para emular um disco rígido em uma máquina virtual, ou até mesmo para centralizar objetos que são necessários para a nuvem.

As unidades de armazenamento destinadas para uso nas infraestruturas com o OpenStack são feitas com o Cinder e o Swift. O componente Cinder implementa o armazenamento através de blocos de armazenamento, já o Swift implementa o armazenamento através de objetos de armazenamento.

O uso do método em que o fornecimento de armazenamento é feito por objetos, permite ao usuário um baixo custo financeiro. Pois o sistema permite o uso

de APIs de comunicação que implementa um armazenamento em um sistema de arquivos não tradicional de forma distribuída em unidades de toda a infraestrutura, sendo altamente escalável e capaz de gerenciar grandes quantidades de dados. Este método pode ser implementado através de serviços como Gluster UFO, onde uma versão modificada do Swift implementa este armazenamento distribuído, permitindo recursos adicionais com a utilização de máquinas virtuais, uma delas sendo a migração ao vivo (PEPPLE, 2011).

Já o método de armazenamento em bloco gerenciado pelo Cinder, fornece um armazenamento persistente. Desta forma, são criados discos de armazenamento estáticos, e assim anexados e desanexados de instâncias de máquinas virtuais na estrutura OpenStack. Este tipo de método, é comumente utilizado em ambientes sensíveis ao desempenho, como no uso de gerenciamento de banco de dados. LVM (*Logical Volume Manager*) é o um dos sistemas que fornecem em conjunto com o Cinder, o armazenamento através de blocos, com ele são criados discos lógicos sobre uma camada de abstração no o disco (PEPPLE, 2011).

A rede também se torna um recurso inevitável para o uso de computação em nuvem. Na ferramenta OpenStack, isso é implementando usando alguns componentes como o *nova-network e o neutron*.

Neutron implementa redes mais complexas. Através dele, é possível implementar usando recursos com VLANs, e ainda disponibilizar diferentes redes para diferentes usuários. Com o *neutron*, existem as abstrações de componentes de rede para as máquinas virtuais, tais componentes como: roteadores, redes, sub-redes, rotas. É importante ressaltar, que existe a possibilidade de integrar redes usadas pelas máquinas virtuais com redes externas e obter acesso a internet, para que isso ocorra, é necessário a configuração do *neutron* para haver a simulação de tais componentes e assim obter a funcionalidade desejada (OPENSTACK [A], 2014).

Com o *neutron*, ainda é possível implementar grupos de segurança. Esses grupos tem o sentido de definir regras de segurança no tráfego de rede entre as máquinas virtuais e redes externas (OPENSTACK [A], 2014) .

Um componente que está intimamente ligado com o *neutron* e um dos mais importantes nas tarefas de redes, é *Open vSwitch.* Através dele que são feitos os recursos de *bridges* e portas de conexões que são representadas ao sistema operacional como interfaces físicas. (OPENSTACK [B], 2014).

O uso de *bridges* representa interfaces virtuais gerenciadas pelo *neutron* e *open vSwitch* e junto com elas são definidos os tipos de redes virtuais como a GRE ou VLANs. GRE (*Generic Routing Encapsulation)* é um tipo de rede que comumente é usado para conexões VPN, onde tem como o objetivo de fazer um encapsulamento diferente dos pacotes de rede, alterando informações sobre seu roteamento, e assim encaminha-los ao seu destino. Já o uso de VLAN, altera o cabeçalho do pacote *ethernet,* acrescentando as informações da tag de VLAN, assim *switches* e o próprio *open vSwitch* fará o compartilhamento destes pacotes somente com as interfaces e redes que pertencem a tag VLAN especificada (OPENSTACK [B], 2014).

O *nova-network* é um componente que apresenta funções semelhantes ao *neutron*, mas gradativamente será descontinuado e substituído pelo *neutron* (OPENSTACK [C], 2014).

Outro componente importante na estrutura da ferramenta OpenStack é o *Glance*. Através dele é possível fornecer serviços para o gerenciamento de imagens que serão usadas pelas máquinas virtuais. Além disso, é possível controlar uma série de objetos como *snapshots* de máquinas virtuais. Com todo esse controle de dados, ele necessita de uma fatia de armazenamento de dados trabalhando em conjunto com o *Swift*. (OPENSTACK [D], 2014).

Trove é serviço que foi projetado inteiramente para ser executado na ferramenta OpenStack, o objetivo é oferecer recursos e serviços de banco de dados relacional, aonde é possível gerenciar instâncias de banco de dadas. A proposta da OpenStack em inserir esse serviço a partir da Juno, é proporcionar um ambiente isolado com alto desempenho no gerenciamento de instâncias de banco de dados, incluindo a automatização de tarefas administrativas complexas, como configurações, implantações, *backups,* restaurações e monitoramento, serviços relacionados a administração de banco de dados (OPENSTACK [E], 2014).

Preocupado com a segurança, a ferramenta OpenStack possui um serviço que gerencia os usuários destinados aos serviços da infraestrutura OpenStack, definido regras referente ao acesso de certos serviços (OPENSTACK [F], 2014)

2.5.2 **OpenNebula**

A ferramenta OpenNebula surgiu de um projeto iniciado por Ignacio M. Llorente e Rubén S. Montero, em 2005. Desde a primeira versão publicada em 2008, manteve seus códigos na linha de *softwares Open Sources*. E hoje como resultado, surge uma ferramenta para administração de ambientes de computação em nuvem, sendo elas privadas, públicas e híbridas (SOTO, 2011).

De acordo com Soto (2011), o modelo básico da arquitetura da ferramenta OpenNebula, se baseia nos modelos clássicos de *cluster*, como o Cluster Beowulf. Neste modelo, é necessário a utilização de um nodo da infraestrutura para ser o nodo mestre, ou *front-end*, que é o responsável em executar os principais serviços que mantem a ferramenta em atividade e administrar os recursos disponíveis no restante da infraestrutura.

Este *front-end*, executa o serviço *oned* (OpenNebula Deamon), que consiste na administração dos principais serviços da infraestrutura da nuvem, como a rede, armazenamento, virtualizador, e na supervisão das máquinas virtuais do ambiente (SOTO, 2011).

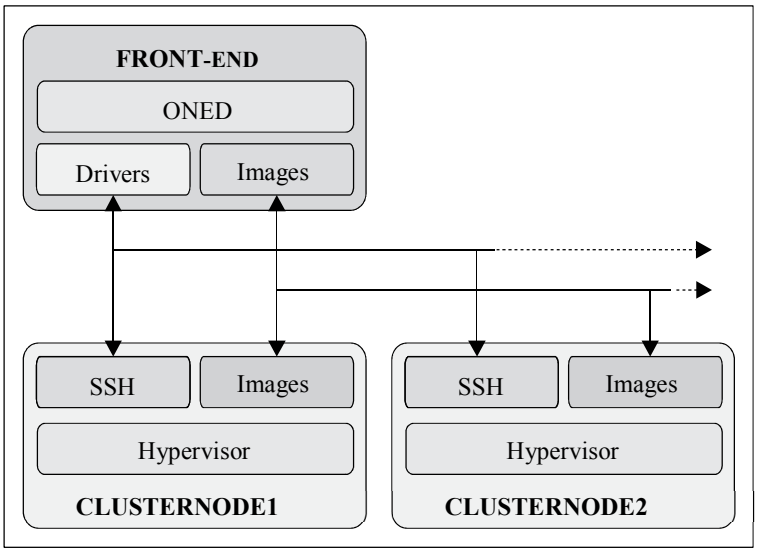
Além do *core* principal da ferramenta OpenNebula, existem outros componentes que tornam a ferramenta funcional. É o exemplo OneGate, um servidor que especificamente atende requisições HTTP das máquinas virtuais (OPENNEBULA, 2014).

Outro componente, é OneFlow, um recurso que permite ao usuário executar aplicações multicamadas que exige uma cooperação entre as outras máquinas virtuais (OPENNEBULA, 2014).

A comunicação entre os nodos da infraestrutura OpenNebula, se faz com o protocolo SSH. Com este modo a ferramenta disponibiliza armazenamento, e prove

as imagens que deverão ser instanciadas na estrutura. Ainda existe a possibilidade de implementar recurso NFS, um sistema de arquivos compartilhado (OPENNEBULA, 2014).

A comunicação via protocolo SSH é importante para a infraestrutura da ferramenta OpenNebula. De acordo com Toraldo (2012) através do SSH, o *core* principal aplica verificações do estado atual de cada máquina virtual que roda na infraestrutura.



Fonte: TORALDO, 2013

Figura 23: Diagrama do sistema OpenNebula.

A Figura 23, faz uma representação básica de como a ferramenta opera, demonstrando a necessidade da comunicação via SSH entre os nodos da infraestrutura, e no *front-end* demonstrando o serviço principal *oned*, e em cada nodo a importância do *hypervisor*.

2.5.3 **Análise sucinta sobre OpenStack e OpenNebula**

As ferramentas de administração citadas anteriormente (OpenStack OpenNebula), tem um expressivo período de mercado, e ambas surgiram para um propósito semelhante. O desenvolvimento destas ferramentas envolve grandes equipes, que analisam cada decisão sobre o projeto das ferramentas.

Toda a estrutura que envolve e garante todas as suas funcionalidades é grande, comparado com o que foi apresentado sobre as duas ferramentas. Porém, é notável que a ferramenta OpenStack tem uma quantidade maior de componentes específicos da própria ferramenta que gerenciam determinados recursos.

A ferramenta OpenNebula de igual forma apresenta uma estrutura de componentes essenciais e importantes para o seu funcionamento, mas com o estudo da bibliografia, estes componentes são em menores quantidades, mas que muitas vezes podem ter uma eficiência maior de alguns componentes específicos no momento do gerenciamento dos recursos, dispensando assim a quantidade e focando na qualidade.

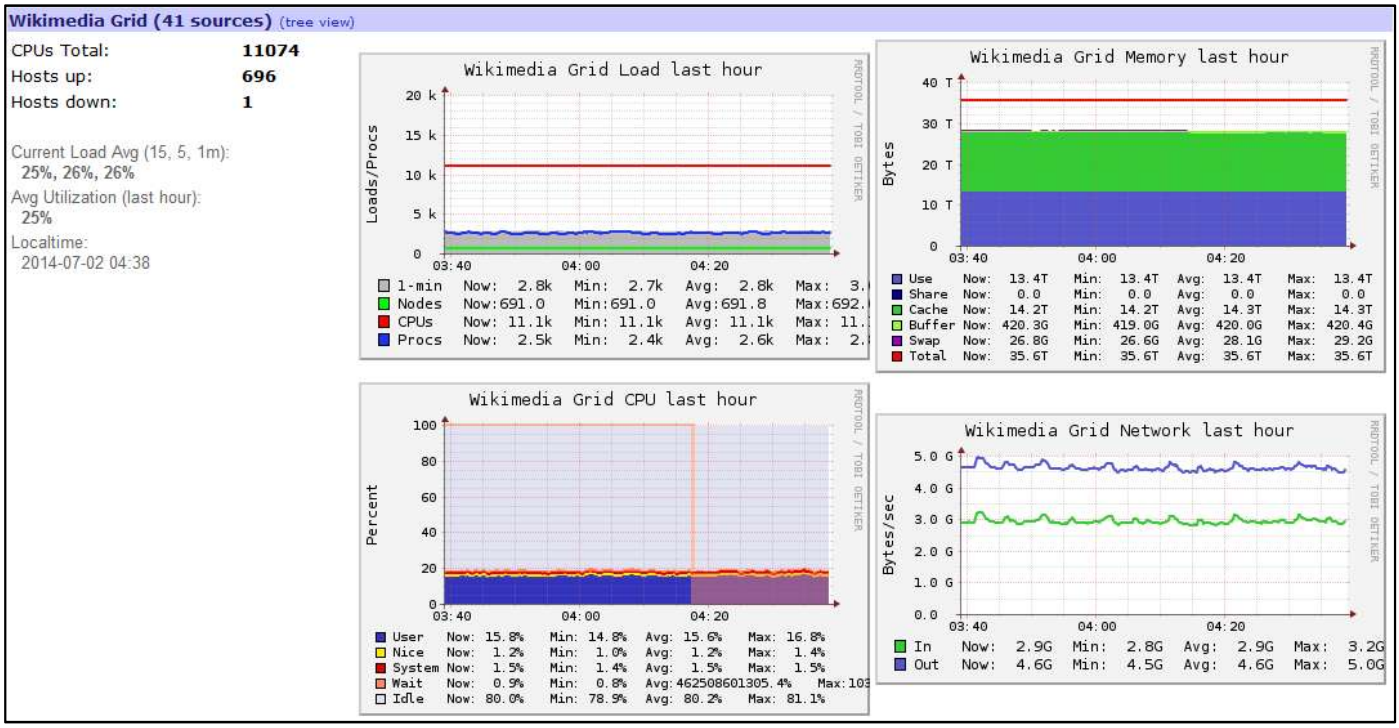
É Importante ressaltar que a quantidade de componentes, ou a centralização do gerenciamento de recursos em uma ferramenta pode estar profundamente ligada à utilização de recursos computacionais, o que pode ser um problema em infraestruturas de nuvens do tipo IaaS, que tem seu principal objetivo a alocação da infraestrutura para prover algum tipo de processamento.

2.5.4 **Ganglia**

Ganglia é um projeto de código aberto vigente sobre os termos da *BSD-licensed*, que foi desenvolvido na Universidade Berkeley da Califórnia, Estados Unidos. É um sistema de monitoramento altamente escalável e distribuído, desenvolvido principal para computação de alto desempenho (GANGLIA, 2014).

Ganglia tem o objetivo de monitorar valores de processamentos de nodos nas infraestruturas de *grid* e clusters, pois foi desenvolvido para gerar o mínimo de impacto durante o tráfego das informações. Todos seus componentes foram cuidadosamente projetados, e acima de tudo, aproveitando tecnologias como uso de XML para representação dos dados, XDR sendo o protocolo de transporte de dados compactos na rede, e ainda o RRDtool para armazenamento e ainda permitindo a visualização dos dados em formas de gráficos (GANGLIA, 2014).

É possível perceber a capacidade real de monitoramento do Ganglia através da Figura 24, nela consta um exemplo real do monitoramento de unidades computacionais dos *datacenters* da fundação WIKIMEDIA. Na figura, consta apenas a visualização principal dos 41 clusters e *grids* monitoradas pelo Ganglia, totalizando em 697 *hosts*, e 11075 CPUs.



Fonte: Wikimedia, 2014.

Figura 24: Monitoramento Grid Wikimedia usando Ganglia.

2.6 MEDIDAS DE DESEMPENHO EM COMPUTAÇÃO

Em meio as plataformas de serviços de computação em nuvem, é necessário que haja nestes ambientes um nível aceitável de desempenho para as atividades que serão processadas. Em nuvens de modelo IaaS, uma de suas principais características é a robustez diante ao processamento, por essas grandes características que usuário substituem arquiteturas locais de processamento, e alocam serviços na nuvem.

A avaliação de desempenho de ambientes de computação em nuvem é muito importante, pois os resultados destas análises poderão determinar a estrutura necessária na nuvem para que o usuário receba um equivalente em processamento à uma estrutura local. Buscar projetar melhores sistemas e projetar plataformas de nuvem que substituam uma arquitetura local são os grandes desafios desta abordagem de desempenho com as estruturas de computação em nuvem, que se define na capacidade de mensurar de forma confiante o desempenho, a elasticidade, a estabilidade e a resiliência (IOSUP, et al., 2013).

Zia (2012), em sua pesquisa, busca características que são relevantes para o desempenho em computação em nuvem. Através da análise de trabalhos relacionados (Seção 3.1) pode perceber que os serviços de armazenamento, serviços de rede, programação, especialmente em nuvens do tipo IaaS, são os que envolve as cargas de trabalhos em processadores virtuais e utilização de memória. O processo de avaliação de desempenho em ambiente de computação em nuvem vem aos poucos tomando forma, sendo que os elementos que caracterizam um *benchmark* e atributos de avaliação para este tipo de infraestrutura se mantinham inalterados desde os anos 90.

Iosup (2013) busca trazer em seu trabalho uma abordagem unificada sobre a avaliação de desempenho nestes ambientes, aonde seu principal objetivo de contribuição é para definir as características de *benchmarks* e seus principais desafios para aquisição de um sistema de gestão de desempenho, neste sentido, Iosup (2013) buscou introduzir uma abordagem genérica para aferição de uma estrutura IaaS de computação em nuvem. Buscou definir as abordagens para avaliação comparativa de nuvens, se concentrando principalmente na metodologia de avaliação, no sistema, nas cargas de trabalho expostas à infraestrutura e buscar o relacionamento com métricas de avaliação.

Nas seções seguintes serão abordados alguns cálculos para valores de desempenho em ambientes computacionais, no qual podem ser usados para infraestruturas de computação em nuvem.

2.6.1 **Speed-up**

De acordo com Gonçalves (2007), O cálculo de *speed-up* é a equação mais apropriada para alcançar as métricas de execuções de sistemas multiprocessados, incluindo as formas de sistemas distribuídos, como *grids* e *clusters*.

A fórmula do *speed-up* se resume no cálculo entre tempo de execução sequencial e o tempo de execução em paralelo (ROCHA, 2007).

A fórmula usada para cálculo do *speed-up* é listado a baixo.

$$S(p) = \frac{T(1)}{T(p)}$$

A variável $\mathrm{T}(1)$ deve ser substituída pelos valores de tempo de execução em um único processador, no caso processamento sequencial. Já a variável $\mathrm{T}(\mathrm{p})$ é substituída pelos valores de tempo de execuções em paralelo onde a variável p deve ser substituída pela quantidade de processadores existentes na arquitetura. Após a substituição das variáveis, segue-se as regras tradicionais de cálculos matemáticos.

2.6.2 **Eficiência**

O grau de eficiência de um processamento é definido pelo aproveitamento dos recursos computacionais disponíveis no momento do processamento. A eficiência é alcançada através do cálculo entre o grau de desempenho e os recursos computacionais disponíveis (ROCHA, 2007).

A fórmula usada para o cálculo da eficiência é listada a seguir.

$$E(p) = \frac{S(p)}{p} = \frac{T(1)}{p \times T(p)}$$

Na equação da eficiência, a variável $\mathrm{S}(\mathrm{p})$ deve ser alterada pelo valor total calculado no *speed-up*. A variável $\mathrm{T}(1)$ é alterada pelo valor de tempo do processamento sequencial. A variável $\mathrm{T}(\mathrm{p})$ deve ser trocada pelo tempo de execução

do processamento em paralelo. E todas as variáveis p devem ser trocadas pelo valor da quantia de processadores existentes na arquitetura de processamento. Após a substituição das variáveis, segue-se as regras tradicionais de cálculos matemáticos.

2.6.3 **Redundância**

Os valores da redundância são encontrados a partir da medida entre o número de operações realizadas pela execução paralela e pela execução sequencial do processamento. Através deste cálculo é possível ter conhecimento sobre o grau de aumento do processamento (ROCHA, 2007).

A fórmula para o cálculo de redundância em computação paralela é mostrada a seguir.

$$R(p) = \frac{O(p)}{O(1)}$$

Na equação, a variável $\mathrm{O(p)}$ é o valor total das operações realizadas com a quantidade de processadores existentes na arquitetura. E a variável $\mathrm{O(1)}$ é trocada pelos valores de processamento exercidos em um único processador. Após a substituição das variáveis, segue-se as regras tradicionais de cálculos matemáticos.

2.6.4 **Utilização**

Os valores de utilização são adquiridos pelo cálculo entre a capacidade computacional utilizada e a capacidade disponível. A utilização mede o grau de aproveitamento da capacidade de processamento (ROCHA, 2007).

A fórmula usada para alcançar os dados de utilização é mostrada a seguir.

$$U(p) = R(p) \times E(p)$$

Para encontrar o valor de utilização é necessário pegar os resultados dos cálculos de redundância e de eficiência. Na equação, a variável $\mathrm{R(p)}$ é trocada pelo

valor obtido do cálculo da redundância, e a variável $\mathrm{E(p)}$ é substituída pelo valor obtido no cálculo de eficiência.

2.6.5 **Capacidade de desempenho em disco**

O fator de desempenho em disco é algo que deve levar-se em conta em qualquer tipo de sistema computacional. Para isso é importante compreender o fator desempenho tanto pelo lado do *hardware* como também pelo lado do *software.* Primeiramente, leva-se em conta as características físicas do dispositivo, e o segundo, qual é a exigência de IOPS (*Input/Output Operations Per Second*) do *software* pelo disco.

O disco é tido como principal gargalo em uma arquitetura computacional, pelo motivo de ser uma peça mecânica, pois em determinados casos, o desempenho fica por conta da rotação dos componentes internos do disco. De acordo com Lowe (2010), o importante em definir o desempenho de um disco é saber seus valores de desempenho aleatório, desempenho sequencial e a combinação dos dois. Além de existirem ferramentas que avaliam de forma prática o desempenho de um HD, é possível faze-lo também através de uma equação matemática que será ilustrada a seguir:

$$IOPS = \frac{1}{(Tempo\ latência + Tempo\ médio\ de\ Busca)}$$

Para encontrar o valor de IOPS, é necessário substituir as variáveis $\mathrm{Tempo\ de\ latência}$ e $\mathrm{Tempo\ Médio\ de\ Busca}$, valores no qual se consegue nos detalhes técnicos do dispositivo.

2.6.6 **Capacidade de desempenho em processador**

O processador tem seu papel fundamental na execução das tarefas e demais instruções direcionadas à ele, mas além disso é importante ele apresentar um nível de desempenho na execução destas tarefas.

Para saber o seu poder computacional real, somente aplicando ferramentas que medem o seu desempenho. No entanto, ao dimensionar a estrutura de processamento, ainda não é possível aplicar estes tipos de ferramentas, então é necessário saber a capacidade teórica de um processador e assim poder estimar o produto correto para a estrutura de processamento.

Para estimar a capacidade total teórica de um processador usa-se a seguinte equação:

$$GFlops = (Vel.CPU\ em\ GHz) \times (\ Num.cores\ por\ CPU) \times (Num.instruções\ ciclo\ clock)$$

As variáveis velocidade da CPU em GHz, número de *cores* por CPU e número de instruções por ciclo de *clock,* são encontradas nos manuais técnicos dos dispositivos (NOVATTE, 2013). Após as substituições, aplicar as regras padrões de cálculos matemáticos.

2.6.7 **Desempenho em memória RAM**

A memória RAM é uma característica fundamental na infraestrutura física de um ambiente de computação em nuvem e em qualquer outra estrutura de processamento. Seu princípio básico é trabalhar densamente com o processador, armazenando informações que continuamente são buscadas por ele.

A memória tem seu processo técnico de gravação e leitura muito rápido, então até o momento da escrita desta seção não foi encontrado uma equação matemática para definir um valor teórico da capacidade de uma memória RAM (Alecrim 2006).

Para poder perceber a capacidade de uma memória RAM de forma teórica, existem várias características onde é possível identificar a destreza no processamento, conforme Alecrim (2006), temporização e latência são características que quanto menores em uma memória, mais rápidas elas são.

Para entender esses valores de latência e temporização, Alecrim (2006) aponta essas características. Como em um exemplo, é possível encontrar em uma

memória valores de latência deste modo: 5-4-4-15-1T. Estes valores são descritos desta forma: tCL-tRCD-tRP-tRAS-CR, sendo particularizado a seguir cada um dos parâmetros:

- tCL: é a quantidade de ciclos de *clock* necessária para a memória RAM ler os endereços de dados solicitados pelo processador (No exemplo o valor é 5).
- tRCD: é a quantidade de ciclos de *clock* para entrega de um endereço ao processador (No exemplo o valor é 4).
- tRP: intervalo medido em *clocks,* é o tempo exercido para desativar uma linha de memória e ativar outra. (No exemplo o valor é 4)
- tRAS: é medido em *clock*, é definido como o tempo de um comando de ativar uma linha de memória até o próximo. (No exemplo o valor é 15).
- CR: é o parâmetro que define o intervalor de ativação de um comando para definir se é leitura ou escrita na memória. (No exemplo é 1).

Além da importância das características físicas contribuir para o desempenho efetivo, é importante ainda poder medir seu desempenho através de ferramentas que executam cargas de trabalhos nestes equipamentos, e assim são aferidas. Essas cargas de trabalho são induzidas por *benchmarks*, e para uma memória RAM, um dos principais fatores medidos por um *benchmark* é seu *throughput.*

Sabendo disso, Chavence (2005), aborda alguns problemas em contraste com o processador e a memória RAM no que diz respeito ao *throughput.* Pois de acordo com o autor, os processadores estão cada vez mais exigindo desempenho nas transferências de informações da memória RAM, pelos seguintes motivos: o tamanho de blocos da memória cache do processador estão aumentando. As velocidades dos processadores e suas interfaces estão evoluindo rapidamente. Programas ocupando grandes áreas na memória RAM.

Porém, o comportamento da memória se apresentam ao oposto das necessidades do processador, pois o rendimento disponível em *chips* de memória melhora com menos intensidade que as do processador. Outra é o número de pinos

de uma memória ser limitada, dificultando o paralelismo no acesso aos dados em memória (CHEVANCE, 2005).

2.7 BENCHMARKS PARA AVALIAÇÃO DE DESEMPENHO

A utilização de *benchmarks* tem o princípio básico de avaliar o desempenho de sistemas computacionais ou arquiteturas de *hardware*. Estes *benchmarks* são *softwares* desenvolvidos e padronizados para a aplicabilidade de se obter resultados de algum determinado processamento através da aplicação de cargas de trabalhos (*workloads*). Estes tipos de procedimentos compõe uma análise de desempenho (MELO, et al., 2014)

No caso de *benchmark* como um *software*, aplica-se cargas de trabalho nos itens a serem testados e assim contabiliza-se as informações importantes para categorizar os resultados do desempenho (MELO, et al., 2014).

Existem vários tipos de *benchmarks* para diversas finalidades, e abrangendo várias áreas de negócio. Um exemplo é a existência de *benchmarks* para avaliar a atuação de funcionários nas empresas, avaliação de desempenho de um determinado setor de empresa. Seguindo nesta linha de pensamento, existem *benchmarks* para avaliação de desempenho na área da tecnologia da informação.

Existe uma grande variedade de *benchmarks* onde torna possível avaliar quase todo tipo de estrutura ou *software*. Um *benchmark* nada mais é do que um banco de referência. O seu uso é para tomar conhecimento da dimensão computacional que sua estrutura ou *software* consegue alcançar, resultando em vários tipos de informações importantes. Nas seções seguintes são mostrados alguns *benchmarks*.

2.7.1 **Linpack**

É um *software* escrito em linguagem Fortran que executa rotinas de cálculo para resolução de equações lineares. O princípio geral é a decomposição algébrica linear envolvendo matrizes simples (DONGARRA, et al., 2001).

Neste sentido, o *benchmark* solicita ao processador requisições de cálculos de ponto flutuante, onde o Linpack contabiliza os resultados dos valores que indicam a capacidade de total de cálculos por ciclo de *clock* do processador, contabilizando essa quantia em Mflop/s, Gglop/s, ou Tflops. Este *benchmark* comumente é utilizado para medir a capacidade de processamento da CPU de *datacenters* (DONGARRA, 2007).

FLOPS é a medida da capacidade de operação de ponto flutuando que um processador exerce durante um segundo. Enquanto o *clock* de um processador é medido em mega-hertz, FLOPS é quantidade de cálculos que um processador consegue resolver um ciclo de *clock* (TECHTERMS, 2009).

2.7.2 **OLTPBenchmarck**

Sendo específico para avaliação de desempenho em banco de dados, o *framework* OLTPBenchmark aplica cargas multi *threaded* variáveis a um banco de dados. Este *framework* permite configurar vários tipos de *workloads* que serão aplicados ao banco de dados, permite ainda personalizar taxas de tempo para submissões de requisições ao banco, definir os porcentuais de transações dos *workloads* e além de catalogar todas as informações referente ao processamento exercido, facilitando na classificação dos dados (OLTPBENCHMARK, 2014).

O OLTPBenchmark pode ser utilizado em banco de dados MySQL, ORACLE, SQLServer, DB2, HSQLDB, Amazon RDS MySQL, Amazon RDS Oracle, SQL Azure (OLTPBENCHMARK, 2012).

2.7.3 **NetPerf**

NetPerf é um dos vários *benchmarks* para avaliar o desempenho da rede de comunicação. Seu princípio básico é avaliar o tempo de requisição e resposta entre um servidor e um cliente através de *sockets* TCP e UDP, tendo neste caso o valor de *throughput*, que seria o total de vazão da comunicação entre o canal do cliente e servidor.

Para poder criar os *sockets* de comunicação, o próprio NetPerf inicia em um *host* um processo servidor que irá catalogar o desempenho do tráfego e, em um outro *host* é iniciado um processo cliente indicando o *host* servidor para que cliente inicie as transferências via rede, e o servidor avalie esses dados.

2.7.4 **Iperf**

Iperf é uma ferramenta para avaliação de desempenho de redes de comunicação. Originalmente desenvolvido por NLANR/DAST, ele mede o desempenho de uma rede usando os protocolos TCP e UDP (IPERF, 2014).

Usando o Iperf com o protocolo TCP, é possível medir a largura de banda de uma rede. Obter relatórios tamanho de MSS/MTU dados de leitura dessas informações. Obter um tamanho de janelas específicas para os testes, através de via *buffers* de *socket* (IPERF, 2014).

Usando o Iperf definindo suas configurações com o protocolo UDP, a ferramenta consegue criar larguras de bandas especificadas. Ainda é possível medir as perdas de pacotes durante a avaliação, atrasos de *jitter*, e utilizar um recurso de *multi-threaded*, aonde cliente e servidor da ferramenta são capazes de ter várias conexões simultâneas (IPERF, 2014).

2.7.5 **NetPIPE**

Network Protocol Independent Performance Evaluator (NetPIPE) é uma ferramenta de avaliação de desempenho de redes de comunicação. Com ele são realizados testes do tipo *ping-pong* entre cliente e servidor, aonde os tamanhos das mensagens usadas constantemente se alteram em tamanhos aleatórios, além deste recurso de troca de tamanho de pacote, o sistema ainda aplica no momento dos testes tipos de interferências (NETPIPE, 2009).

2.7.6 **STREAM**

STREAM é um *benchmark* escrito em linguagem FORTRAN, e é usado para avaliar o desempenho de memória RAM em *clusters.* O *benchmark* executa 4 tipos de

testes básicos: copia, escala, soma e tríade. E destes testes estima o desempenho da arquitetura (MCCALPIN, 1996).

2.7.7 **SPECvirt_sc2010**

Este *benchmark* é usado especificamente para plataformas de virtualização. O SPECvirt_sc2010 aplica a medição do desempenho em grande parte dos componentes do sistema, inclusive testes no *hardware,* na plataforma de virtualização e no sistema operacional convidado (SPEC, 2003).

O *benchmark* utiliza cargas de trabalho SPEC, representado cargas comuns em um ambiente virtualizado, corresponde a utilização de CPU, memória, disco I/O (SPEC, 2014).

2.7.8 **IOzone**

É uma ferramenta de *benchmark* para dispositivos de armazenamento, sendo compatível em várias plataformas de sistemas operacionais. O IOzone faz uma ampla análise do dispositivo de armazenamento e das operações de I/O do sistema, executando testes do tipo: Ler, escrever, reler, re-escrever, ler de trás para frente, leitura *strided, fread, fwrite*, leitura aleatória, *pread*, mmap, *aio_read, aio_write* (IOZONE, 2006).

2.7.9 **BONNIE++**

Inicialmente é um *benchmark* desenvolvido por Tim Bray, mas Rssell Coker implementou seu código em C++. Bonnie++ é uma ferramenta usada para testar unidades de armazenamento com capacidades maiores que 2GB, em uma arquitetura de 32 bits. Os testes baseiam-se em operações de exclusão, verificação e desligamento de arquivos em uma unidade de armazenamento (BONNIE++, 2014).

Os modos dos testes baseiam-se em 3 tipos de testes, sendo eles: saídas sequenciais por caractere, por blocos e através de reescritas de arquivos. Entradas sequenciais por blocos e por caractere. E através de testes randômicos (BONNIE++, 2014)

2.7.10 **NPB (*NAS PARALLEL BENCHMARK)***

Este conjunto de pequenos *benchmarks* lançado pela NASA em 1992 tinha como objetivo suprir a falta de *benchmarks* para computadores altamente paralelos. Os *benchmarks* são derivados da dinâmica de fluídos computacionais de aplicativos que consistem em cinco núcleos e três pseudo-aplicações, e ainda com *benchmarks* multi-zona, I/O, e de malhas adaptativas de grades computacionais paralelas. (NASA, 2012).

As intensidades das cargas de trabalho são definidas por classes no momento da utilização dos *benchmarks*. As classes serão particularizadas a seguir:

- Classe S: cargas pequenas, usado para testes rápidos.
- Classe W: tamanho para estações de trabalhos.
- Classes A, B, C: tamanhos de testes padrões. Aumenta em torno de 4 vezes entre uma classe e outra.
- Classes D, E, F: Grandes cargas no trabalho. Aumenta em torno de 16 vezes entre cada classe.

De acordo com NASA (2012) os 5 (cinco) *benchmarks* usados para os cálculos de movimentação da dinâmica de fluidos computacional são:

- IS – *Integer Sort:* Ordena números inteiros usando *bucket sort*.
- EP – *Embarrassingly Parallel:* Geração independente de valores *Gaussian* e variáveis randômicas usando o método Polar Marsaglia
- CG – *Conjugate Gradiant*: Cálculo de valores de matrizes.
- MG – *Multi-Grid*: Comunicação intensiva de curta e longa distância com a memória.
- FT – *Fast Fourier Transform:* método Transformada de Fourier, usando a comunicação todos para todos.

Complementando o conjunto de *benchmarks* NAS, de acordo com NASA (2012), as 3 pseudo-aplicações aplicam a resolução de um sistema de equações diferenciais parciais não-lineares, sendo elas:

- BT – *Block Tri-diagonal:* Algoritmo Bloqueio Tridiagonal.
- SP – *Scalar Penta-diagonal:* Algoritmo Penta-diagonal.
- LU – *Lower-Upper* – Algoritmo de Gauss.

E para avaliação de I/O paralelo e movimentos de dados e computação não estruturada, segundo NASA (2012), são usados os seguintes *benchmarks*:

- UA – *Unstructured Adaptive:* Resolução de equações em malhas adaptativas não estruturadas, acesso à memória dinamicamente e irregularmente.
- BT-IO – testes de diferentes técnicas de I/O paralelo.
- DC – *Data Cube:* Cálculos com valores comumente usados para série temporal.
- DT – *Data Traffic.*

O Quadro 03 representa a forma de como os *benchmarks* da suíte NPB 3.3, aplica nos testes durante a execução em uma infraestrutura. Na coluna "*Benchmark"* são identificados todos os *benchmarks* composta na suíte da versão NPB 3.3. Na coluna "Parâmetro", são identificados os parâmetros importantes de cada *benchmark*. E nas colunas "Classe S", "Classe W", "Classe A", "Classe B", "Classe C", "Classe D", "Classe E", são representados os valores que serão repassados para cada parâmetro representados na coluna anterior.

Benchmark	Parâmetro	Classe S	Classe W	Classe A	Classe B	Classe C	Classe D	Classe E
CG	no. of rows	1400	7000	14000	75000	150000	1500000	9000000
	no. of nonzeros	7	8	11	13	15	21	26
	no. of iterations	15	15	15	75	75	100	100
	eigenvalue shift	10	12	20	60	110	500	1500
EP	no. of random-number pairs	2^{24}	2^{25}	2^{28}	2^{30}	2^{32}	2^{36}	2^{40}
FT	grid size	64 x 64 x 64	128 x 128 x 32	256 x 256 x 128	512 x 256 x 256	512 x 512 x 512	2048 x 1024 x 1024	4096 x 2048 x 2048
	no. of iterations	6	6	6	20	20	25	25
IS	no. of keys	2^{16}	2^{20}	2^{23}	2^{25}	2^{27}	2^{31}	
	key max. value	2^{11}	2^{16}	2^{19}	2^{21}	2^{23}	2^{27}	
MG	grid size	32 x 32 x 32	128 x 128 x 128	256 x 256 x 256	256 x 256 x 256	512 x 512 x 512	1024 x 1024 x 1024	2048 x 2048 x 2048
	no. of iterations	4	4	4	20	20	50	50
BT	grid size	12 x 12 x 12	24 x 24 x 24	64 x 64 x 64	102 x 102 x 102	162 x 162 x 162	408 x 408 x 408	1020 x 1020 x 1020
	no. of iterations	60	200	200	200	200	250	250
	time step	0.01	0.0008	0.0008	0.0003	0.0001	0.00002	0.000004
(BT-IO)	write interval	5	5	5	5	5	5	5
	Gbytes written	0.0008	0.022	0.42	1.7	6.8	135.8	2122.4
LU	grid size	12 x 12 x 12	33 x 33 x 33	64 x 64 x 64	102 x 102 x 102	162 x 162 x 162	408 x 408 x 408	1020 x 1020 x 1020
	no. of iterations	50	300	250	250	250	300	300
	time step	0.5	0.0015	2.0	2.0	2.0	1.0	0.5
SP	grid size	12 x 12 x 12	36 x 36 x 36	64 x 64 x 64	102 x 102 x 102	162 x 162 x 162	408 x 408 x 408	1020 x 1020 x 1020
	no. of iterations	100	400	400	400	400	500	500
	time step	0.015	0.0015	0.0015	0.001	0.00067	0.0003	0.0001
UA	no. of elements	250	700	2400	8800	33500	515000	
	no. of mortar points	11600	26700	92700	334600	1262100	19500000	
	levels of refinements	4	5	6	7	8	10	
	no. of iterations	50	100	200	200	200	250	
	heat source radius	0.04	0.06	0.076	0.076	0.067	0.046	
DC	input tuples	10^3	10^5	10^6	10^7			
	no. of dimensions	5	10	15	20			

Fonte: NASA, 2014

Quadro 03: Parâmetros para cada uma das classes definidas no NPB 3.3

CAPÍTULO 3: **RESULTADOS ALCANÇADOS**

Nas seções seguintes, serão abordados todos os resultados adquiridos na pesquisa. Serão detalhados os ambientes de testes, os *benchmarks* utilizados para realizar os testes, testes estatísticos de hipóteses. Além disso, serão apresentados os trabalhos relacionados na área, que serviu de grande importância para adquirir conhecimento.

3.1 TRABALHOS RELACIONADOS

A busca por trabalhos relacionados é importante, pelo motivo que é preciso posicionar a pesquisa que se pretende fazer. Com o estudo destes trabalhos, é possível perceber detalhes que ainda não foram estudados e agrega-los como um diferencial na pesquisa que se deseja fazer.

Além de ser importante para busca da originalidade do trabalho, a pesquisa por trabalhos já realizados na área de estudo, proporciona um conhecimento adicional ao assunto que se deseja pesquisar. Estudando estes trabalhos, é possível encontrar relatos específicos dos experimentos, detalhes técnicos que poderão ser categóricos durante o andamento da pesquisa.

Nas seções seguintes serão apresentados alguns trabalhos relacionados encontrados na literatura.

3.1.1 Performance Evaluation of Container Based Virtualization for HPC Computing Environments

De acordo com Xavier et al. [a] (2013), o principal objetivo deste trabalho foi avaliar o desempenho em um ambiente virtualizado em container (virtualização a nível de sistema operacional), comparando os resultados de desempenho com um sistema virtualizado em *hypervisor*. Além disso, buscaram testar o uso de aplicações voltadas a HPC (*High Performance Computing).* Além do desempenho, ao mesmo tempo buscaram avaliar o nível de isolamento entre os processos dos vários sistemas virtualizados em nível de sistema operacional.

A motivação para realização desta pesquisa, é que maioria dos ambientes virtualizados não são usados para computação de alto desempenho. Ao mesmo tempo, parte dos trabalhos são voltados para ambientes virtualizados baseados em *hypervisor*. Portanto, um grande diferencial neste trabalho é a avaliação de desempenho em ambientes em que a virtualização seja baseada em container.

As ferramentas de virtualização utilizadas na pesquisa foram Linux Vserver, OpenVZ e LXC, todas ferramentas de virtualização a nível de sistema operacional. E como ferramenta para virtualização com *hypervisor*, foi utilizado a ferramenta Xen. Estas ferramentas foram instaladas em um ambiente composto por 4 servidores PowerEdge R610. Todos com processadores Intel Xeon 2.27 GHz E5520, com 8 núcleos de 8 MB de cache L3, 16 GB de memória RAM e adaptadores de rede *gigabit ethernet*. Em todos os nodos utilizou-se como sistema operacional o Ubuntu 10.04 LTS (Lucid Lynix), com versão de *kernel* 2.6.32-28, para manter a compatibilidade com todas as ferramentas de virtualização por container, e manter uma padronização no ambiente físico.

A avaliação de desempenho dos sistemas foi realizada por *benchmarks* específicos para cada componente da infraestrutura. STREAM foi utilizado para avaliar a largura de banda da memória RAM. Para desempenho de disco, foi utilizado o IOzone. Para avaliação da rede, foi utilizado o *benchmark* NetPipe. E para análise de desempenho de aplicações HPC, foi utilizado o pacote *benchmark* NPB 3.3.

De acordo com Xavier et al. [a] (2013), os resultados de desempenho de memória, CPU, rede e disco do ambiente ficaram próximos aos resultados obtidos em ambiente Nativo, ou seja, sem nenhuma virtualização. Porém, para uso em HPC, só é possível se o desempenho fundamental seja reduzido, mas isto resulta em um baixo isolamento e falta de segurança entre os sistemas. Dentre os sistemas avaliados LXC alcançou uma melhor avaliação para uso em HPC. Em comparação com o Xen e as outras ferramentas utilizadas, os resultados no desempenho dos quesitos avaliados não foram satisfatórios para o *hypervisor*.

3.1.2 **Towards Better Manegeability of Database Clusters on Cloud Computing Plataforms**

Neste estudo, de acordo com Xavier et al. [b] (2014), foi com a intenção de provisionar em clusters banco de dados em nuvem, e assim avaliar a capacidade neste ambiente. Além do desempenho, buscou-se avaliar o consumo de energia perante o desempenho gerado pelo sistema, e a eficácia de migração de instâncias ao vivo.

Esse trabalho foi motivado pela preocupação do uso de energia elétrica em um ambiente de processamento. Isto motivou a testar o desempenho de banco dados em um ambiente de computação em nuvem levando em consideração o consumo de energia deste processamento. É pretendido que os resultados desta pesquisa, sejam utilizados em criações de novas modalidades de serviços em computação em nuvem, como “*pay-as-you-go”*.

Em um ambiente composto por 3 servidores, onde 2 (dois) tinham processadores Intel Xeon de 2.27 GHz, com 8 (oito) núcleos com recurso HT (*Hyper-Threading),* 16 GB de memória RAM e 4 (quatro) placas de rede *gigabit ethernet*. E o restante incluíam processadores Intel Xeon X5690 4.46 GHz, com 6 núcleos com recurso HT, 64 GB de memória RAM e 4 adaptadores de rede *gigabit ethernet*, e um armazenamento SAN compartilhado com capacidade de 1TB.

Para ambiente de testes, foram utilizadas as ferramentas KVM para virtualização dos sistemas operacionais instanciados e plataforma de computação em

OpenStack Folsom. Para teste do banco de dados, utilizou-se um agrupamento Oracle RAC, que permite o acesso de várias instâncias em um único banco de dados, com o *benchmark* Hammerora, que gera transações TPCC e OLTP ao banco. E para catalogar a energia consumida no processamento, foi acoplado um multímetro durante a realização dos testes.

Durante os testes ao banco de dados, o *benchmark* simulou transações TPCC e OLTP de 5, 10, 30, 50, 100 e 200 usuários durante 30 minutos. O consumo de energia durante os testes em plataforma de nuvem e em ambiente Nativo se mantiveram semelhantes, sem grandes diferenças de consumo, porém o consumo da plataforma OpenStack foi menor.

Conforme Xavier et al. [b] (2014), os resultados foram bem positivos com relação a medição do consumo de energia, pois um ambiente virtual há um consumo consciente da energia elétrica. Sobre o desempenho do banco de dados, a plataforma OpenStack manteve um desempenho aceitável até os 30 usuários, a partir disso, o sistema começou a degradar. A conclusão com relação a degradação do sistema incide sobre o virtualizador KVM, porém uma infraestrutura física mais eficiente poderia melhorar alguns resultados.

Conforme os resultados, o recurso de migração em tempo real das máquinas funcionou, porém não pode-se concluir o motivo que o banco de dados parou de responder.

3.1.3 **A Performance Comparison of Container-based Virtualization Systems for MapReduce Clusters**

De acordo com Xavier et al. [c] (2014), o principal objetivo de análise neste trabalho foi avaliar o desempenho e o isolamento de ambiente com virtualização por container, executando aplicações MapReduce. O estímulo para esta pesquisa, foi que a ocorrência da falta de estudos sobre avaliações de cargas MapReduce em ambientes virtuais usando a virtualização por container.

A avaliações se basearam em implementação atuais das ferramentas de virtualização por container Linux-Vserver, OpenVZ, LXC, com uso de micro e macro-*banchmark*, onde o primeiro busca medir desempenho de componentes básicos do Hadoop e, o segundo busca a performance para o modelo MapReduce e todo o sistema.

Todos os sistemas foram provisionados em um ambiente composto por 4(quatro) nós idênticos com duplo processadores de 2.27 GHz com 8 núcleos de 8MB de cache L3, 16 GB de memória RAM, disco de 146 GB e placa de rede *gigabit ethernet*. Para o sistema operacional, foi necessário a compilação do *kernel* versão 2.6.32-28, por se apresentar mais compatível com as ferramentas e assim mantendo um ambiente homogêneo para os testes.

No conceito micro-*benchmarks*, foram testados vários componentes. O desempenho do sistema de arquivos HDFS, foi usado o *benchmark* TestDFSIO. Para o componente NameNode, foi usado o *benchmark* NNBench. E MRBench para teste da camada MapReduce. No conceito macro-*benchmark* foram usadas as ferramentas WordCount, Terasort e IBS.

De acordo com Xavier et al. [c] (2014), com os testes e resultados obtidos pode-se perceber que a virtualização é útil para ambientes MapReduce. Utilizando 2 (dois) contêineres por nodo, o uso de CPU apresenta um bom isolamento, porém, no uso de memória percebe-se uma degradação no desempenho nestes ambientes. No restante, os resultados se assemelham com um ambiente Nativo.

3.1.4 **Evaluation of HPC Applications on Cloud**

De acordo com Gupta, et al., (2011) o trabalho buscou avaliar aplicações de alto desempenho em infraestruturas de computação em nuvem. O trabalho foi motivado devido à grande aceitação que serviços de nuvem tem para execução de aplicações de negócio e para a WEB. Porém, *datacenters* para execução de aplicações de alto desempenho são projetados com um tipo de *hardware* especializado e isso instigou-os para aplicar testes de aplicações de alto desempenho

que com a utilização de máquinas virtuais na nuvem e assim avaliar o impacto sofrido durante os testes.

Foram utilizadas 3 plataformas diferentes para realização dos testes. A primeira Cluster Taub: plataforma física com comunicação infiband, Sistema Operacional Scientific Linux, 12 processadores Intel Xeon X5650 de 2.67 Ghz e memória de 48 GB. OpenCirrus, plataforma de nuvem, era composto por conexões de rede *gigabit ethernet* de 10 Gbps internos e 1 GB Gbps *ethernet* x-rack 4 processadores Intel Xeon5430 3.0 Ghz de 3.0 Ghz e 48 GB, o sistema operacional ubuntu 10.04. E a infraestrutura Eucalyptus, plataforma de nuvem com 2 QEMU Virtual CPU 10.04 de 2.67 Ghz com 2.67 Ghz e 6 GB, com placa de rede emulada 10/100/100 e KVM como virtualizador com Ubuntu 10.04

Como escolha dos *benchmarks* foi usado a suíte do NPB3.3-MPI compilado para 256 núcleos e como complemento foi utilizado um *benchmark* para simulação de cargas reais do ambiente, o Nqueens. Durante os testes, ocorreu que o NPB-MPI não executou seus testes em mais de 64 núcleos. Já o *benchmark* LU da suíte NPB-MPI e NAMD tiveram seus testes interrompidos devido à alta comunicação de rede, que acabou se tornando um gargalo no momento dos testes.

Como conclusão Gupta, et al. (2011) considera importantes as plataformas de nuvem voltadas para execução de aplicações de alto desempenho. Pois durante os testes, foram avaliados valores de processamento, utilizando como estimativa, os valores que provedores de nuvens públicas aplicam aos seus clientes, e desta forma constatou que a computação em nuvem se torna rentável e viável para determinadas aplicações de alto desempenho.

3.1.5 **Cloud Computing for parallel Scientific HPC applications: Feasibility of running Coupled Atmosphere Ocean Climate Models on Amazon's EC2**

A pesquisa realizada por Evangelinos, et al. (2008) teve como proposta inicial avaliar o comportamento de aplicações de alto desempenho utilizando instâncias de máquinas virtuais dos serviços oferecidos pela Amazon, na modalidade EC2.

E empresa Amazon que presta serviços relacionados a computação em nuvem, proporciona aos usuários 5 modalidades de servidores, e para realização do trabalho, fora analisada cada modalidade oferecida pela empresa, aonde acabou sendo escolhida 2 modalidades, as que acabaram se tornando mais rentável para os testes.

Com as duas Instâncias (Máquinas Virtuais) de capacidades diferentes, foi usado o *benchmark stream* para aferição da largura de banda da memória RAM. Para avaliação das aplicações paralelas, foi usado o *benchmark* desenvolvido pela NASA, NPB v3.3, especificamente usando as classes W e A com os *benchmarks* BT, CG, FT, IS, MG, SP, UA, onde o DC não fez parte dos testes. Para avaliação de desempenho de disco, utilizou-se o *benchmark* IOR em modo POSIX, realizando altas cargas nas unidades locais e em unidades NFS. Buscando explorar mais ainda os resultados de desempenho em unidades de armazenamento, os *benchmarks* do NPB v3.3 o BT-IO foram acrescentados junto ao conjunto dos testes.

Com os resultados dos testes foi possível perceber uma alta largura de banda para a instância mais básica da Amazon, pelo motivo de utilizar memória RAM DDR2. Porém, na instância com um processador mais superior, a largura de banda se manteve semelhante. Nos testes de desempenho de disco, é grande a diferença entre a leitura e a escrita, que de acordo com os testes, esteve relacionado ao uso dos núcleos de processamento. Com os resultados dos testes utilizando o BT-IO, a diferença do resultado entre as instâncias é cerca de o dobro.

Como conclusão, os testes usando as aplicações paralelas em uma infraestrutura da Amazon não foram satisfatórios, percebendo que o desempenho num geral foi considerado baixo, levando em consideração à clusters de alto desempenho. Nestes testes, o percursor do mau desempenho foi devido à baixa largura de banda e a latência alta na troca de mensagens. Contudo, os testes foram considerados animadores, pois os resultados puderam representar a execução de trabalhos em clusters de baixo custo, onde o desempenho pode ser inferior. Enfim, uma conclusão relevante é que em infraestruturas de nuvem, é possível utilizar conexões especializadas como a Myrinet ou Infiniband, podendo resultar em melhores resultados.

3.1.6 **VM Consolidation: A real case based on OpenStack Cloud**

A pesquisa realizada por Corradi, et al. (2012), primeiramente tem o objetivo de realizar um estudo bibliográfico com a finalidade de adquirir conhecimento mais específico, e ainda conseguir uma visão geral mais atualizada sobre a consolidação de máquinas virtuais. E por fim, avaliar o desempenho de uma nuvem utilizando a ferramenta OpenStack, levando com conta, a consolidação de máquinas virtuais como ponto principal da análise.

Segundo Corradi, et al. (2012), *Green Computing* foi uma das principais motivações para desenvolver uma pesquisa neste sentido, devido a preocupação com a eficiência energética em *dataceters*, e principalmente em infraestruturas de computação em nuvem, já que Eucalyptus e OpenStack eferecem soluções primitivas para o provisionamento e redirecionamento de máquinas virtuais em infraestruturas.

O trabalho utilizou uma infraestrutura de estações de trabalho convencionais e as máquinas virtuais instanciadas tinham configurações que não ultrapassavam 512 MB de memória RAM e 1 vCPU. Como ferramenta de administração a OpenStack, codinome Diablo, e utilizando o KVM como virtualizador. Oriundo da pesquisa bibliográfica, durantes os testes foi seguindo uma ordem para dar maior legibilidade e organização durante os testes. Primeiramente, foram realizados os testes de degradação da vCPU. Posterior a isso, observar a degradação de entradas e saídas, utilizando um servidor Apache nas máquinas virtuais. E por fim, a alta comunicação de rede utilizando o MapReduce e Iperf.

Para os testes de degradação de vCPU, as mesmas cargas de trabalhos foram sendo executadas em todas as máquinas virtuais e sendo gradativamente aumentadas. Desta forma, quando os níveis totais de todas as máquinas virtuais chegaram a valores de 200%, foi constatado a degradação real de desempenho, e um consumo energético de 75 Watts.

Já nos testes de desempenho de rede, os valores alcançados foram de 200 Mb, em conexões de gigabits, aonde neste teste de rede, foi constatado que a comunicação de rede nas máquinas virtuais, fazem com que os processos da

ferramenta KVM atinjam um alto uso de processamento nos ambientes não virtuais, provando que a comunicação de rede é extremamente custosa.

Com o entendimento de Corradi, et al. (2012) sobre o estudo da consolidação de máquinas virtuais, foi possível perceber que se domonstra eficiente o consumo energético aplicando estas formas de virtualização, porem deve ser usado com extrema cautela, pois em alguns testes, é comporvado que a comunicação de rede tem um alto consumo no processamento, acarretando em degradação do desempenho.

3.1.7 **IaaS Cloud Benchmarking: Approaches, Challenges and Experience**

IOSUP, et al.(2013) em seu trabalho buscou encontrar as principais abordagens e desafios para a realização de testes em nuvens do tipo IaaS, que de acordo com o autor, a avaliação de desempenho em nuvens IaaS de larga escala tem um propósito de seguir 3 tendências emergentes que são a computação paralela e em larga escala, onde a forma como a intensidade dos trabalhos estão se tornando diferentes, a duração de cada tarefa e a diminuição dos fluxos de trabalho estão sendo divididos através do uso de escalonadores.

Com isso, os principais elementos de um *benchmark* definidos por IOSUP, et al. (2013) através de estudos bibliográficos são: portabilidade, escalabilidade e simplicidade. Contudo, ainda o autor aborda questões metodológicas para os tipos de execuções de *benchmarks* em nuvens IaaS que são elencados desde os anos 90.

Segundo IOSUP, et al. (2013), é importante os testes e avaliações em nuvens IaaS devido à grande procura que está se tendo para as execuções de trabalhos científicos, e junto a isso, as aplicações de alto desempenho.

3.1.8 **Identifying key challenges in performance Issues in Cloud Computing**

De acordo com ZIA, et al. (2012), devido ao crescimento expressivo em serviços de computação em nuvem, a pesquisa teve como objetivo identificar os principais problemas de desempenho na área de computação em nuvem. Já que a avaliação de um ambiente deste tipo, será de grande proveito para usuários e

administradores que farão o uso dos resultados para melhorar suas infraestruturas e ainda poder definir sua colocação ao possível cliente.

O trabalho resumidamente teve uma base no estudo da bibliografia de diversos outros autores ligadas à área de computação em nuvem. Onde a pesquisa realizada por Zia, et al. (2012), buscou escolher trabalhos que envolvessem o maior número possível dos assuntos e áreas sobre a computação em nuvem, envolvendo desde as questões administrativas e burocráticas, até chegar aos estudos técnicos.

Resumidamente, o trabalho apresentou os diferentes aspectos para a busca de desempenho em computação em nuvem, como: Gestão de arquitetura, eficiência, confiabilidade, tempo de respostam qualidade dos serviços, empregando metodologias de desenvolvimento de software. Além da abordagem das questões de confiança entre consumidores e prestadores de serviços, a análise ainda aborda questões mais técnicas como a necessidade de melhorias em serviços de armazenamento e serviços de redes. E prevê ainda grandes perspectivas para computação de alto desempenho em infraestruturas de nuvens.

3.1.9 **A Component-Based Performance Comparison of Four Hypervisors**

A pesquisa sobre a avaliação de desempenho de virtualizadores (Hyper-V, VMware, KVM e XEN) realizada por HWANG, et al., (2013), surgiu devido à grande adoção destes tipos de ferramentas em meio ao mercado de computação em nuvem, outro fator importante é a *Green Computing*, que deve ser levado em conta quando no momento de uma implantação de um *datacenter*.

A pesquisa partiu para o uso de 4 grandes *hypervisors*, onde individualmente foram instalados em maquinas virtuais com capacidades de 2 GB de memória RAM, com uma vCPU e Linux Ubuntu 10.04. Os *benchmarks* usados para aferir o ambiente foram o ByteMark, para desempenho de processador. RAMSpeed, para cargas de trabalho sobre a memória RAM. Bonnie++ e Filebench para desempenho de disco.

Com à realização dos testes executados por HWANG, et al., (2013), acabou chegando a uma conclusão que é difícil definir nos casos avaliados um *hypervisor* que seja perfeito. A escolha sempre partirá das necessidades de cada ambiente, mas no geral o vSphere da VMware teve os melhores resultados. Na análise individual de cada *hypervisor* os testes constataram ainda que em tarefas de I/O, o Xen apresenta altas cargas de CPU para realização de pequenas operações de disco, e tem ainda resultados de *throughput*. O KVM apresenta despesas de memórias mais altas quando todos os núcleos virtuais estão ativos. Hyper-V tem uma perda de desempenho quando múltiplos núcleos são dedicados a pequenas execuções.

3.1.10 **Recommendations for virtualization technologies in high performance computing**

A pesquisa realizada Regola, et al., (2010) tem como principal objetivo a avaliação e comparação de desempenho de virtualizadores como O OpenVZ, Xen e KVM. E os grandes principais motivos para a realização dos testes, é justamente a *Green Computing* e a utilização de aplicações de alto desempenho em virtualizadores.

A infraestrutura utilizada nos testes foi com o propósito de avaliar os principais virtualizadores *open source* em infraestrutura próprias, com servidores de multiprocessadores, e ainda executar a comparação dos resultados de testes realizados em servidores de provedores de serviços de computação em nuvem como a Amazon. Os testes focaram no desempenho de rede, discos, I/O e na execução de aplicações paralelas como o NAS 3.3 OMP e NAS 3.3 MPI, que possuem um conjunto de *benchmarks.*

Com todos os resultados catalogados durante os testes, de acordo Regola, et al., (2010) a sobrecarga de CPU foi praticamente eliminada nas modalidades de paravirtualização, virtualização completa e virtualização por container. Porém, os resultados não foram satisfatórios nos testes de I/O. Os testes de aplicações paralelas mostraram-se economicamente viáveis nas execuções de cargas de trabalho em nuvens da Amazon.

3.1.11 **Análise e Comparação dos Trabalhos Relacionados**

Tabela 01: Comparação de trabalhos relacionados

Trabalhos	Infraestrutura	*Hardware*	Objetivo	*Benchmarks*	Resultado	Trabalhos futuros
(XAVIER [A], et al., 2013)	Maquinas virtuais com Ubuntu 10.04 LTS usando capacidade total do *host.* Virtualizadores: vServer, OpenVZ, LXC, Xen Server	(4) Dell PowerEdge R610, (2) Intel Xeon E5520 2.27 GHz. 16 GB RAM. Network Gigabit. Switch PowerConnect 5548	Avaliar desempenho de HPC. Isolamento. Desempenho do ambiente	Linpack, Stream, IOzone, NetPipe, NPB3.3-OMP (IS,EP,CG,MG,FT, BT, SP, LU)	LXC mais adequado para HPC. Máquinas virtuais (vServer, LXC, OpenVZ) desempenho similar ao Nativo. Xen não atinge os resultados da virtualização por container.	Avaliar desempenho e isolamento de sistemas baseados em container com outras cargas de trabalhos
(XAVIER [B], et al., 2013)	OpenStack Folsom. Virtualizador KVM Armazenamento SAN. Banco de dados de 10GB ORACLE RAC. Capacidade total do *host* físico.	(2) Processadores 2.27 GHz HT, 16 Gb RAM, (4) Placas de rede gigabit. (2) (2) Intel Xeon X5690 3.46 GHz HT, 64 GB RAM (4) placas de rede gigabit, SAN 1TB	Provisionar banco de dados na nuvem. Avaliar desempenho banco de dados. Consumo energético. Eficácia migração ao vivo de VMs	Hammerora. TPC-C	ORACLE RAC parou de responder após migração ao vivo. KVM não apresentou bom desempenho com a rede.	Estudar OpenStack e KVM. Estudar o recurso de migração ao vivo. Estudar outros sistemas de virtualização usando o ORACLE RAC. Levar em consideração o tempo de provisionamento nos próximos trabalhos.
(XAVIER [C], et al., 2013)	Virtualizadores OpenVZ, LXC, vServer. Cluster Hadoop, com HDFS Usando 2 Namenodes e 4 DataNodes. 2 VM por *hosts* (para testar isolamento)	(4) Nodos de (2) processadores 2.27 GHz, 16 GB RAM, 146 GB, Rede gigabit	Desempenho Mapreduce isolamento de ambientes virtualizados por containers.	TestDFSIO, NetPipe, NNBench, MRBench, Wordcount, Terasort IBS	LXC possui maior capacidade de restringir VMs. A virtualização por container atinge desempenho quase Nativo para testes com MR	Estudar o Isolamento em um nível de Rede. Aspectos *Green Computing* e desempenho em sistemas baseados em container.
(GUPTA, et al., 2011)	**Cluster TAUB**: 12 Xeon X5650 2.67 GHz, 48 GB RAM, QDR Infiniband, Scientific Linux. **OpenCirrus:** 4 Xeon E5450 3.00 GHz e 48 GB RAM 10 Gbps Ethernet Interna, 1 Gb ethernet x-rack, Ubuntu 10.04. **Eucalyptus Cloud:** 2 QEMU Virtual CPU, 2.67 GHz, 6 GB RAM, Rede Emulada gigabit, KVM *hypervisor*, Ubuntu 10.04. Sistemas de Arquivos NFS	**Cluster TAUB**: 12 Xeon X5650 2.67 GHz, 48 GB RAM, QDR Infiniband, **OpenCirrus:** 4 Xeon E5450 3.00 GHz e 48 GB RAM 10 Gbps Ethernet Interna, 1 Gb ethernet x-rack. **Eucalyptus Cloud:** 2 QEMU Virtual CPU, 2.67 GHz, 6 GB RAM, Rede Emulada gigabit,	Avaliar HPC em infraestrutura de nuvem.	NPB3.3-MPI (256 núcleos), NAMD, NQueens	Plataformas de nuvens vantagens econômicas para execuções de HPC, quando envolvem cálculos, com níveis menores de paralelismo.	Aplicação determinar de forma automática a plataforma para execução. Estender os testes deste trabalho para nuvens com melhores conexões de redes.

(EVANGELINOS, et al., 2008)	Instâncias m1.small Instâncias c1.medium Virtualizador Xen NFS, PVFS2, GlusterFS, SSHFS	Capacidade de processamento similar Opteron 1.0-1.2 GHz	Avaliar desempenho de HPC em modalidade de servidores Amazon EC2	Stream, IOR, NPB 3.3 (BT-IO, BT, CG, FT, IS, LU, MG, SP, UA) nas classes A e W	Desempenho equiparado a clusters de baixo custo	#
(CORRADI, et al., 2012)	1 vCPU, 512 MB RAM. Virtualizador KVM. OpenStack Diablo. Cargas com MapReduce. Servidores Apaches	Estações de trabalho: Core Duo E7600, 3.06 GHz, 4GB de RAM, 250 GB disco	Estudo da literatura. Avaliação desempenho de nuvem OpenStack, levando em consideração consolidação de VM.	Iperf, Apache *benchmarking*	Degradação de vCPU ao executar tarefas complexas. Consumo de 75W com duas VM com processamento a 99%. Comunicação de rede virtual gera altas eleva o processo KVM no sistema físico. Viável no consumo energético, delicado para execuções com alto uso de processamento.	Como a consolidação de servidores afeta serviços principais e o papel de SLAs na decisão deste processo. Identificação automática de cargas de trabalho (introduzidas em rede ou CPU, por exemplo) para melhor prever interferências de consolidação de VM. Aumentar a capacidade da nuvem OpenStack, para assim aumentar os testes de cargas, e migração automática ao vivo para economia de energia.
(HWANG, et al., 2013)	Maquinas com 1 vCPU, 2GB RAM, Ubuntu 10.04 LTS (Kernel 2.6). 10 GB para cada virtualizador, KVM e Xen instalado em mesma partição.	Intel Xeon 5160 3.00 GHz, 8 GB RAM, disco SAS LSI 1064E 3 Gbps, (2) placas de rede gigabit *ethernet*.	Avaliar desempenho de *hypervisor* (Hyper-V, KVM, vSphere, Xen)	Bytemark, RAMSpeed, Bonnie++, FileBench, NetPerf, FreeBench,	vSphere melhor desempenho nos testes.	#
(REGOLA, et al., 2010)	Amazon EC2 com EBS, 10 Gb *ethernet.* Virtualizadores com disco SAS, sistemas de arquivos ext3. Sistema Operacional virtual RedHat 5.4, rede 1 Gb *ethernet*, Inifiband.	(4) Dell R610 (2) Xeon E5520 2.27 GHz, 24 GB RAM, 73 GB SAS, rede ethernet gigabit e um Qlogic 7240 Infiband. Switches Extremes Summit 450 e Qlogic Silverstorm 9040	Avaliar desempenho de *hypervisor* (KVM, Xen, OpenVZ) executando cargas de trabalho OpenMP e MPI.	NPB3.3-OMP, NPB3.3-MPI (EP, MG, CG, FT, IS, BT, SP, LU	I/O não apresentou bons resultados. Carga de CPU praticamente eliminada nos testes com virtualização. OpenVZ apresentou desempenho quase Nativo. Amazon EC2, plataforma economicamente viável para execução de HPC.	#
TRABALHO ATUAL	**OpenStack Havana**: (4) vCPUs, 4 GB RAM, 10 GB LVM, rede megabit Open vSwitch, Ubuntu Server 12.04 LTS. **OpenNebula 4.7.8**: (4) vCPUs, 4 GB RAM, 10 GB NFS, rede megabit, Ubuntu Server 12.04 LTS. Virtualizador KVM.	(8) Estações de trabalho: Core i5 650 3.20 GHz, 4 GB RAM DDR2, 250 GB SATA II, Fonte ATX 200w. Switch 10/100	Avaliar e comparar alto desempenho em nuvem utilizando estações de trabalho. Analisar o impacto de aplicações paralelas em diferentes ferramentas de administração de computação em nuvem. Comparar o desempenho com o ambiente Nativo.	NPB3.3-OMP (BT, CG, EP, IS, MG, UA, CG, FT, LU, SP), NPB3.3-MPI (BT, CG, EP, FT, IS, LU, MG, SP), STREAM, IOzone, IPERF, LINPACK	#	Avaliar as ferramentas de computação em nuvem com outros virtualizadores. Utilizar outros *benchmarks.* Analisar e comparara o consumo energia das ferramentas. Avaliar testes com ou outros sistemas de arquivos distribuídos. Realizar testes de desempenho com a máquina virtual em migração. Realizar testes com redes de alto *throughput*.

Fonte: Maron, Griebler. 2014.

Na Tabela 01, somente foram relacionados os trabalhos que usaram uma infraestrutura e alguma ferramenta em específico. Porém, não menos desmerecidos, existem os trabalhos que se basearam em um estudo bibliográfico sobre avaliação de desempenho, computação em nuvem e virtualização.

O estudo da literatura, contribui com fatores que são importantes para o desenvolvimento de uma pesquisa em campo, posicionando, determinando e até prevendo situações que poderão ser enfrentadas durante a prática da pesquisa. Um importante trabalho realizado por Thomé, Hentges, Gribler (2013), mostra a necessidade e a importância de um estudo bibliográfico. O trabalho realizado, buscou intensamente o relato da bibliografia das ferramentas OpenStack e OpenNebula e demais ferramentas para aplica-las em uma infraestrutura, servindo de grande valia para este trabalho.

Além de trazer um intenso estudo bibliográfico das ferramentas OpenStack e OpenNebula, pode-se comprovar na prática que as ferramentas *open source* de computação em nuvem não se restringe às características de *hardware*, podendo ser utilizadas em estações de trabalho.

Quanto aos detalhes teóricos de trabalhos, Iosup, et al. (2013), aborda a questão de avaliação de desempenho de nuvens do tipo IaaS, define que a perspectiva não é somente em unir a infraestrutura com os *benchmarks,* executa-los, coletar os dados, e concluir qual a melhor infraestrutura obteve melhor desempenho. Em nuvens do tipo IaaS, desempenho deve ser uma característica importante, e como qualquer outra infraestrutura, é essencial o amadurecimento, e a sustentação destas características.

Entende-se que *benchmark* e a maneira tradicional de verificação de desempenho, principalmente em nuvens computacionais. Mas o Iosup et al (2013) aborda que a questão de avaliação vai muito além da prática, levando sempre em conta as questões metodológicas. Com a entrada de serviços de computação em nuvem do tipo IaaS no mercado, é esperado uma nova abordagem, onde a realização de testes seja aplicada com visão para o cliente.

Com isso, definem várias abordagens sobre a aplicação de *benchmarks* em nuvens IaaS que precisam ser superadas, onde uma delas, está ligada as questões metodológicas, que aborda a dificuldade na execução de testes de avaliação em nuvens computacionais. Principalmente, devido a sua elasticidade, dificultando uma média mais estável com relação ao desempenho, mas outra ainda é a complexidade que envolve seus componentes de suas infraestruturas. O trabalho teve um estudo voltado para aplicação das diversas abordagens em infraestruturas de grande porte.

Zia, et al. (2012) também aprofunda sua análise em um estudo da bibliografia. Diferentemente do estudo de Iosup, et al. (2012) aonde aborda também um relato da experiência dos autores. Zia, et al (2012) propõe seu estudo com uma análise em trabalhos realizados por outros autores, partindo desde as questões burocráticas e metodológicas até as questões técnicas nas infraestruturas de computação em nuvem, expondo ao final, diferentes técnicas para busca de desempenho de computação em nuvem.

Expressivamente, os dois trabalhos buscam abordagens em estudos realizados por outros autores, e posteriormente realizando uma análise abordando desafios e pontos importantes sobre a avaliação de desempenho em computação em nuvem. Os trabalhos serviram para trazer abordagens para procedimentos e entender mais sobre a realização de testes em nuvem.

Como resultado do estudo bibliográfico realizado para o trabalho atual, pode-se compreender que aplicações paralelas necessitam de alto poder computacional, pois envolvem grandes cargas de trabalhos em diversos segmentos, com um propósito científico. Grande parte dos trabalhos apresentados até aqui, utilizam infraestruturas dedicadas com conjunto de *hardware* com capacidades consideráveis de processamento, usando clusters com servidores e com arquiteturas multi-processadas, redes com grande *throughput,* e grandes quantidades de memória RAM. Até mesmo utilizando infraestruturas oferecidas por empresas que fornecem serviços de computação em nuvem do tipo IaaS, e assim realizando testes que envolvem a análise e comparação de infraestruturas virtuais e nativas, e desempenho de aplicações.

No quesito de infraestrutura, apenas a pesquisa de Corradi, et al (2012), que saiu deste paradigma e utilizou em seus testes estações de trabalhos, onde a capacidade de processamento é mais limitada em relação às infraestruturas dos outros trabalhos. No trabalho, a proposta é avaliar o nível de desempenho da consolidação de máquinas virtuais executando *benchmarks* para avaliar um servidor de aplicação e contrapondo isto ao consumo energético em uma infraestrutura com a ferramenta OpenStack e a sua degradação do desempenho, usando um *hardware* mais limitado.

O que difere o trabalho de Corradi, et al (2012), é que neste trabalho foi proposto a utilização de uma infraestrutura mais limitada em processamento para execuções de cargas de trabalho que comumente são executadas em infraestrutura com grande capacidade computacional, executando estas cargas em um ambiente virtual e Nativo, e avaliando o desempenho usando versões mais atuais de ferramenta de computação em nuvem, e como principal, comparar os resultados entre as ferramentas.

Como foi relatado no capítulo anterior deste trabalho pode ser comprovado pelo estudo dos trabalhos relacionados e na implantação de ferramentas de administração de nuvens, que infraestruturas do tipo IaaS utilizam ferramentas de virtualização. Portanto, o estudo apresentado pelos trabalhos abordados aqui, de igual forma serviram para entender na descrição da prática, como são realizados os testes em ferramentas de virtualização, e poder conhecer alguns resultados obtidos pelos testes realizados.

O trabalho de Regola, et al. (2010), teve como objetivos, a comparação de performance de ferramentas de virtualização de diferentes modos de operação, avaliando e contrapondo os resultados de testes executados em um ambiente local, com um ambiente de um provedor de serviços de nuvem IaaS. O trabalho se torna importante devido a sua abordagem com os outros virtualizadores em que se é avaliado, e os resultados obtidos pelos virtualizadores. Porém, em alguns testes existem uma disparidade na avaliação aplicada pelos autores entre o ambiente local e a infraestrutura do provedor, aonde a própria influência do ambiente é questionada quanto a comparação dos resultados de testes locais. Relacionado o trabalho de

Regola et al. (2010) mostra que este trabalho, primeiramente busca uma homogeneidade do ambiente, para não haver vantagens em determinados ambientes durante os testes, mas contraponto os resultados com uma infraestrutura de computação em nuvem de um provedor de serviços.

Hwang, et al (2013), busca avaliar o desempenho dos principais virtualizadores do mercado em uma infraestrutura com grande capacidade operacional. No trabalho de Hwang et al (2013), ele busca manter a homogeneidade no ambiente dos testes, alocando as máquinas virtuais em uma infraestrutura com capacidade considerável, diferenciando do objetivo deste trabalho, que propõe como ambiente Nativo, o mesmo do virtual, e assim avaliando a performance de cada ambiente.

Os trabalhos de Xavier [B] et al, Corradi, et al (2012), abordam questões importantes, levada muito em conta em grandes *datacenters*, que é a eficiência do desempenho, considerando o consumo de energia. Comparando o trabalho realizado pelos autores, com este aqui proposto, é que os resultados de desempenho não serão relacionados com o consumo de energia durante os testes. Mas algo que estamos cientes e que em nenhum dos trabalhos foi posicionado no momento da descrição da infraestrutura, que a potência da fonte de alimentação dos equipamentos usados pode se tornar um fator prejudicial no desempenho dos testes. Isto deverá ser analisado com mais critério em trabalhos futuros

Maioria dos trabalhos são relacionados a virtualização. Alguns abordam questões gerais referenciando a computação em nuvem, onde existem um questionamento em seus objetivos, quanto aos resultados de testes em ambientes de nuvem. Porém, torna quase semelhantes no sentido da avaliação de desempenho, diferentemente do estudo proposto neste trabalho.

Observa-se que existe uma lacuna entre o trabalho aqui proposto com relação aos outros abordados no estudo bibliográfico dos trabalhos relacionados. Inicialmente, este trabalho busca um ambiente menos convencional para execução de aplicações paralelas. Em seguida, os desempenhos das aplicações são questionados. E por fim, e não menos importante, a utilização de estações de

trabalhos convencionais com capacidades idênticas, e a influência de determinadas ferramentas de computação em nuvem nos resultados dos testes.

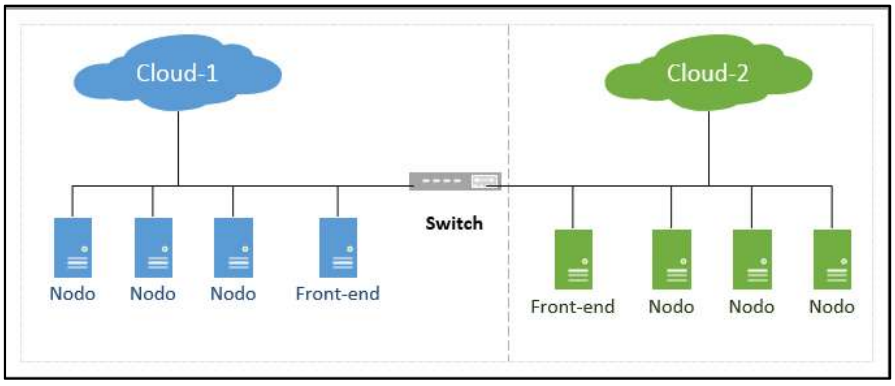
A Tabela 01, serve para auxiliar na comparação com o trabalho proposto aqui, os objetivos, infraestruturas, resultados dos outros trabalhos. No entanto, há de se considerar, e que pode ser comprovada pela Tabela 01, que neste trabalho estamos utilizando versões mais novas das ferramentas de computação em nuvem, e estamos de fato, mantando o ambiente virtual, o mais próximo possível do ambiente Nativo.

3.2 INSTALAÇÃO E CONFIGURAÇÃO DOS EXPERIMENTOS

3.2.1 **Ambientes de testes**

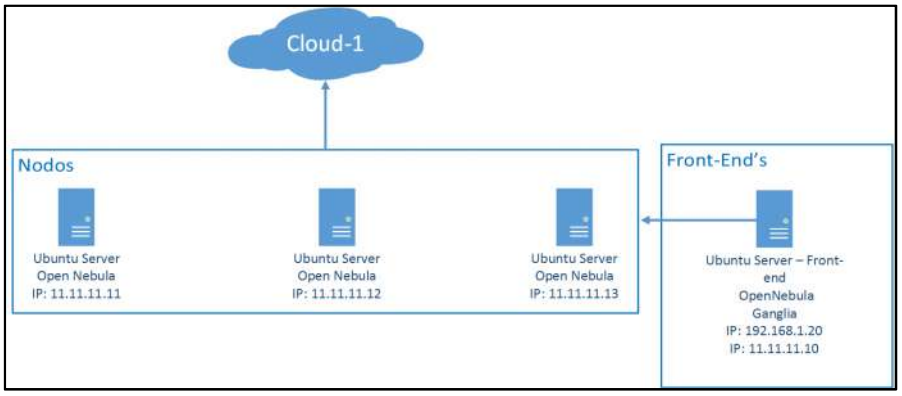
A infraestrutura utilizada nos testes é composta por 8 (oito) máquinas idênticas. As características técnicas das estações são: Intel Core i5 650 com 3.20 GHz, memória RAM de 4 GB DDR3 de 1333 MHz, disco de 500 GB operando em sata II, todos os nodos operando em rede 10/100 Mbits.

Como mostra a Figura 25, na disposição física dos equipamentos eles estão conectados em um mesmo *switch*, porém as configurações de endereçamento IP os qualificam em redes diferentes.



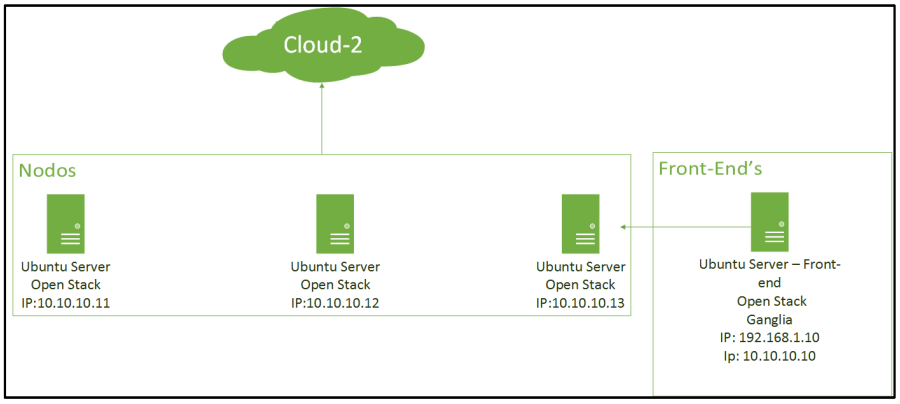
Fonte: Maron, Griebler. 2014
Figura 25: Infraestrutura de testes.

Na Figura 26 é possível ver a infraestrutura da Cloud-1. Nela estão alocados os principais serviços referente a ferramenta OpenNebula, e serviços essenciais para o funcionamento de uma estrutura de *cluster*.



Fonte: Maron Griebler
Figura 26: Infraestrutura da Nuvem 1.

Na Figura 27 é possível ver a infraestrutura da Cloud-2. Nela estão alocados os principais serviços referente a ferramenta *OpenStack*, e serviços essenciais para o funcionamento de uma estrutura de *cluster*.

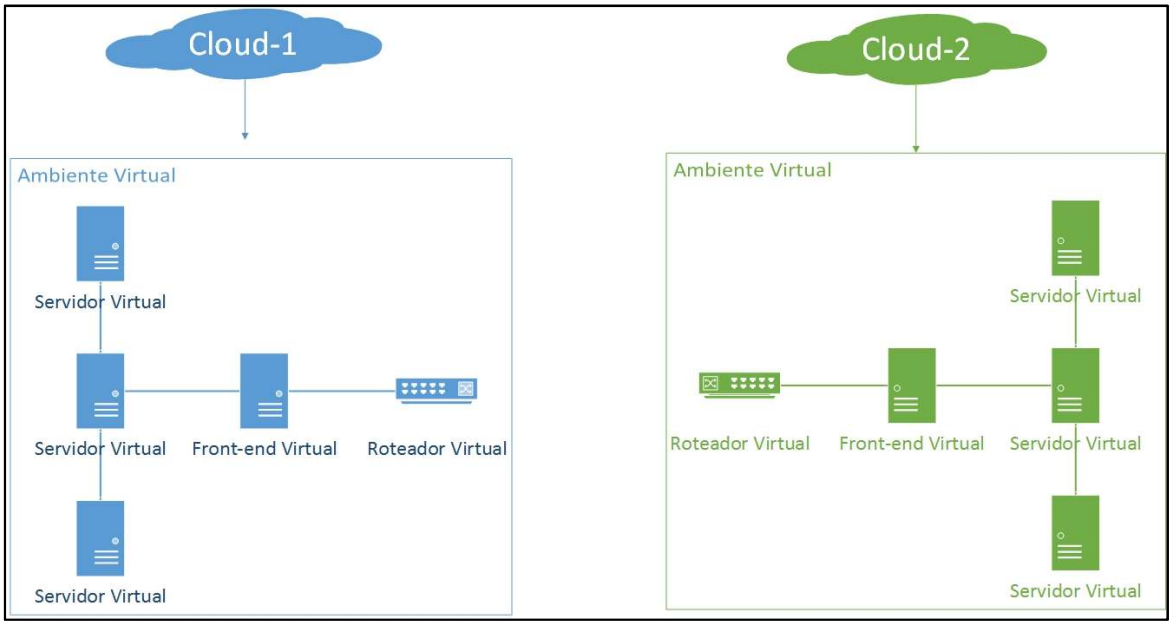


Fonte: Maron, Griebler, 2014
Figura 27: Infraestrutura da nuvem 2.

É necessário levar em consideração a disposição do ambiente virtual alocado nos nodos. Pois de acordo com nossos objetivos, será analisado o desempenho do ambiente virtual e do ambiente Nativo.

A ideia proposta é que cada máquina virtual seja o mais semelhante possível do ambiente Nativo, para que assim, as ferramentas responsáveis pelo

gerenciamento, cedam a capacidade necessária para os testes. Sendo assim, o *layout* da infraestrutura virtual está descrito na Figura 26.



FONTE: Maron, Griebler. 2014.

Figura 28: Ambiente virtual de testes.

Na Figura 28 consta uma representação do ambiente virtual dos testes. Este ambiente é composto por máquinas virtuais que unicamente são provisionadas em cada nodo, sendo um *front-end* para coleta dos resultados. Portanto, será necessário 3 (três) servidores virtuais (1 (um) em cada nodo), e 1 (um) *front-end*, e ainda um roteador virtual, que é necessário para que as máquinas virtuais se comuniquem entre si e com o ambiente externo.

A criação do ambiente virtual pode ser vista no Apêndice A, referente a criação de instância no OpenStack. E no Apêndice B, no qual refere-se à criação de instâncias no OpenNebula

3.2.2 **Instalação OpenStack**

A instalação do OpenStack inicia-se no momento em que o sistema operacional está devidamente instalado e atualizado. Neste caso utilizou-se a versão Ubuntu Server 12.04 como sistema operacional, por ser um dos sistemas compatíveis com a versão OpenStack.

A versão escolhida para implantação da nuvem, foi a OpenStack Havana, está que foi lançada no segundo semestre de 2013, sendo a versão mais estável até o momento da escolha.

De acordo com o estudo sobre a ferramenta OpenStack, foi possível perceber a grande quantidade de componentes que são necessários para seu funcionamento. Durante a instalação, foi seguido as documentações oficiais da ferramenta, porém com o decorrer dos processos aconteceram algumas inconsistências e então foi necessário recorrer a algumas outras fontes para consulta como, manuais, dicas e tutoriais de entusiastas que tiveram alguma experiência com a ferramenta, onde nenhum material foi encontrado na língua portuguesa.

Para iniciar a instalação da versão Havana, é necessário configurar os repositórios desta versão no sistema operacional. Pois somente assim, estará garantido o *download* dos componentes corretos para funcionamento nesta versão.

Os componentes instalados durante o procedimento foram MySQL sendo o gerenciador do banco de dados. RabbitMQ como *middleware* para controle de mensagens entre os serviços. Instalação de componentes como VLAN, *bridge-utils*. Instalação do Keystone, usado para controle de permissões para os diversos serviços do OpenStack. Instalação do Glance, no qual faz o gerenciamento de imagens e *snapshots* de instâncias. Neutron que faz a criação, gerenciamento e controle de redes e roteadores virtuais. Também, em conjunto com o Neutron o OpenVSwitch que permite a criação de interfaces virtuais no sistema operacional. Também a instalação do *hypervisor* KVM e do componente Nova, que juntos gerenciam recursos das máquinas virtuais. Cinder que faz um controle específico sobre as unidades de armazenamento utilizados pelas máquinas virtuais. E o componente Horizon, a interface gráfica acessível por um navegador de internet, ao qual permite controle menos complexo aos componentes instalados e as instâncias da infraestrutura.

Resumidamente foram descritos os procedimentos para a instalação da ferramenta OpenStack. Detalhes sobre arquivos de configuração e comandos utilizados estão descritos no Apêndice A deste trabalho.

3.2.3 **Instalação OpenNebula**

A versão escolhida para instalação da ferramenta OpenNebula foi a versão 4.7.80, a versão mais estável no momento da escolha desta ferramenta. A ferramenta seguiu o mesmo ambiente usado na ferramenta OpenStack. Porém, no estudo da ferramenta realizado no Capítulo 2, pode-se perceber uma diferença muito grande na composição dos componentes de cada uma das ferramentas. O OpenStack tem um número considerável de componentes que gerenciam determinados recursos, a OpenNebula segue um caminho contrário, tendo um número bem reduzido de componentes que necessitam de configurações no momento de sua instalação.

Portanto devido ao número de componentes menor que a outra ferramenta avaliada, a OpenNebula exigiu um tempo menor de configuração. Os procedimentos se basearam em documentos oficias de distribuição da OpenNebula. Porém, em algumas etapas foi necessário a consulta em algumas outras fontes de pesquisas, para deixar mais nítidos os procedimentos que deveriam ser seguidos e resolução de alguns problemas em determinadas etapas do processo. Todas as fontes consultadas além das documentações oficiais, eram de sites que continham manuais e dicas de entusiastas que tiveram alguma experiência com a ferramenta OpenNebula.

Inicialmente, na configuração do *frontend* foi instalado um recurso para criação *bridges*, o bridge-utils. Através deste componente que são realizadas as configurações de interfaces de redes como *bridges,* pelo meio da alteração do tradicional arquivo de interfaces do sistema operacional Linux.

O OpenNebula necessita da criação de um usuário específico para uso em seus componentes, portanto, foi realizado a criação de um grupo e usuário exclusivo, e a definição de um diretório privilegiado para este usuário criado.

O NFS, é usado para armazenamento distribuído entre o restante dos nodes da infraestrutura. Através deste armazenamento, é necessário a criação de chaves públicas para o acesso sem senha entre os nodes.

Antes da instalação efetiva do OpenNebula, é importante preparar o ambiente do sistema operacional com instalação de uma grande variedade de bibliotecas e pacotes, que são importantes para o funcionamento dos componentes da ferramenta. Também é necessário a instalação de um gerenciador de banco de dados, que neste procedimento foi usado o MySQL.

Antes da instalação é preciso informar aos scripts de instalação a alteração para utilizar MySQL como gerenciador do banco de dados, invés de SQLite. E após este procedimento, é executado um script que acompanha os arquivos baixados do site, aonde no momento da execução, são repassados alguns parâmetros básicos como usuário (oneadmin), grupo (oneadmin) e diretório exclusivo criado para a ferramenta (/var/lib/one).

Todos os componentes já são instalados com a execução do script, porém é preciso a alterações de alguns arquivos de configurações para efetivamente entrarem em operação. Os detalhes de toda a instalação dos componentes são descritos no Apêndice B.

3.2.4 **Adversidades durante as configurações das ferramentas OpenStack e OpenNebula**

As ferramentas OpenNebula e OpenStack são ferramentas *open source* para a implantação de plataformas de nuvens do tipo IaaS, tanto para uso comercial, didático e até mesmo doméstico. Porém, como qualquer outra ferramenta ou serviço que necessite ser configurado em um ambiente, o mesmo pode em alguns casos, ocorrer erros nas etapas de configurações e instalações.

As ferramentas avaliadas nesta pesquisa, não saíram deste contexto, apresentando muitas dificuldades em algumas etapas do processo de configuração, em especial a ferramenta OpenStack. No início da implementação desta ferramenta, através do estudo de tutoriais e manuais, estimou-se que dentre alguns dias a ferramenta estivesse com o mínimo dos recursos em operação. Aonde isso se provou ao contrário, levando alguns meses além do previsto para entrar em operação e entender mais detalhes de seu funcionamento. Durante todo este tempo, várias vezes

a melhor opção para prosseguir com as tentativas de configuração, foi a reinstalação padrão do sistema operacional, pois em alguns momentos, era impossível prosseguir pois vários arquivos e componentes tinha sido instalado, o que tornou duvidosa se isto era de fato a solução para os problemas.

Para a implantação da ferramenta OpenStack na infraestrutura proposta nesta pesquisa, inicialmente optou-se em usar a mesma versão utilizada no trabalho realizado por THOMÉ, et al. (2013), utilizando a versão Folson. Pois o objetivo em criar um ambiente totalmente do início, permitiria novas configurações, aonde poderia ser importante no decorrer dos objetivos propostos no trabalho. Porém, as adversidades começaram a surgir nas configurações iniciais dos serviços básicos da ferramenta, como Nova apresentando erros ao iniciar os serviços, instâncias que eram criadas mas que na metade do processo de criação retornavam com erros. E após uma grande pesquisa para encontrar soluções, onde os problemas acabaram não sendo solucionados, isto forçou a partir para uma versão mais atual.

Após novamente realizar a preparação do ambiente para iniciar as configurações do OpenStack versão Grizzly, obteve-se maior sucesso em comparação com a versão usada anteriormente. Porém, quando achou-se que a ferramenta estava pronta para inciar os testes de avaliação de desempenho, é que o problema surgiu. Como as ferramentas necessitam constantemente o acesso a redes externa, e obviamente acesso ao nodos restantes da infraestrutura virtual das ferramentas, é onde o problema ocorreu. Nenhuma instância criada conseguia comunicava-se com a rede externa (internet) e com o restante dos nodos virtuais.

A versão Grizzly utiliza o Quantum como principal serviço que controla as ações ligada da rede. No entanto ao analisar que na versão subsequente, Havana, o serviço Quantum foi descontinuado, e como sucessor o Neutron assumiu seu lugar. Porém, não foi somente este fator que instigou para a mudanção de versão, a intensiva pesquisa para tentativa de encontrar a solução do problema ligado a rede, percebeu-se que outros usuários tinham um problema muito semelhante, porém ainda sem solução, ou até mesmo, em algumas ocorrências a orientação era de usar o componente sucessor do Quantum. Então, isto definitivamente fez com que todo o

ambiente fosse novamente preparado, e assim inciou-se as instalações com a versão Havana.

De forma geral, desde a versão Folsom, seguindo pela Grizzly, até chegar na versão definitiva para o ambiente, as adversidades estiveram ligadas aos seguintes itens:

- Iniciação dos componentes (Nova, Cinder, Horizon).
- Autenticação dos componentes com o Keystone.
- Autenticação com o gerenciador de banco de dados MySQL.
- Criação de imagens, e discos virtuais.
- Criação de instâncias.
- Criação de redes e roteadores virtuais.
- Acesso a redes internas e externas.
- Acesso através de conexões SSH entre ambiente Nativo e o virtual.
- Serviços de redes inoperantes após reinicialização do node físico.

Com a instalação da versão Havana, pode-se perceber uma evolução significativa desde a versão Folsom. Através da prática, percebeu-se que as ferramentas estavam mais maduras, com novos componentes e que alguns problemas que antes eram mais corriqueiros no processo de configuração, tinham sido solucionados, mas que não tirou a complexidade de sua configuração.

Na configuração da ferramenta OpenNebula, o tempo de implementação foi menor, mas que exigiu o mesmo esforço para o aprendizado aplicado na ferramenta OpenStack. Seu tempo de implementação foi menor, em poucos dias a ferramenta estava instalada e com uma infraestrutura virtual já configurada.

Na ferramenta OpenNebula, as adversidades não foram com tanta intensidade como a ferramenta OpenStack. A OpenNebula em sua estrutura de componentes apresenta um número menor de itens e arquivos que precisam ser configurados, e isto facilitou um pouco o processo inicial. Os problemas com a ferramenta OpenNebula estiveram ligados ajustes no sistema operacional, e alguns erros na inicialização de componentes. Uma experiência que pode ser importante no

momento da configuração da ferramenta OpenNebula, é que a comunicação entre os nodos do *cluster* é realizada por conexões SSH. Isto significa que todos os nodos devem ter a comunicação através deste tipo de conexão, e não exigir senha durante este processo. Caso isto não ocorra, vários erros podem ocorrer durante o processo de configuração.

Durante a configuração dos *benchmarks* e do ambiente virtual, incluindo também as redes e o monitoramento das instâncias, surgiu um problema que de fato tornou-se um mistério para solucioná-lo. Este problema estava ligado à ferramenta Ganglia, usada para monitoramento de recursos físicos do ambiente computacional, e ela de fato não conseguia monitorar as instâncias virtuais que precisavam ser monitoradas. Isto foi um problema que não foi possível solucionar e optou-se em apenas monitorar os recursos do ambiente Nativo. Entretanto, no momento da execução dos testes do *benchmark* NPB–MPI, o mesmo executou seu ciclo de execução de um processo normalmente. Mas quando o número passou para dois processos, simplesmente as execuções ficaram paralisadas, continuavam consumindo recursos, mas que após longas horas de espera, percebeu-se nenhum progresso nas execuções.

Achou-se que os problemas com o Ganglia e o NPB-MPI, poderiam ter ligações com algum componente que não tenha sido configurado corretamente, ou até mesmo a falta de algum. Mas de certa forma, não foi encontrado uma solução cabível, e então, o ambiente novamente foi reconfigurado, desde a instalação inicial do sistema operacional até os pacotes da última versão do OpenNebula, aonde após a instalação, os problemas que antes havia ocorrido, não puderam ser vistos nesta nova instalação.

3.2.5 **Criação do ambiente de testes nas ferramentas OpenStack e OpenNebula**

Com as ferramentas instaladas e em execução, partiu-se para a criação dos ambientes que seriam usados no decorrer dos testes nas duas ferramentas. Como propósito deste trabalho é avaliar o alto desempenho em diferentes

ferramentas, procurou-se manter o máximo de uniformidade entre cada um dos ambientes OpenStack e OpenNebula, e também o Nativo e o virtual.

Portanto, para os testes do ambiente Nativo os recursos de *hardware* se mantiveram os mesmos, com dedicação total aos testes. Foi apenas necessário adequar o ambiente de *software* para que as execuções dos *benchmarks* fluíssem com mais naturalidade. Essas adequações são como a autenticação entre os nodos da infraestrutura sem a necessidade de digitar as credenciais de *login*. Outro recurso que se fez necessário em ambos os ambientes, é a utilização de um diretório compartilhado para armazenamento dos resultados das execuções. Este diretório foi implementado usando o recurso NFS. Além da ferramenta OpenNebula usar este recurso como parte de seus componentes, foi necessária à instalação deste na infraestrutura OpenStack, criando um diretório único compartilhado em cada uma das ferramentas.

No ambiente virtual, a criação dos ambientes também buscou-se manter a mesma homogeneidade de recursos e adequações para a execução dos testes. Devido as diferentes características entres as duas ferramentas, nem todos procedimentos para criação e definição do ambiente virtual se mantiveram iguais entre cada umas das ferramentas.

Para criação do ambiente no OpenStack foi usado o recurso de criação de discos com as imagens dos sistemas operacionais já inclusos (Ver Apêndice A, seção A.4). No qual este procedimento permite que no momento da criação de uma instância, seja usado este disco contendo o sistema operacional Linux. Na definição dos recursos físicos para a instância foi definido as mesmas capacidades conforme o ambiente Nativo.

Na criação do ambiente OpenNebula, o procedimento de criação de discos virtuais LVM é um procedimento padrão, e já incluso automaticamente no processo de criação de instâncias. Entende-se com a prática do trabalho com a ferramenta, é que a mesma executa a criação de discos para cada máquina virtual, tanto que na criação de uma instância durante os testes, uma única máquina demorava um tempo considerável.

Uma característica que não foi possível manter semelhante entre os dois ambientes de nuvem, é as interfaces de redes e as próprias redes. No OpenStack, toda a implementação é feita através dos componentes que criam as interfaces e redes virtuais. Já na OpenNebula, a implementação das redes e interfaces das instâncias aparentemente são implementadas nas interfaces *bridges* criadas no momento da configuração dos componentes e dos nodos, e utilizando um *driver* básico para criação das interfaces, denominada na ferramenta como rede *Dummy*.

Sobre os ambientes em questão, o Nativo se coloca sobre uma infraestrutura utilizando estações de trabalho convencionais com uma capacidade de processamento mais limitada. Entretanto, os objetivos se voltam para os resultados das ferramentas, e não sobre o tipo de infraestrutura utilizada e para isso o ambiente Nativo serve como um parâmetro para os resultados com as duas ferramentas.

O propósito das ferramentas de administração em nuvem é provisionar instâncias com um objetivo de otimizar recursos, alocando se possível, mais de uma máquina virtual em um único node. Porém, principalmente para as aplicações paralelas, a ideia é conseguir utilizar o máximo de recursos disponíveis em ambos os ambientes, este é o motivo que se explica a alocação de uma máquina virtual utilizando a capacidade total dos nodes físicos (um para um).

No ambiente OpenStack como foi descrito nos parágrafos anteriores, o objetivo foi provisionar instâncias que mais se assemelham com o ambiente Nativo. Porém, é importante mencionar que por se tratar de máquinas virtuais, é preciso um arranjo, o que é totalmente natural em ambientes de virtualização. Criação de discos, redes, interfaces de rede e até processadores virtuais, o que acredita-se que possam ter um desempenho inferior ao um ambiente Nativo. Neste caso, para criação dos discos, foi realizado a configuração de um componente para suporte à discos LVM para as instâncias. Para rede, criação das interfaces, acontece por um componente denominado OpenVSwitch, e para redes e roteadores o componente Neutron.

No ambiente OpenNebula, segue um contexto semelhante ao OpenStack. Porém, seu modo de implantação se difere em alguns quesitos, como na criação dos discos para as instâncias, igualmente é feito através do suporte à LVM, porém eles

são alocados em uma unidade compartilhada entre os nodos, através do NFS. Para interfaces de rede das máquinas virtuais, utilizou-se os recursos padrões da ferramenta. E ainda para processadores e memória RAM, ficou a cargo dos serviços do KVM.

Vale lembrar que em todos os ambientes a rede é 100 Mbits/s, se tornando assim um meio mais ultrapassado para comunicação de rede em execuções de aplicações paralelas e distribuídas. Buscou-se seguir apenas atributos para tornar as duas ferramentas operantes diante do cenário proposto, mas que mantessem a maior homogeneidade possível, não foi aprofundado nesta etapa outros recursos que não foram citados neste trabalho em ambas as ferramentas.

3.2.6 **Configuração dos testes**

Uma das etapas principais durante este trabalho é a configuração dos testes, pois os resultados destes, serão decisivos na execução deste trabalho. As hipóteses levantadas no projeto, caminham em busca da comparação de desempenho de ambientes virtuais de infraestruturas de computação em nuvem utilizando estações de trabalho, e também a execução de aplicações de alto desempenho.

Com o estudo da literatura e dos trabalhos relacionados, pode-se elencar alguns dos principais *benchmarks* aplicados na área de análise de desempenho de ambientes virtuais e ambientes Nativos, e *benchmarks* para execuções de aplicações paralelas. Para a escolha dos *benchmarks* foi levado em consideração também a compatibilidade de execução em todos os ambientes usados na pesquisa.

Antes de efetivar a configuração dos *benchmarks*, foi necessário a configuração de uma ferramenta de monitoramento de recursos, o Ganglia. Essa ferramenta como mostra o estudo no capítulo 2 deste trabalho, foi especialmente desenvolvida para monitoramento de *clusters*, no qual monitora e exibe em forma de gráficos a utilização dos recursos físicos de cada node. Isso se fez necessário para que durante a conclusão dos testes, fosse possível analisar eventuais anomalias no decorrer das execuções referente ao uso dos recursos físicos.

Para configuração do Ganglia, é preciso a configuração de um *frontend* com o serviço Apache como servidor HTTP, e posteriormente a configuração dos serviços referentes ao Ganglia. Nos nodes, somente deve ser instalado e configurado o serviço de envio para o servidor.

Com o sistema de monitoramento em todas as infraestruturas devidamente configurado, partiu-se para a preparação dos testes com os *benchmarks* de aplicações paralelas. Para os testes foi escolhido o NPB-3.3 (*NAS Parallel Benchmark*) devido à larga utilização desta suíte na avaliação de desempenho de aplicações paralelas, como pode ser observado no estudo dos trabalhos relacionados.

O NPB-3.3 possui seus *kernels* de *benchmarks* implementados em padrões MPI (*Message Passing Interface)* para execuções de aplicações paralelas distribuída entre nodos de uma estrutura, e OMP (*Open Multi-Processing*), no qual executa aplicações em modalidade multi-processadas. Assim, MPI executa seus testes em forma distribuída, escalonando cargas de trabalho entre os nodos do ambiente. E OMP distribuí suas cargas entre os núcleos e *threads* do processador do node ou instância.

Por se tratar de equipamentos idênticos, pode-se ter uma flexibilidade na execução dos testes da NBP. Para execução da suíte NPB usando o padrão MPI, pode-se usar um dos *clusters* (4 nodes) para a execução dos testes no modo distribuído. E para execução da suíte NPB usando o padrão OMP, pode-se usar um único nodo divergente do *cluster* escolhido para os testes com o MPI. Deste modo, devido ao padrão MPI usar constantemente o tráfego de rede, e o NPB-OMP usar somente processamento local, em nenhum momento os testes sofreram interferência no processamento dos mesmos. Esse padrão foi usando nos ambientes Nativo e virtual, e nas ferramentas OpenStack e OpenNebula.

Com a definição da estrutura dos testes, a suíte NPB OMP e MPI foram preparadas usando a classe B. Chegou-se a esta definição devido aos testes realizados com as outras classes, onde algumas praticamente não exerciam nenhum estresse significativo no ambiente, e outras não foram usadas devido a grade carga que aplicavam no momento de sua execução. Outro fator importante na escolha desta

classe, é que em algumas classes poderiam ocorrer que alguns programas poderiam ficar de fora devido a incompatibilidade imposta pelo próprio conjunto.

Portanto, a suíte NPB ficou da seguinte forma: Através da classe B, que se mostrou em um grau mais compatível com a infraestrutura usada na pesquisa, e se manteve com maior similaridade entre os *benchmarks* dos padrões MPI e OMP, os programas usados para as execuções foram: *BT, CG, EP, FT, IS, LU MG* e *SP,* implementados em padrão MPI. E os programas: *BT, CG, EP, FT, IS, LU, MG, SP*, *UA* implementados no padrão OMP.

A execução de todos os programas da suíte NPB, em ambos os padrões, tiveram uma quantidade de 40 repetições em cada ambiente, seguindo uma ordem aleatório mas continua. Ou seja, em nenhum momento executou-se as 40 repetições de uma só vez de determinado programa, foi seguindo uma ordem um após o outro até todos terem completado as 40 execuções. Isso para não haver viés devido ao modo que cada programa da suíte opera.

É importante destacar ainda que nos testes com o padrão MPI, a forma que é aplicado as cargas de trabalho é pela criação de processos que cada programa executa. E a forma que será distribuído as cargas de trabalho, é consequência da quantidade destes processos criados. Portanto, durante as execuções, além de manter uma ordem aleatória e continua de cada programa, tem-se um aumento programado de processos nos programas da suíte.

Para melhor compreender esse método, os *benchmarks* escolhidos pela quantidade de processos, sendo do padrão MPI foram: *SP, BT, CG, EP, FT, IS* e *LU*, por implementarem 1 (um) processo no momento da execução. Já os programas que implementam seus testes ao executar 2 (dois) processos foram: *CG, EP, FT, IS, LU, MG*. Os que implementam 4 (quatro) processos foram: *SP, BT, CG, EP, FT, IS, LU, MG*. Com 8 (oito) processos foram: *CG, EP, FT, IS, LU, MG*. Com 9 (nove) processos: *SP, BT, EP*. E os que são implementados com 16 processos são: *BT, CG, FT, IS, LU, MG, SP, EP.*

Para fundamentar a escolha de todos os *benchmarks* citados no parágrafo anterior, o Quadro 4, representa a suíte NPB compilado usando a classe B para o ambiente usado na pesquisa, aonde é possível ver todos os programas disponíveis para a execução e a quantidades de processos disponíveis durante a execução. No quadro, o símbolo “X” representa que o programa está disponível para a execução na quantidade descriminada de processos.

NPB-MPI Classe B																
	Quantidade de processos															
Programas	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
BT	X			X					X							X
CG	X	X		X				X								X
EP	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
FT	X			X				X								X
IS	X	X		X				X								X
LU	X	X		X				X								X
MG	X	X		X				X								X
SP	X			X					X							X
DT																

Fonte: Maron, Griebler. 2014.

Quadro 3: Programas e quantidades de processos da suíte NPB-MPI compilado na Classe B.

No Quadro 3, percebe-se que a programa EP é o que está disponível no maior número de processos. Porém entre o 10 e o 15, os outros programas não estão disponíveis. Então optou-se em não usar o EP nessa faixa de processos.

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+  COMMAND
16132 root      20   0  196m   32m  3980 R   100   0.9   1:16.12 sp.B.16
16134 root      20   0  196m   31m  3596 R   100   0.9   1:16.12 sp.B.16
16135 root      20   0  196m   31m  3600 R    99   0.9   1:16.01 sp.B.16
16133 root      20   0  196m   31m  3604 R    99   0.9   1:15.96 sp.B.16
```

Fonte: Maron, Griebler. 2014.

Figura 29: Demonstração do *benchmark* NPB-MPI em execução.

A Figura 29 demonstra a execução do programa SP em 16 processos, mas sendo executado em cada *thread* de um único node. Este exemplo se repete no restante dos nodes (3 nodes) da infraestrutura. Resumindo, em cada node da infraestrutura tem 4 processos do programa SP, totalizando em 16 processos no cluster, utilizando a carga total dos processadores.

Para usar o NPB-OMP, a suíte foi compilada para trabalhar com a classe B. Neste caso, a escolha se manteve devido aos mesmos fatores que levaram a escolha do padrão MPI. Devido as intensidades de cargas de trabalho geradas por cada classe, e a compatibilidade com o ambiente usado na pesquisa.

A suíte NPB-OMP possui os mesmos programas que a suíte NPB-MPI, porém sua execução não é distribuída, e sim paralelizada entre os núcleos e *threads* do processador. Devido ao padrão da classe B no NPB-OMP um programa adicional pode ser usado, os programas usados na lista do NPB-OMP ficaram da seguinte forma: *BT, CG, DC, EP, FT, IS, LU, MG, SP,* e *UA.* Diferenciando do padrão MPI, este teve o programa *UA* a mais durante as execuções.

Para a execução de todos os programas da suíte do padrão OMP, foram realizadas 40 repetições, aleatórias mas seguindo uma sequência única. Ou seja, os programas eram executados respeitando uma fila, o mesmo método usado na suíte MPI, para que deste modo não haja viés ou resultados que pudessem sofrer depreciações durante as execuções. Além de seguir esta ordem de execução, todos os programas também seguiram o aumento gradativo das *threads* de execuções. Isto significa que após ter sido iniciado, ao momento em que todos os programas concluírem um ciclo de 40 execuções, um laço de repetição no *script* dos testes, incrementa a variável que define a quantidade de *threads* que o programa irá usar. Seguindo este ciclo até que seja concluída as 40 execuções utilizando as 4 *threads* do processador. O valor destas *threads* foi definido devido ao modelo de processador possuir essa quantidade em sua arquitetura física.

Tanto as execuções do NPB-MPI e NPB-OMP, foram automatizadas através de *shell scripts*. Todo os esquemas de repetições, chamadas dos programas, incremento de variáveis usadas pelos programas e, a coleta de resultados, foram cuidadosamente planejados e escritos em dois arquivos contendo os códigos das execuções dedicados a cada uma das suítes NPB. Alguns dos resultados dos programas podem ser vistos no Apêndice F. E para melhor compreender os scripts utilizados, os mesmos podem ser vistos nos Apêndices C e D deste trabalho.

Para preparação dos testes do ambiente Nativo e virtual em cada uma das ferramentas foram escolhidos *benchmarks* para avaliar o desempenho do processador, da memória RAM, da unidade de armazenamento, e desempenho de rede.

Para avaliação do processador, foi usado o *benchmark* LINPACK. Este que devido ao longo período se manteve como um *benchmark* usado para a avaliação de desempenho de processadores. Seu histórico é muito conhecido na avaliação do processamento de supercomputadores, estes nos quais podem chegar ter milhões de núcleos de processamento. (TOP500, 2014). E consequentemente em grande parte dos trabalhos relacionados utilizarem este *benchmark* para suas avaliações.

O código usado no LINPACK é escrito em linguagem C, para a execução dos testes, este *benchmark* teve que ser editado para que o valor das matrizes usada para gerar as cargas de trabalho no processador, pudessem ser exatamente as mesmas no decorrer dos testes, pois originalmente, o *script* necessitava a interação do usuário para que o valor da variável seja definido. Portanto, a variável dentro do código que identifica o valor da matriz, foi alterado, para dimensão de 4000X4000, e assim automatizar este processo. Esse valor foi definido devido alguns testes realizados com outros valores, estes nos quais levavam um longo tempo para concluírem, e ainda eram fragmentados para que o processo se concluísse.

Para avaliar a unidade de armazenamento, utilizou-se o IOzone. Este que é voltado para testes de *throughput* e desempenho da unidade, e ainda testes com sistema de arquivos. A instalação desde *benchmark* foi realizada através dos repositórios do Ubuntu 12.04, não necessitou de nenhuma configuração especial. Apenas no ambiente virtual, foi necessário configurar e acrescentar os mesmos repositórios utilizado no ambiente Nativo, pois por padrão estes não eram configurados.

No momento da execução dos testes da unidade de armazenamento, utilizou-se o modo automático do IOzone, sendo que gradativamente o tamanho do arquivo usado vai aumentando até chegar em um nível específico. Porém ao chegar em uma etapa do processo, percebeu-se que a instância virtual da ferramenta

OpenStack estava em um estado inconsciente, não respondia a nenhum comando de comunicação, e nos logs armazenado na unidade compartilhada do *cluster* virtual não havia atualizações de informações.

Então devido a este problema ocorrido no ambiente virtual, optou-se em limitar o modo dos testes implementado pelo IOzone. Através de seus parâmetros de testes, foi possível escolher quais seriam os testes aplicados ao arquivo criado por ele mesmo, e ainda qual seria o tamanho deste arquivo, no qual chegou-se a 100 MB. Ao propor este tamanho, os testes fluíram normalmente, mas a decisão da escolha não partiu de nenhum parâmetro ou processo metodológico. Entende-se que foi um tamanho considerável, isto porque, seu tempo de operação era o mais longo dentre os *benchmarks* de avaliação do ambiente, e outra, que o objetivo ainda não é definir se o ambiente OpenStack ou OpenNebula, é compatível com a utilização de banco de dados, que exigem uma quantidade maior de recursos de armazenamento.

Concluindo, o IOzone avaliou o desempenho das unidades de armazenamento, implementado testes de escrita, reescrita, leitura e releitura, em um arquivo de 100MB.

O objetivo inicial para avaliar o desempenho de rede era em utilizar o *NetPipe,* porém alguns problemas surgiram na execução dos testes nas instâncias virtuais. O *benchmark* trabalha com conexões cliente/servidor, é necessário um servidor rodando em uma porta definida, e um cliente que se conectará para iniciar o tráfego dos dados e assim medir a vazão obtida. Porém, toda vez que esta conexão se encerra, o processo servidor também é encerrado, ficando indisponível na próxima execução. Mas dentre os parâmetros do NetPipe, um deles é a opção de persistir o processo do servidor, para que aguarde receber novas conexões, após o encerramento das mesmas, mas infelizmente acabou não surtindo efeito.

Então, devido a este problema o NetPipe foi descartado nesta ocasião, e então optando pela utilização do IPERF, que pode ser facilmente instalado através dos repositórios do sistema operacional Linux, sem a necessidade de configurações específicas. Este que também faz sua implementação muito semelhante ao NetPipe, através de cliente e servidor.

Para que então seja executado de forma automatizada, um processo servidor foi iniciado em um nodo, utilizando a sintaxe padrão do *benchmark*, e apenas repassando um comando do ambiente Linux, para que este processo servidor seja executado em segundo plano, e em outro nodo, um comando era executado para que o cliente se conectasse neste servidor e assim iniciar a transferência das informações. Para a execução do cliente, foi utilizado apenas a sintaxe padrão de conexão com o processo servidor.

Sendo assim, os testes de rede seguiram todos os processos padrões do IPERF, sendo que em nenhum momento foi utilizado os parâmetros de definição de tempo de execução, tamanho do tráfego entre cliente e servidor, e entre outros. Isto tudo foi automaticamente definido pelo próprio *benchmark*, que no qual se manteve constante em todas as execuções. Então de acordo com os logs dos testes, sempre as conexões se mantiveram em execuções de 10 segundos, e durante este tempo, a maior quantidade de dados possível era transferida, onde posteriormente era calculada a vazão deste tráfego, e quanto foi transferido durante o tempo.

Para avaliar o desempenho da memória RAM, foi escolhido o *benchmark* STREAM. Este que foi escolhido devido a sua utilização nos trabalhos relacionados, abordados no início deste capítulo.

O processo de execução do STREAM, não teve nenhuma complexidade no momento de sua utilização. Para executa-lo, foi necessário baixar o código escrito em C, e após isto, compilá-lo nos ambientes em que seria utilizado. Basicamente, o STREAM não necessita de nenhum parâmetro especial de execução, porém, em cada teste, o código interno utiliza os mesmos parâmetros para avaliar o desempenho da memória RAM. Estes parâmetros são, o tamanho da matriz endereçada na memória, sendo o valor 2000000. O tamanho de memória requerida para o armazenamento desta matriz, sendo o valor de 45,8 MB. Onde cada chamada de execução do STREAM, o código interno é executado 10 vezes no sistema, e destas 10 execuções, apenas a de menor tempo seria catalogado para os resultados.

Com estes parâmetros implementado pelo código do STREAM, ele executa os testes de cópia, adição, escala e tríade dos valores na memória RAM.

Então, para avaliação do ambiente (Nativo, Virtual) foram utilizados os *benchmarks* IOzone para teste da unidade de armazenamento. LINPACK para avaliar o desempenho do processador. STREAM para teste de memória RAM, e IPERF para avaliar o desempenho da rede. Todos estes *benchmarks* seguiram um ciclo de 40 execuções, aonde foram implementadas através de um *Shell Script*, que automatizou todo este processo de execução dos *benchmarks* e coleta destes resultados. O *script* utilizado para execução dos testes encontra-se no Apêndice E. Também alguns dos arquivos de logs se encontram Apêndice F deste trabalho.

Todos os *benchmarks*, tanto para avaliar os recursos do ambiente, e também as aplicações paralelas, seguiram a mesma quantidade de execuções, sendo 40 vezes. Este valor dará uma média maior de confiabilidade aos experimentos no cálculo das médias.

Além de considerar pontos importantes durante a execução e configuração dos testes, percebe-se que a pesquisa limitou-se em utilizar apenas um *benchmark* para avaliar cada componente (processador, memória, rede, disco). Algo que traria maior legitimidade nos resultados seria a execução de mais *benchmarks* com o mesmo fim, para o comportamento dos ambientes pudessem ser testados ao máximo. Além disso, é possível observar os testes se limitaram inicialmente a avaliar seus recursos mais básicos. Como em casos das execuções do LINPACK, que aborda outros resultados como DGEFA, DGESL e ainda *overhead.*

Apesar das suítes NPB serem *benchmarks* de conceito no que diz respeito a execução de aplicações paralelas e distribuídas, isto também deveria seguir o mesmo contexto, pois a pesquisa também limitou-se em utilizar a suíte NPB.

3.3 ANÁLISE DOS RESULTADOS

Nesta seção, serão apresentados os resultados das avaliações obtidas no ambiente Nativo, OpenStack e OpenNebula, através das execuções de *benchmarks* de avaliação destes ambientes, e também o resultado das aplicações paralelas. Primeiramente serão abordados uma análise dos resultados da infraestrutura do

ambiente Nativo, virtual no OpenStack e virtual no OpenNebula, e seguindo com uma análise comparativa. No mesmo contexto para as aplicações paralelas.

Para todos os resultados das execuções foram criados gráficos para comparação de resultados com auxílio da ferramenta Gnuplot. Uma ferramenta *open source* para criar gráficos em ambiente Linux.

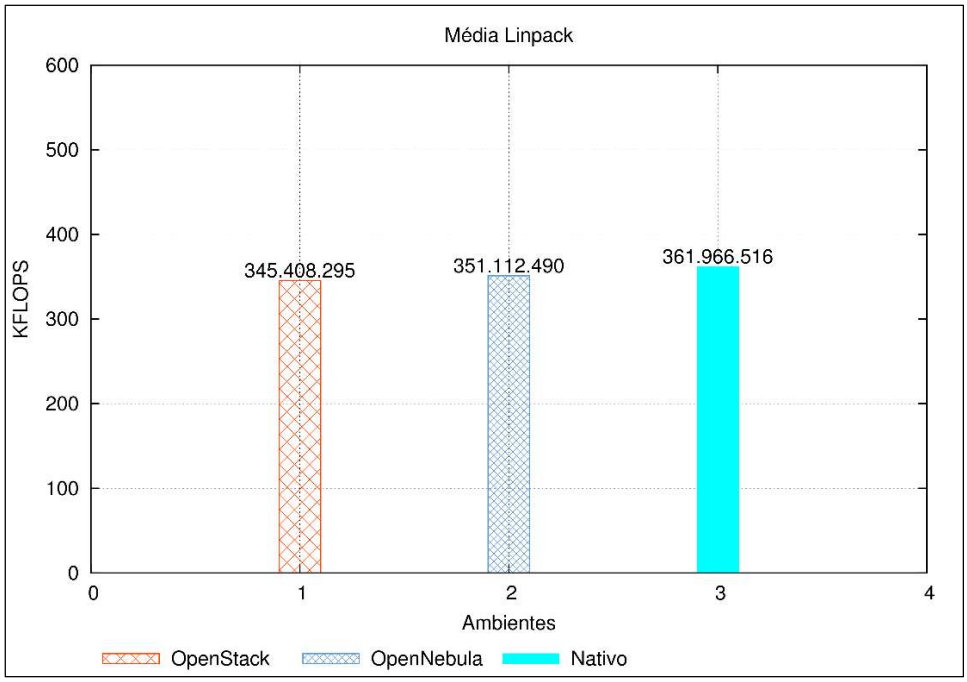
3.3.1 **Infraestrutura**

Inicialmente optou-se em avaliar o desempenho dos recursos computacionais em cada um dos ambientes (Nativo, virtual OpenNebula, e virtual OpenStack). Em cada uma das execuções, seguiu-se as descrições abordadas na seção anterior, aonde aborda a preparação do ambiente para cada um dos testes de avaliação de memória RAM, armazenamento, rede e processador.

Os resultados do ambiente Nativo foram importantes para servir como comparativo entre o restante dos ambientes da pesquisa. E por se tratar de um ambiente com não possui camadas adicionais para acesso aos componentes físicos, esperara-se que o ambiente Nativo tenha um nível maior de desempenho nos testes.

3.3.1.1 Análise Comparativa

A análise comparativa inicia-se com um gráfico da média de execuções do *benchmark* LINPACK. A Figura 30 representa as médias obtidas nas execuções do ambiente OpenStack, OpenNebula e Nativo. Os valores são representados em KFLPOS na vertical e cada um dos ambientes sendo representados na horizontal.



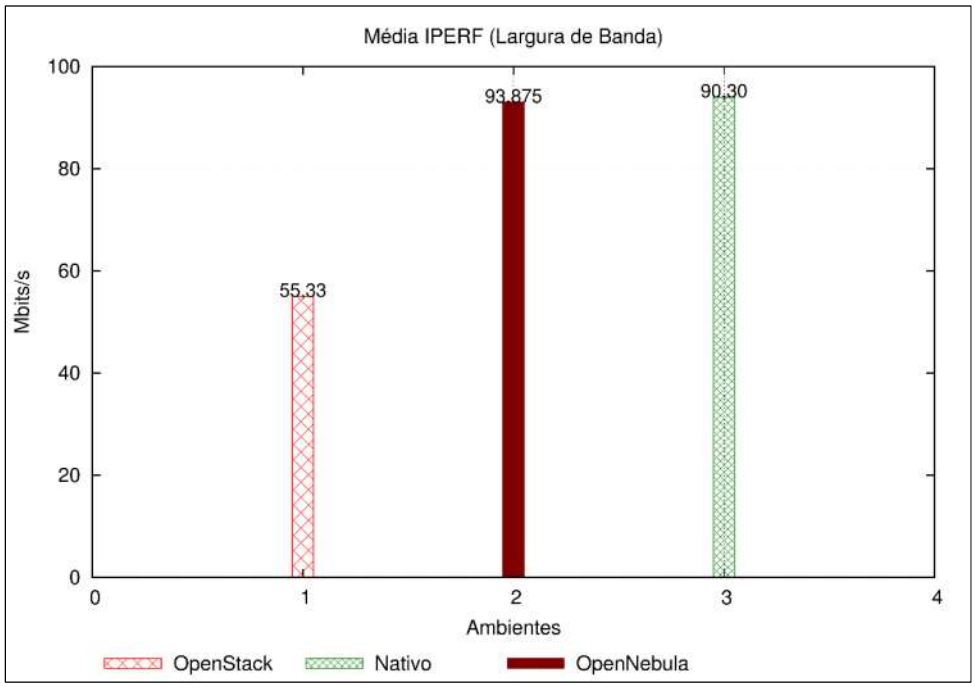
Fonte: Maron, Griebler. 2014.

Figura 30: Média dos resultados do LINPACK

Como era o esperado, o ambiente Nativo alcançou um melhor resultado em comparação aos demais. A instância na ferramenta OpenNebula conseguir o melhor resultado em comparação com a ferramenta OpenStack. Porém analisando o gráfico com seus valores, em cada um dos ambientes, os resultados se mantiveram muito próximos.

Algo importante de que pode ser levado em conta nestes resultados de desempenho do processador, na ferramenta OpenStack no momento da criação de uma instância, apenas são definidos a quantidade de vCPUs que cada instância poderá utilizar para suas execuções. E na ferramenta OpenNebula, a descrição é trazida com as opções de escolha da quantidade de CPUs e também a quantidade de vCPUs. Mesmo sabendo que o ambiente Nativo não continha 4 CPUs, optou-se em definir este mesmo valor na configuração da instância, e ainda 4 vCPUs, pois não se conseguiu encontrar na literatura uma explicação para estas características no momento da criação da instância, mas entende-se que as vCPUs representam a quantidade de núcleos para uma instância. Portanto, acredita-se que este pode ser o fator que tenha possibilitado ao ambiente OpenNebula ser melhor que o OpenStack.

A Figura 31, representa a média obtida dos resultados com o *benchmark* IPERF. Na vertical são apresentados a média dos valores obtidos da largura de banda dos testes, sendo representado na unidade Mbits/s, e na horizontal, são apresentados cada um dos ambientes.



Fonte: Maron, Griebler. 2014.

Figura 31: Média dos resultados do IPERF

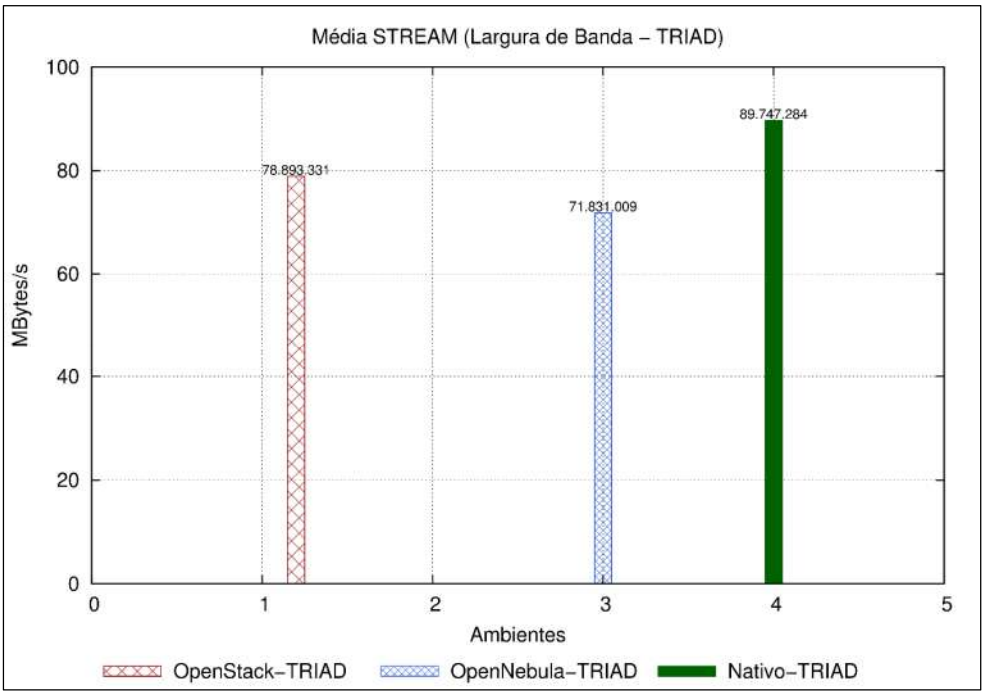
Os resultados dos testes de redes surpreenderam devido ao péssimo desempenho obtido na ferramenta OpenStack, e ao bom desempenho obtido na ferramenta OpenNebula.

Acredita-se que o bom desempenho obtido pelo OpenNebula, foi devido a placa de rede destinada aos nodos do *cluster* ser *gigabit,* mesmo que o *switch* da rede não forneça este tipo conexão. Outro fator que pode justificar o bom desempenho da rede, é como ela é implementa na ferramenta OpenNebula, em suas configurações foram usadas a forma padrão de implementação, a rede *Dummy* como é descrito na ferramenta. Aonde as redes e as interfaces das instâncias são implementadas através de um driver básico de rede, utilizado pela ferramenta. Diferentemente da OpenStack que implementa suas interfaces de redes através do componente OpenVSwitch, e possui ainda outro componente para gerenciamento de redes e roteadores virtuais.

Nos testes de memória RAM, cada um dos resultados de Cópia (*Copy),* Escala *(SCALE),* Adição *(ADD)* e de Tríade *(TRIAD)* foram representados em gráficos separados, devido ao grande volume de informações.

Em todos os gráficos dos resultados dos testes de memória, na vertical é representado em Mbytes/s o valor alcançado na largura de banda de cada ambiente, e na horizontal, é representado cada um dos ambientes.

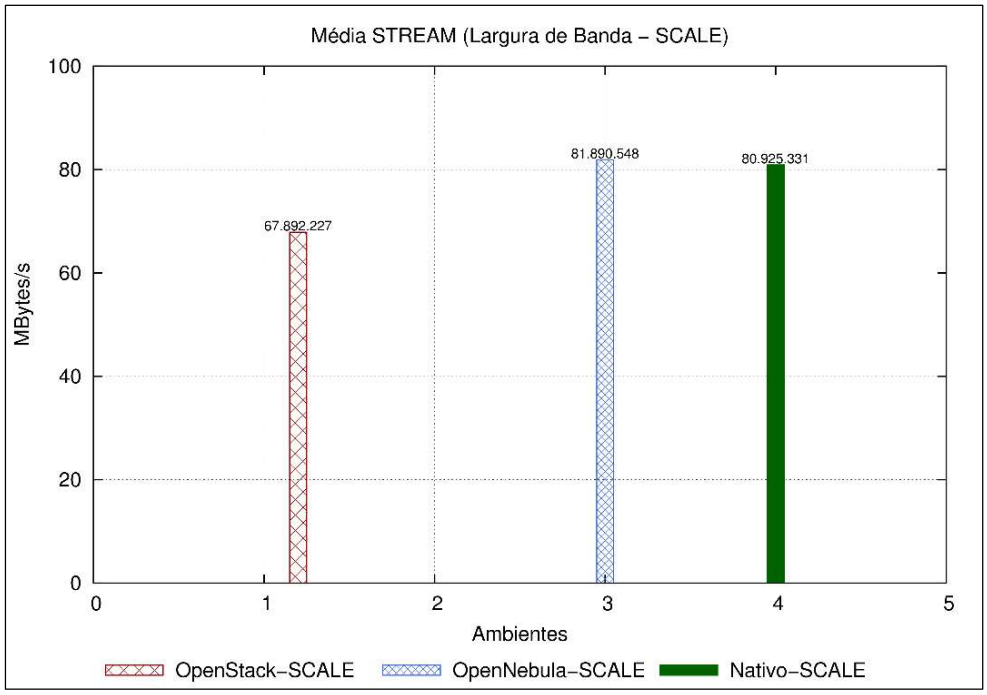
A Figura 32 representa os valores dos resultados da execução de tríade do STREAM, aonde é possível perceber que os valores do ambiente Nativo alcançaram um valor relativamente maior que os outros ambientes. Mas as ferramentas de computação em nuvem ficaram muito próximas em seus resultados.



Fonte: Maron, Griebler. 2014.

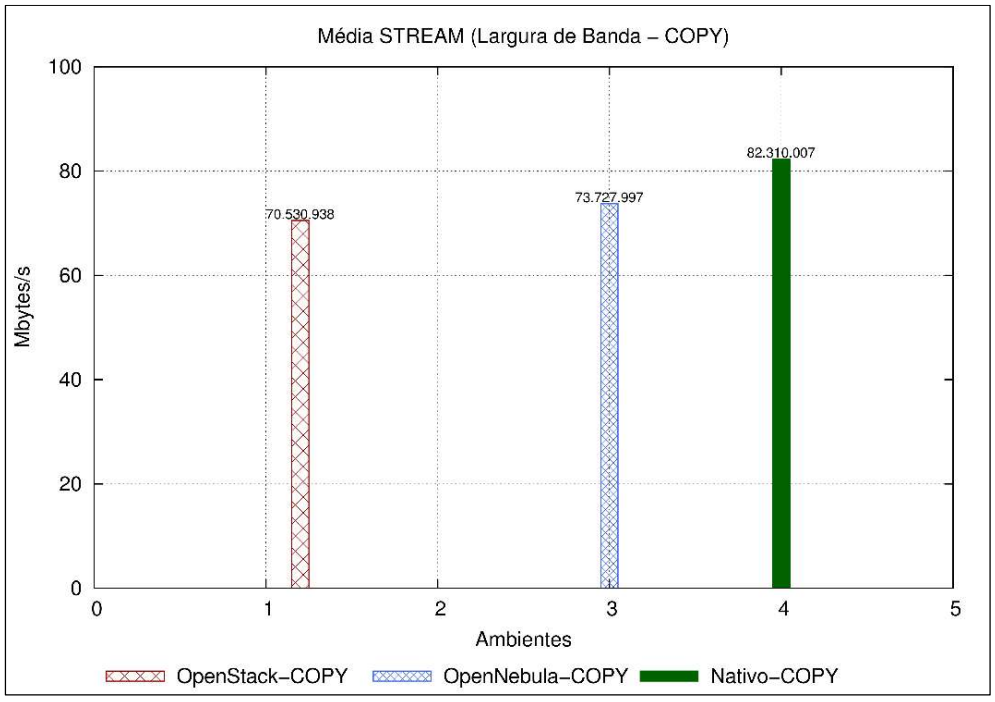
Figura 32: Largura de banda memória RAM (TRIAD)

Para os testes de escala os resultados podem ser vistos na Figura 33. Nestes resultados a ferramenta OpenNebula alcançou uma posição próxima ao ambiente Nativo. E a ferramenta OpenStack teve um desempenho muito inferior com relação ao restante dos ambientes.



Fonte: Maron, Griebler. 2014.

Figura 33: Largura de banda memória RAM (SCALE)



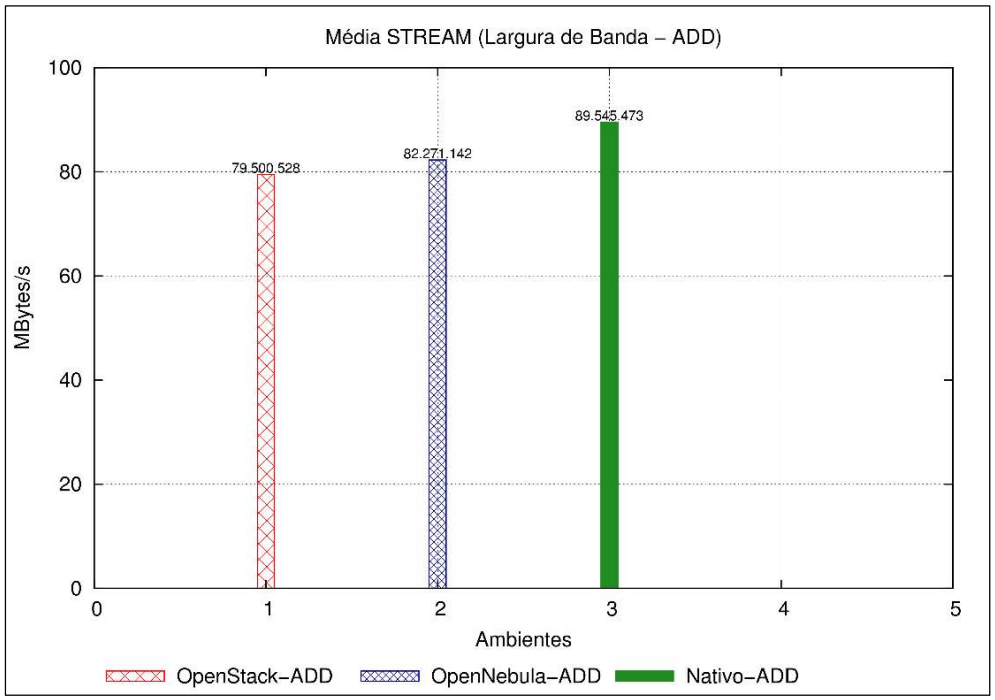
Fonte: Maron, Griebler. 2014.

Figura 34: Largura de banda memória RAM (COPY)

No resultado dos testes de Cópia com o STREAM, a Figura 34 apresenta a média dos valores obtido nas execuções. É evidente que o ambiente Nativo

alcançou o melhor resultado, porém, dentre os outros ambientes, a ferramenta OpenNebula obteve o melhor resultado com relação ao OpenStack.

A Figura 35 apresenta os resultados com os testes de adição executados pelo STREAM. Na figura é possível perceber a superioridade do ambiente Nativo em relação às outras ferramentas. Mas dentre elas, a OpenNebula teve um desempenho melhor que a OpenStack.



Fonte: Maron, Griebler. 2014.

Figura 35: Largura de banda memória RAM (ADD)

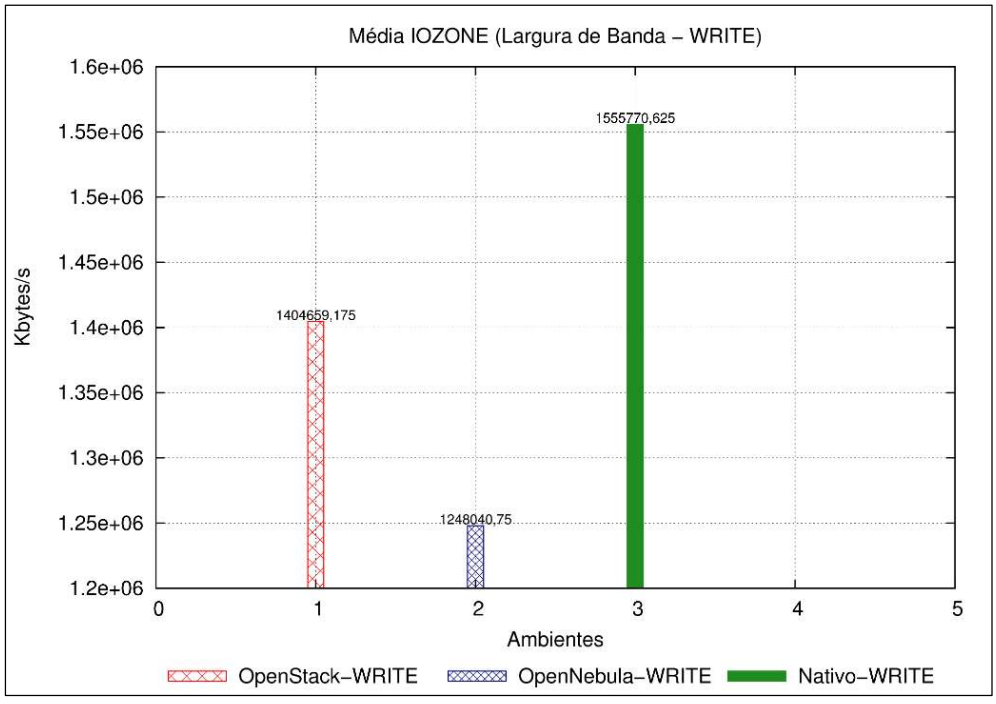
Portanto, em todas as execuções realizadas pelo STREAM, os resultados comparados entre as ferramentas de administração de nuvem, a OpenNebula alcançou os melhores resultados de desempenho. Aonde somente nos testes de tríade é que a OpenStack obteve superioridade. Ainda dentre todos os resultados, se mantiveram próximos ao ambiente Nativo, com exceção ao teste de escala que o OpenStack alcançou um resultado baixo, considerando os outros resultados que mantiveram mais próximos de seus superiores.

Mesmo se tratando dos mesmos virtualizadores, a ferramenta OpenStack possui componentes que estão constantemente monitorando informações das

instâncias. Como é o caso do componente Nova, onde através das funções que este componente desempenha, pode ocorrer de ter degradado o desempenho do ambiente OpenStack.

Nos resultados dos testes de desempenho da unidade de armazenamento, utilizou-se o IOzone para medir operação de escrita, reescrita, leitura e releitura. Cada uma destas operações fora colocada em gráficos separados, devido ao volume de informações, mas contrapondo cada ambiente. Nos gráficos, a coluna vertical representa o valor alcançado nas operações, este valor é apresentado em Kbytes/s, e na horizontal são apresentados os ambientes analisados.

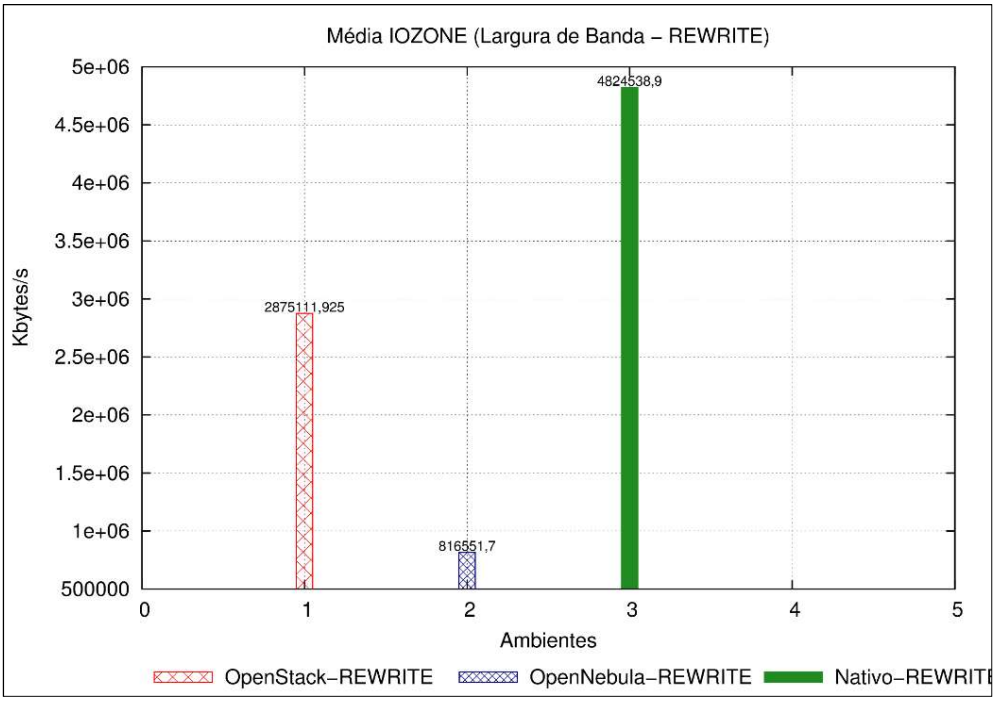
A Figura 36 representa os resultados obtidos com o teste de escrita (*WRITE*) com o IOzone, este representa a criação de um novo arquivo no disco. Em análise com os outros ambientes, já era esperado que o ambiente Nativo superasse nos resultados. É possível perceber que dentre as ferramentas de administração de nuvem, a OpenStack alcançou uma média melhor em comparação com o OpenNebula, mas também muito atrás do ambiente Nativo.



Fonte: Maron, Griebler. 2014.

Figura 36: Média das operações com o IOzone (WRITE).

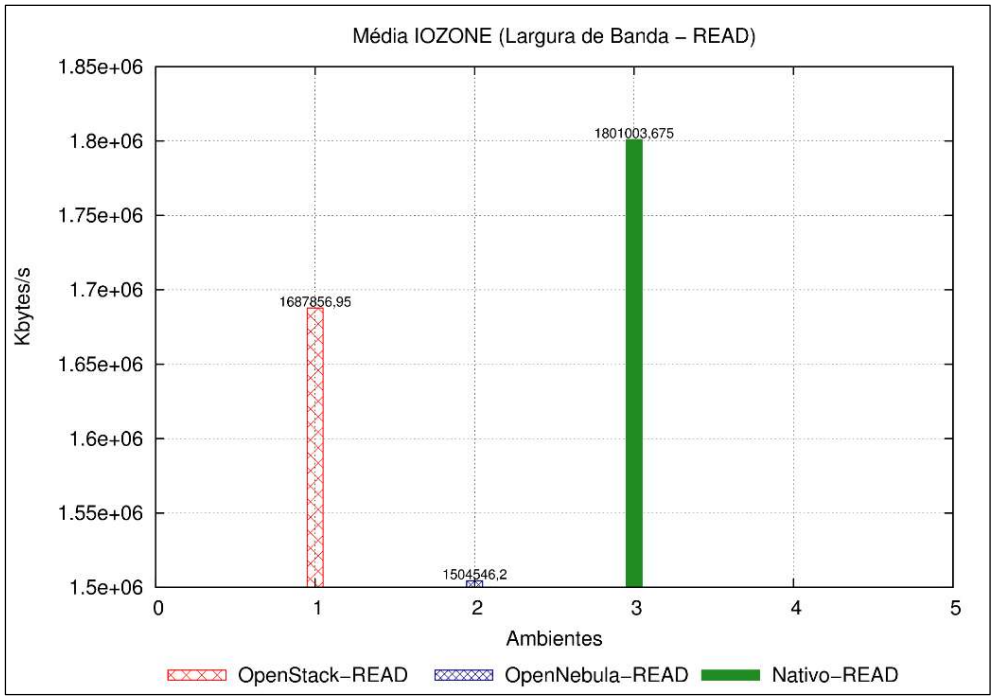
A Figura 37 representa os testes com a função reescrita (*REWRITE*), aonde o IOzone avalia o desempenho desta função em um arquivo já criado no disco. Tendo o ambiente Nativo como referência, o OpenStack obteve um desempenho considerável em relação a outra ferramenta. Mas analisando os valores, o resultado do OpenStack fica muito longe em relação ao ambiente Nativo. Os ambientes Nativo e OpenStack tiveram um ganho de desempenho com relação ao teste anterior de escrita, porém, a ferramenta OpenNebula teve seu desempenho reduzido em uma função que se espera ganhar desempenho devido ao implemento de caches realizados pelas unidades de armazenamento.



Fonte: Maron, Griebler. 2014.

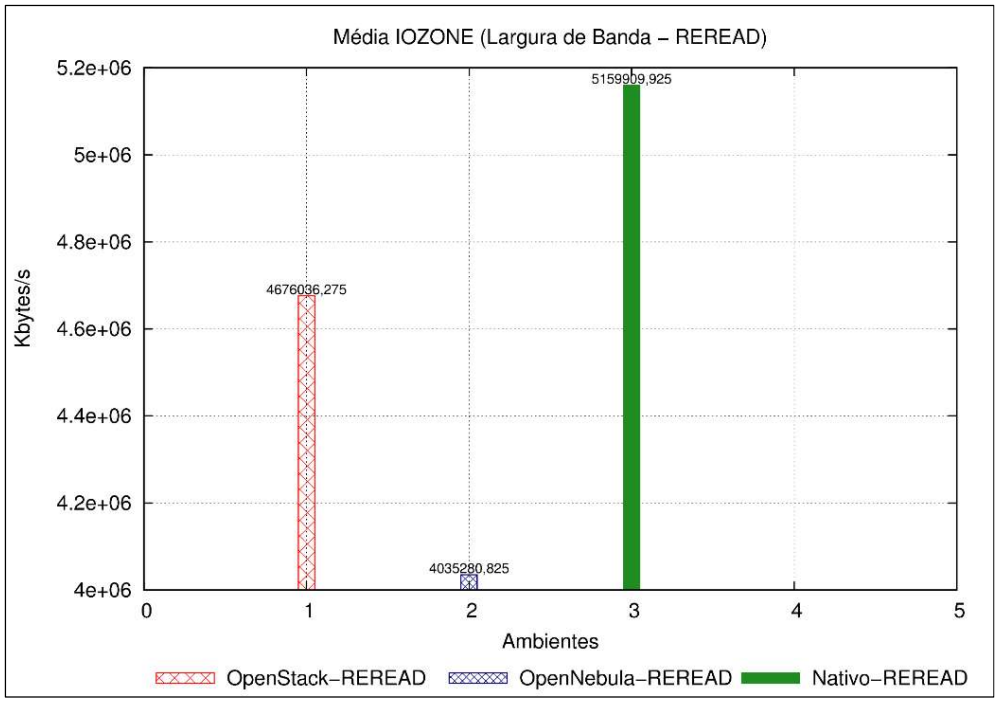
Figura 37: Média das operações com o IOzone (REWRITE).

A Figura 38, representa os resultados de leitura (*READ*) em um arquivo no disco, em relação aos resultados, a ferramenta OpenNebula drasticamente teve seu desempenho muito inferior, e a ferramenta OpenStack ainda se manteve distante do resultado do ambiente Nativo.



Fonte: Maron, Griebler. 2014.

Figura 38: Média das operações com o IOzone (READ)



Fonte: Maron, Griebler. 2014.

Figura 39: Média das operações com o IOzone (REREAD)

Os resultados da Figura 39 representa os testes de releitura (*REREAD*) em um arquivo que foi recentemente lido pelo sistema. De acordo com a ferramenta, este teste é realizado devido alguns sistemas utilizar um recurso de cache de leitura que pode representar um ganho de desempenho nas operações de disco. O ambiente Nativo e OpenStack obtiveram um ganho de desempenho com relação aos testes de leitura (*READ*), mostrando que o recurso de cache possibilita este ganho. E nesta função, é possível ver um leve ganho de desempenho da ferramenta OpenNebula na função de releitura.

Nas funções básicas de disco, o ambiente Nativo sempre acompanhou um grau de elevação no desempenho nas funções de reescrita (*REWRITE*) e releitura (*REREAD*), seguindo o mesmo contexto, onde a ferramenta OpenStack se manteve longe dos resultados do ambiente Nativo, mas acompanhou esse ganho de desempenho nas funções de reescrita (*REWRITE)* e releitura (*REREAD)*.

Na forma como a ferramenta OpenNebula foi implementada no ambiente, seu desempenho alcançou níveis inferiores em relação à ferramenta OpenStack. Aonde é possível perceber um pequeno ganho no desempenho somente na função de releitura de um arquivo (*REREAD*).

A maneira que as unidades de armazenamento são criadas nas ferramentas OpenStack e OpenNebula, são através de discos LVM, porém acredita-se que o resultado do desempenho das funções de disco da ferramenta OpenNebula tenha sido inferior, devido à forma que foi implementado o sistema, através de NFS (sistema de arquivos distribuídos), onde o principal fator da perda no desempenho foi devido ao gargalo na rede. Pois em cada nodo do cluster físico, existe um diretório compartilhado pelo *front end*, onde a ferramenta permite que as máquinas virtuais sejam criadas no restante da infraestrutura. Dessa forma o acesso a rede em operações de disco é muito maior que nas operações aonde o disco é acessado localmente.

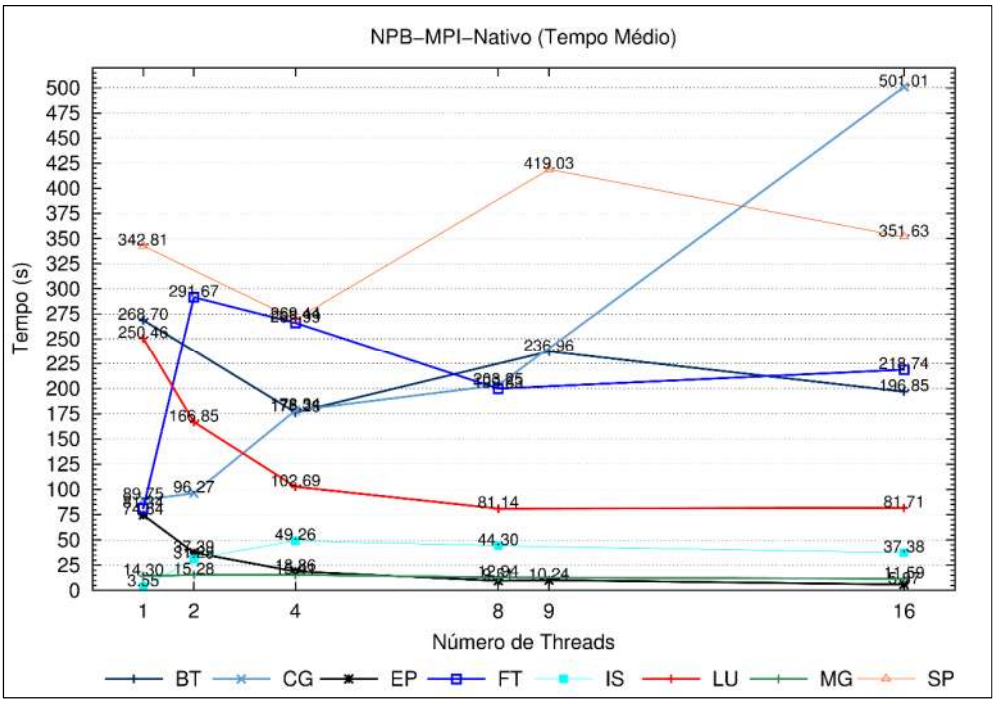
3.3.2 **Aplicações Paralelas**

As aplicações paralelas são usadas para um propósito de unir esforços em processamento para chegar a um resultado único e em menor tempo possível. Nestes testes, é importante considerar o tempo de execução das cargas de trabalho, a eficiência durante o processamento, e o ganho de desempenho durante os testes (*speed-up*).

Nas seções seguintes serão apresentados os resultados das execuções de aplicações paralelas nos ambientes Nativo, OpenStack e OpenNebula. Todas os resultados estão sendo apresentados em forma de gráficos, seguindo por uma análise de cada resultado.

3.3.2.1 Nativo

As execuções no ambiente Nativo, serviram como referência para os testes no ambiente virtual das ferramentas OpenStack e OpenNebula. A Figura 40 mostra o tempo médio das execuções da suíte NBP-MPI no ambiente Nativo.



Fonte: Maron, Griebler. 2014.

Figura 40: Tempo médio das execuções NPB-MPI no ambiente Nativo.

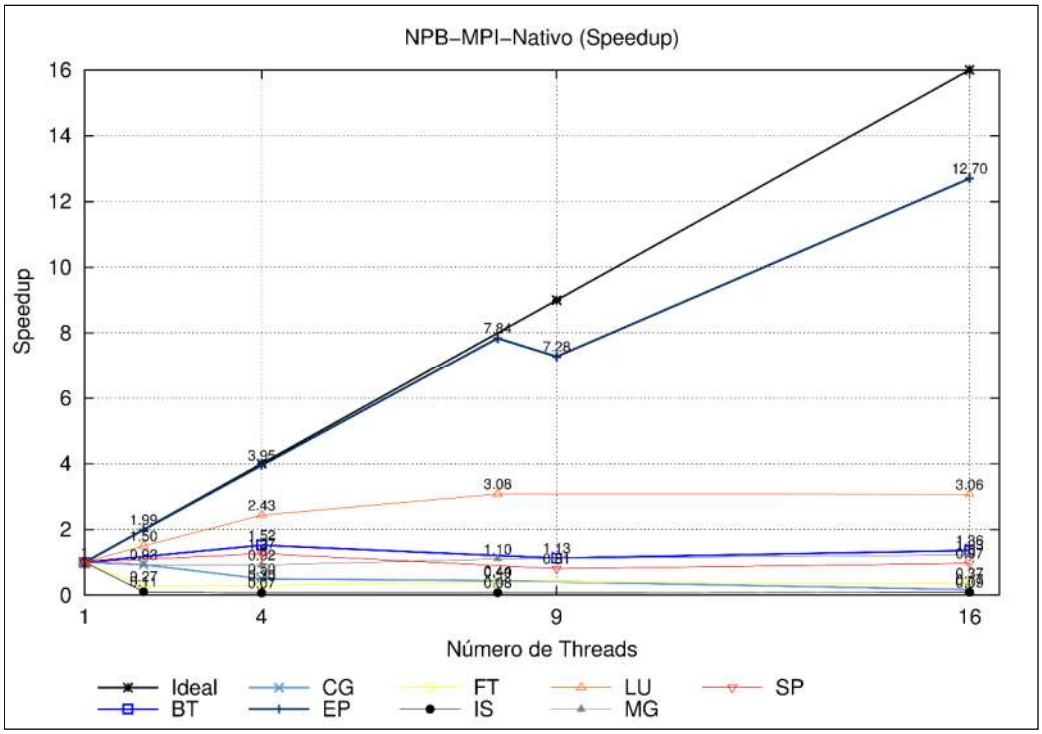
Com os resultados, observa-se que o programa CG com o aumento da quantidade de processos, também tem um expressivo aumento de seu tempo de execução. Mas é com 16 processos em que o valor tem seu maior pico no gráfico. O CG, no momento de seus testes, necessita de constante acesso à rede, portanto, acredita-se que este seja o principal motivo que seu tempo de execução tenha sido elevado.

O programa FT tem o maior tempo de execução com 2 processos dentre os outros programas da suíte NPB. Mas posteriormente, nota-se um declínio no tempo quando o a quantidade de processos se eleva. O FT é programa que faz um intensivo uso do processador, aonde com 2 processos o processador não tem um bom desempenho. Mas era esperado que com 16 processos seu tempo diminuísse, o que se provou ao contrário, onde o tempo de execução acabou aumentando em comparação com a execução de dois processos.

O LU tem seu tempo de execução com 1 processo elevado, porém com o aumento de 2, 4 e 8 é possível perceber um ganho no tempo de execução, mas com 16 processos, tem um leve aumento. O ganho de performance pode ser explicado, pois o programa utiliza a memória RAM para a resolução de matrizes, então com a ampliação de processos, tem-se um aumento de acesso a memória RAM.

As execuções do programa IS com 1 processo tem um tempo pequeno de execução, mas ao passar para 2, 4, 8 e 16, é possível ver seu tempo de execução aumentar e se manter linear.

E o programa SP até 4 processos mantem um ganho no tempo de execução, mas ao passar para 9, drasticamente seu tempo é aumentando, onde após passar para 16 processos, tem uma leve queda no tempo de execução.



Fonte: Maron, Griebler. 2014.

Figura 41: Gráfico speed-up NPB-MPI Nativo.

A Figura 41 mostra o ganho de performance com o aumento de processos em cada programa da suíte NPB. No gráfico existe a linha do *speed-up* ideal, para assimilar os resultados obtidos pela suíte de programas. Em evidência, é possível perceber apenas que o programa EP acompanha a linha ideal do *speed-up*. Pois de acordo com os resultados de tempo de execução mostrados na Figura 40, que seu tempo de execução foi gradativamente diminuindo, mostrando que quanto maior o número de processos maior seu aproveitamento de recursos e consequentemente seu tempo de execução.

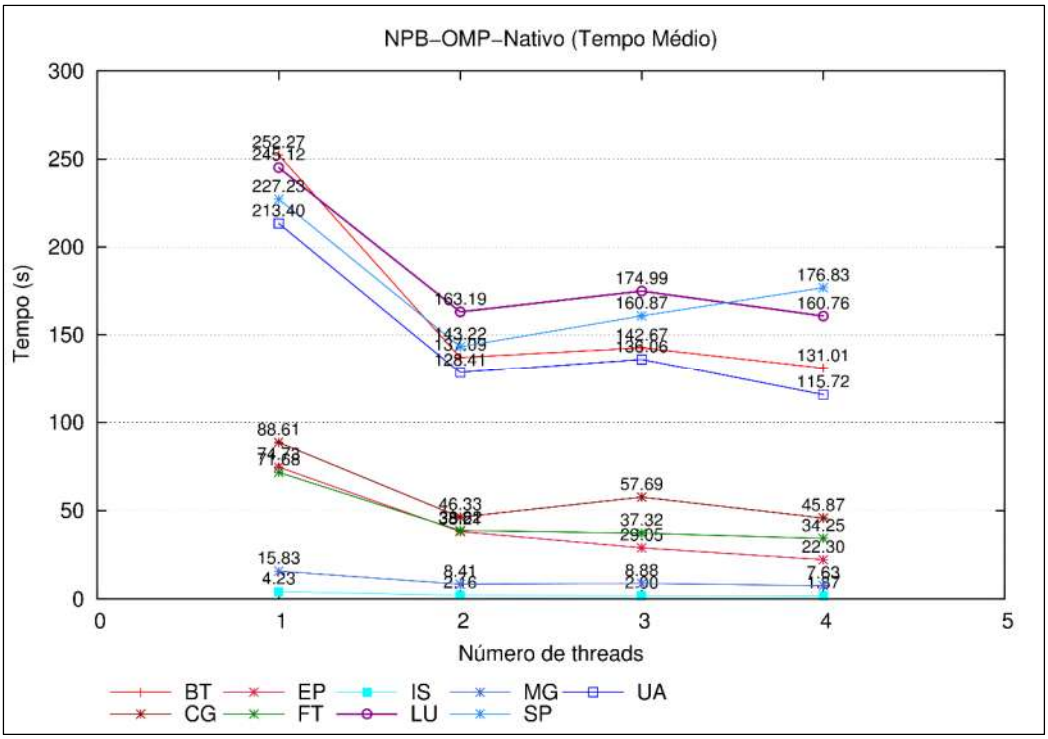
Aonde se esperava que os programas teriam uma curva crescente, nota-se que os programas FT, IS, CG, iniciam sua curva de *speed-up* menor, onde alguns se mantém em uma linha quase horizontal, mostrando níveis muito baixos de ganhos de desempenho.

Mesmo se tratando do ambiente Nativo, os gráficos das Figuras 40 e 41, demonstram que algumas execuções não obtiveram um bom desempenho. Ainda que se trata de um ambiente com uma capacidade computacional reduzida, acredita-se

que o principal problema de alguns programas não manterem uma curva de desempenho aceitável, acabou sendo a rede comunicação. As aplicações que utilizam o padrão MPI, necessitam que o tráfego de rede seja adequado para as execuções, onde no caso de infraestrutura utilizada na pesquisa, eram conexões Megabits.

Com as execuções da suíte NPB-OMP, a utilização da computação paralela se restringe apenas aos recursos locais do servidor, não necessitando em nenhum momento acesso a recursos de rede, como ocorre com o padrão MPI. Portanto, OMP implementa suas execuções com o aumento da quantidade de *threads* disponíveis para a execução dos testes.

A Figura 42 mostra o tempo médio de execuções dos programas da suíte NPB-OMP, aonde na coluna vertical são apresentados os valores de tempo, e na horizontal a quantidade de *threads*.



Fonte: Maron, Griebler. 2014

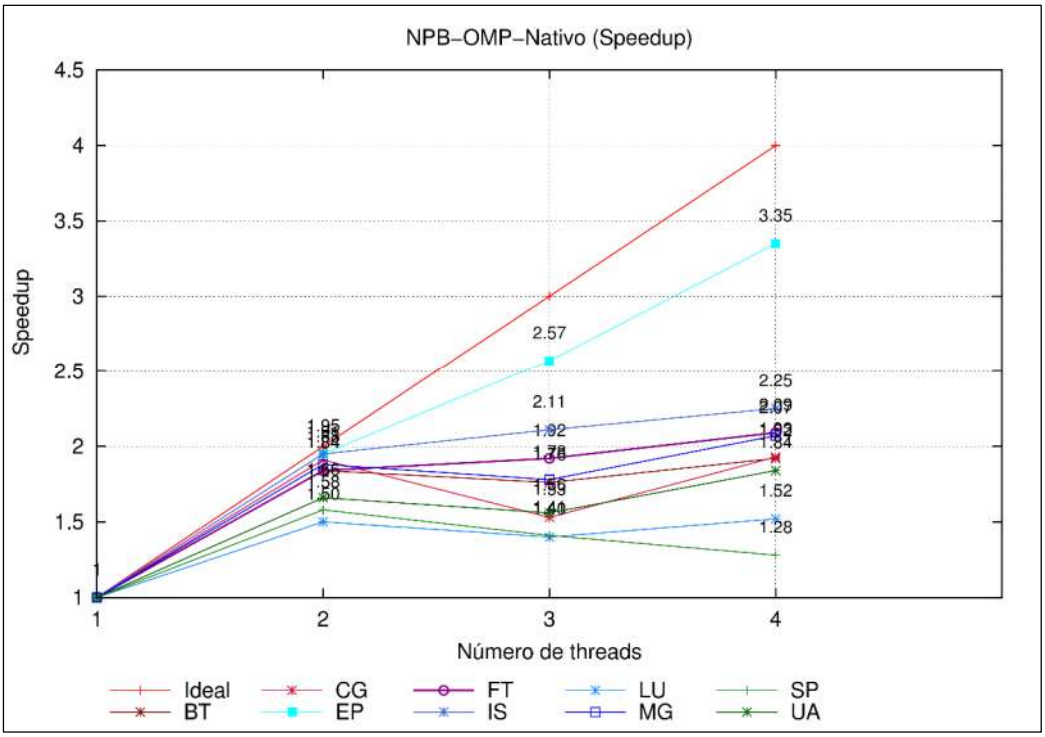
Figura 42: Tempo médio de execução NPB-OMP Nativo.

Na figura aonde mostra o tempo médio das execuções, todos os programas tendem a um ganho de desempenho quando a quantidade de *threads* é aumentada.

Porém, algo intrigante é com o programa CG e LU, onde apenas estes ao executar com 3 *threads*, seu tempo médio é elevado ao ponto de ultrapassar o tempo de execução com 2 *threads*, e quando executado com 4 *threads,* seu nível então quase iguala ao tempo de 2 *threads*. Acredita-se que devido ao programa fazer constantes acesso a memória RAM, e a capacidade de execução das *threads* virtuais serem inferiores às *threads* físicas do processador, a execução tende a ter este aumento, mas que ao passar para 4, o ganho no tempo de execução em alguns casos quase permanece constante.

Contudo, o programa SP, também executa testes com alto uso de memória RAM, mas seu desempenho não tem um pico de elevação alto como o CG, mas seu tempo se mantém linear.

Um fator importante, que pode ser observado no gráfico, é que praticamente todos os programas a partir de 3 *threads*, não tem um ganho expressivo como alguns tem quando passam a executar com 2 *threads*. Isto pode ser um indício que o processador, pode não executar de forma otimizada as aplicações de alto desempenho em núcleos lógicos de sua arquitetura, aonde seu maior desempenho pode ser obtido de forma mais intensa nos núcleos físicos.



Fonte: Maron, Griebler. 2014

Figura 43: Gráfico speed-up NPB-OMP Nativo.

O gráfico da Figura 43, mostra os valores de *speed-up* das execuções dos programas da suíte NPB-OMP, no gráfico a coluna vertical representa os valores do *speed-up,* e na vertical o número de *threads.*

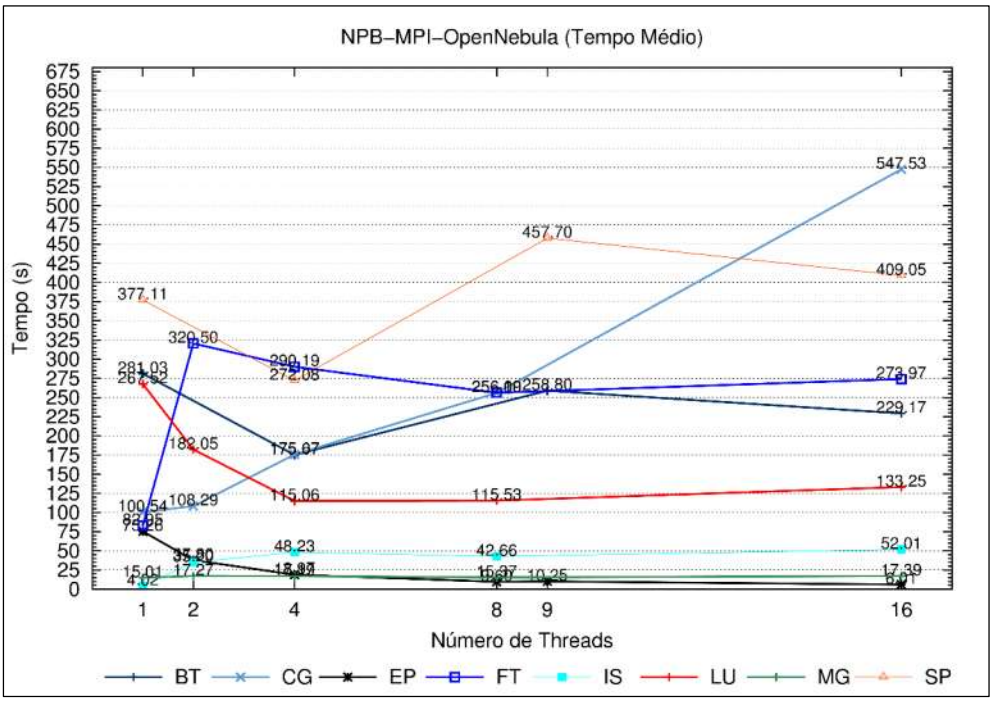
Com pode ser observado no gráfico de tempo de execução, os maiores ganhos estiveram nas execuções com até 2 *threads*, no gráfico da Figura 43, aonde mostra os valores de *speed-up*, o contexto não é diferente, os maiores ganhos de desempenho estão nesta mesma linha, e após 2 *threads* grande parte dos programas se mantém quase lineares. Com exceção do CG que tem a maior queda no ganho, quando executado em 3 *threads,* e o EP que acompanha muito próximo a linha ideal do *speed-up* demonstrando que seu ganho de permanece a cada aumento de *thread*.

3.3.2.2 OpenNebula

Os resultados dos testes no ambiente Nativo, serviram para trazer uma base para os resultados dos ambientes virtuais. Então, da mesma forma como foi aplicado os testes no ambiente Nativo, as instâncias da ferramenta OpenNebula

também passaram pelo mesmo processos de avaliação das aplicações paralelas, o NPB padrão OMP e MPI.

Inicialmente serão apresentados os resultados das execuções do NPB-MPI, aonde na Figura 44, é apresentado um gráfico contendo os valores tempo de execução que é representado na coluna vertical e, na horizontal é representado a quantidade de processos durante a execução.



Fonte: Maron, Griebler. 2014

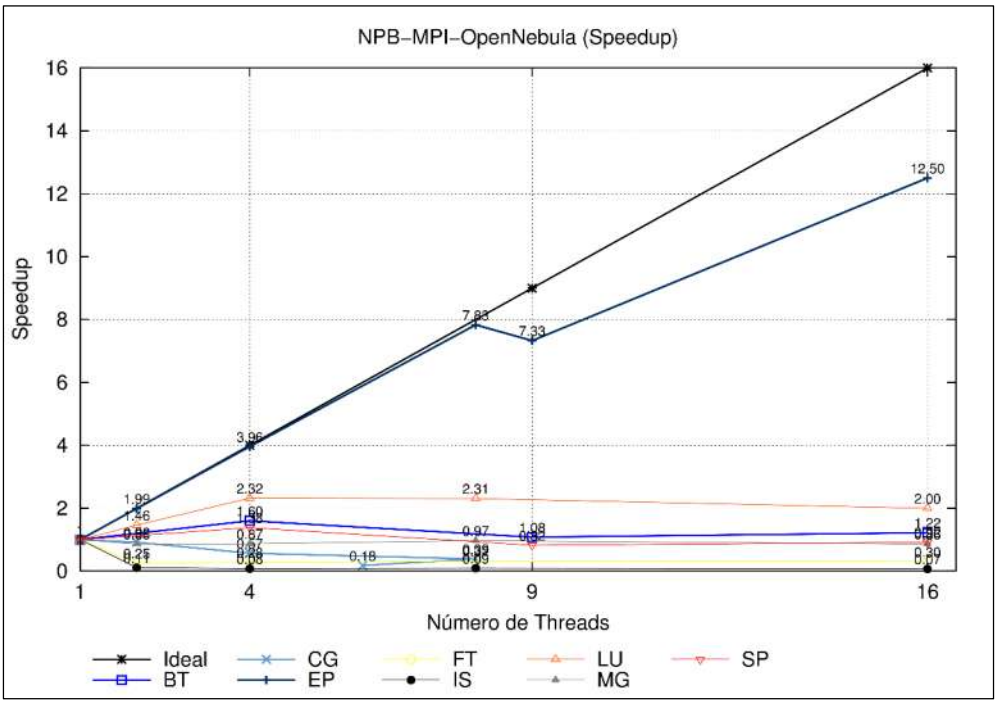
Figura 44: Gráfico com tempo médio do NPB-MPI OpenNebula.

No gráfico aonde são mostrados os valores de tempo médio das execuções no ambiente OpenNebula, é possível perceber uma similaridade nos resultados do ambiente Nativo, na distribuição dos pontos da área do gráfico.

Como pode ser observado na Figura 44, o CG possui seu tempo de execução em elevação ao ser aumentando a quantidade processos no decorrer dos testes. O FT ao implementar dois processos, tem um grande aumento no tempo de execução, que se mantem quase linear até atingir os 16 processos. O Programa LU, gradativamente com o aumento dos processos, tem uma diminuição no tempo de

execução. E o SP e MG desde as execuções com um 1 processo, até atingirem 16, não tem um ganho expressivo no seu tempo, mantem uma linha praticamente linear.

A Figura 45, representa em forma de gráfico os valores de *speed-up* das execuções do NPB-MPI no ambiente OpenNebula. Na coluna vertical da lateral do gráfico são apresentados os valores de *speed-up*, e na horizontal, são representadas as quantidades de processos durante as execuções.



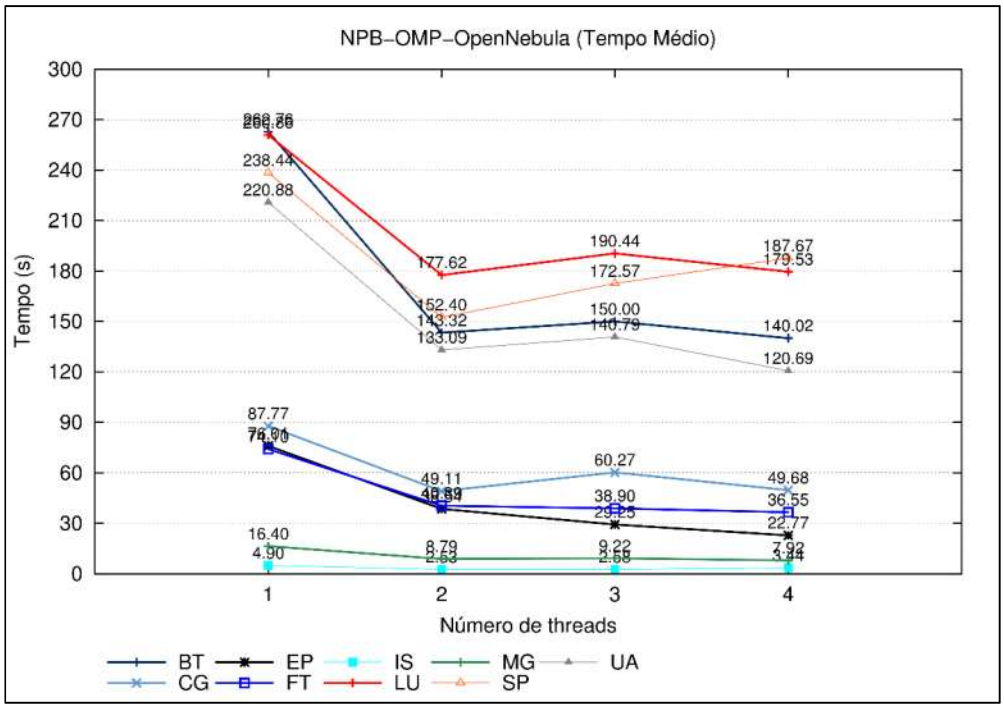
Fonte: Maron, Griebler. 2014

Figura 45: Gráfico de speed-up do NPB-OMP OpenNebula.

No gráfico de *speed-up* espera-se que todas as execuções, tendem a seguir a linha ideal que é representada no gráfico. Entretanto, alguns programas como o IS, o CG, FT, iniciam suas curvas menores ao padrão ideal. E o restante tem uma leve elevação do *speed-up* até 4 processos, e posterior a isto mantem seus ganhos lineares na análise do gráfico. Com exceção do EP que acompanha muito próximo o nível ideal.

A suíte NPB-OMP também seguiu o mesmo padrão de execuções do ambiente Nativo, portanto a Figura 46 representa os resultados obtidos nas execuções dos programas da suíte NPB-OMP no ambiente OpenNebula. Nele são apresentados

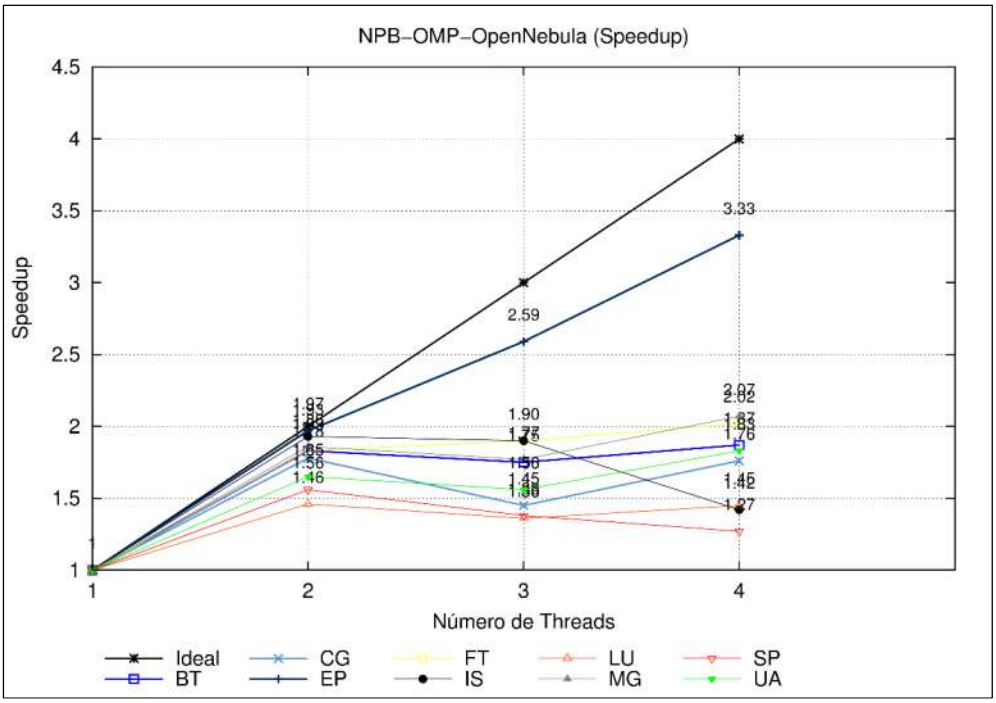
os valores de tempo em segundos na vertical do gráfico e, na horizontal, são representados as *threads.*



Fonte: Maron, Griebler. 2014

Figura 46: Gráfico de tempo médio do NPB-OMP OpenNebula.

Novamente os pontos da área do gráfico apresentam uma semelhança com os resultados dos testes no ambiente Nativo. O CG, apresenta um nível maior no grau de elevação no tempo de execução com 3 *threads,* comparado aos outros programas. Entretanto, praticamente todos tendem a ter um nível de elevação no tempo ao executarem seus testes com 3 *threads.* Com exceção do EP, que tende a manter a sua linha de ganho.



Fonte: Maron, Griebler. 2014.

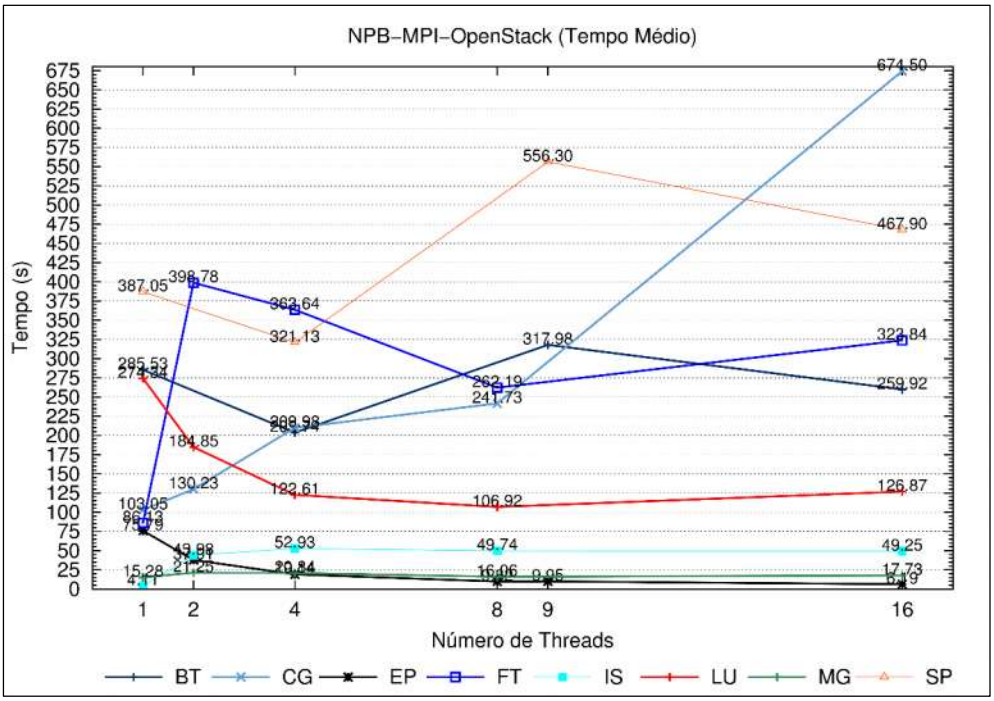
Figura 47: Gráfico do speed-up do NPB-OMP OpenNebula.

O gráfico mostra que o ganho de desempenho, mantem-se até o atingir 2 *threads.* Com 3 e 4 *threads* o ganho fica irregular para os programas executados no ambiente OpenStack. E percebe-se claramente uma perda do programa IS ao ser executado com 4 *threads.* Com exceção o EP, que mantem seu nível de ganho muito próximo ao padrão ideal das execuções.

3.3.2.3 OpenStack

Seguindo os mesmos procedimentos realizados em todos os ambientes, as instâncias da ferramenta OpenStack também serviram para a execução dos programas da suíte NPB nos padrões OMP e MPI.

A Figura 48 representa o tempo médio das execuções dos programas da suíte NPB-MPI no ambiente OpenStack. Nela são representados os resultados dos programas aonde a coluna vertical representa o tempo em segundos, e a coluna horizontal representa a quantidade de processos.



Fonte: Maron, Griebler. 2014

Figura 48: Gráfico de tempo médio do NPB-MPI OpenStack.

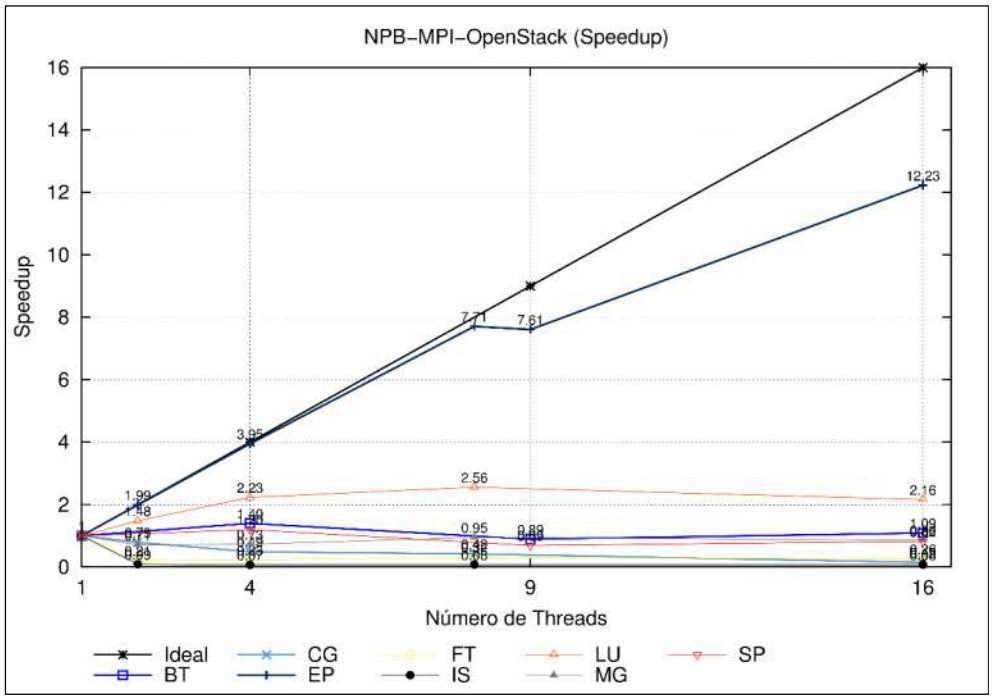
As execuções dos programas no ambiente OpenStack, novamente se observa uma similaridade nos pontos do gráfico em relação às outras execuções.

No FT observa-se um alto custo de execução quando o programa passa a ser executado com 2 processos, mas até 8 processos mantem um ganho no tempo, que ao chegar em 16, novamente tem seu tempo aumentado. CG tem um alto custo no tempo de execução quando a quantidade de processos vai aumentando, o seu tempo é sempre maior até chegar em 16 processos, o que deveria ter o comportamento ao contrário.

O IS novamente tem seu desempenho prejudicado quando a quantidade de processos é maior que 1, mas após isso, segue uma linha linear não tendo muitas variações. O MG tem seu tempo de execução quase que constante em todos os níveis de processos, até chegar aos 16 processos, seu tempo praticamente não sofre alterações.

No EP é possível observar que seu tempo de execução é menor a cada aumento da quantidade de processos. Comportamento que se repetiu durante os outros ambientes.

A Figura 49 representa o gráfico de *speed-up* dos programas da suíte NPB-MPI. Na coluna vertical do gráfico são expressos os valores de *speed-up*, e na vertical são mostrados a quantidade de processos durante as execuções



Fonte: Maron, Griebler. 2014

Figura 49: Gráfico speed-up do NPB-MPI OpenStack.

Novamente é possível perceber uma semelhança nos pontos do gráfico com os resultados dos outros ambientes. O EP segue muito próximo a linha ideal representada no gráfico. Já o IS, FT CG, e MG, não conseguem manter um ganho de desempenho durante as execuções. E o restante dos programas seguem em uma linha contínua onde existem pouca variação a cada aumento da quantidade de processos.

A próxima seção, apresenta um comparativo de cada um dos programas executados da suíte NPB MPI e OMP.

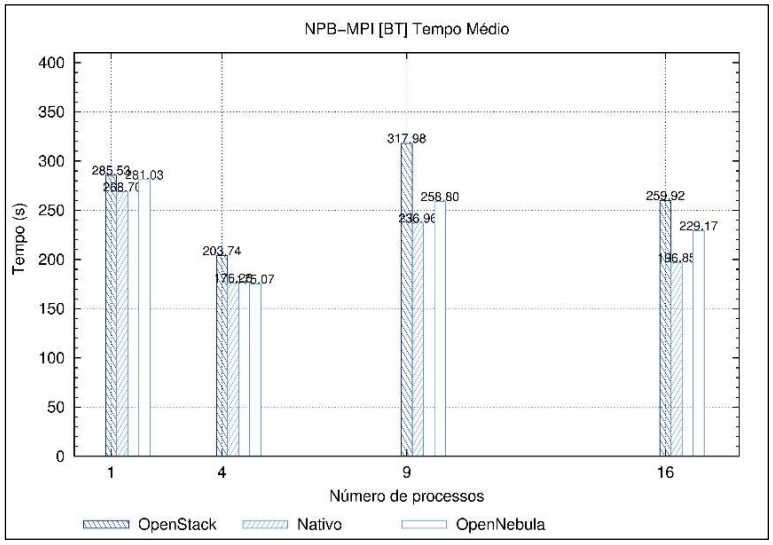
3.3.3 **Comparativo**

Na seção anterior percebeu-se uma grande semelhança nos pontos dos gráficos de *speed-up* e tempo de execução de cada um dos ambientes. Porém esta seção contará com gráficos individuais comparando cada um dos programas no padrão MPI e OMP. Isto para poder perceber a real diferença em cada uma das execuções, em cada ambiente (Nativo, OpenStack e OpenNebula), buscar compreender o real motivo de cada desempenho.

3.3.3.1 MPI

A seguir serão apresentados os gráficos comparando os resultados de cada ambiente usado na pesquisa. Os gráficos tendem a mostrar informações como como tempo médio, a eficiência e o *speed-up,* portanto, em todos os gráficos a seguir a coluna esquerda representa os valores de eficiência, aonde as linhas do gráfico traçam o resultado desta medida em cada ambiente. A unidade de medida de *speed-up* é demonstrada na lateral direita, onde as caixas traçam os valores obtidos em cada ambiente. E na horizontal, é informado apenas os processos paralelos das execuções.

Os gráficos aonde são comparados os tempos de execução, na vertical, são representados os tempos. E na horizontal são representados a quantidade de processos.



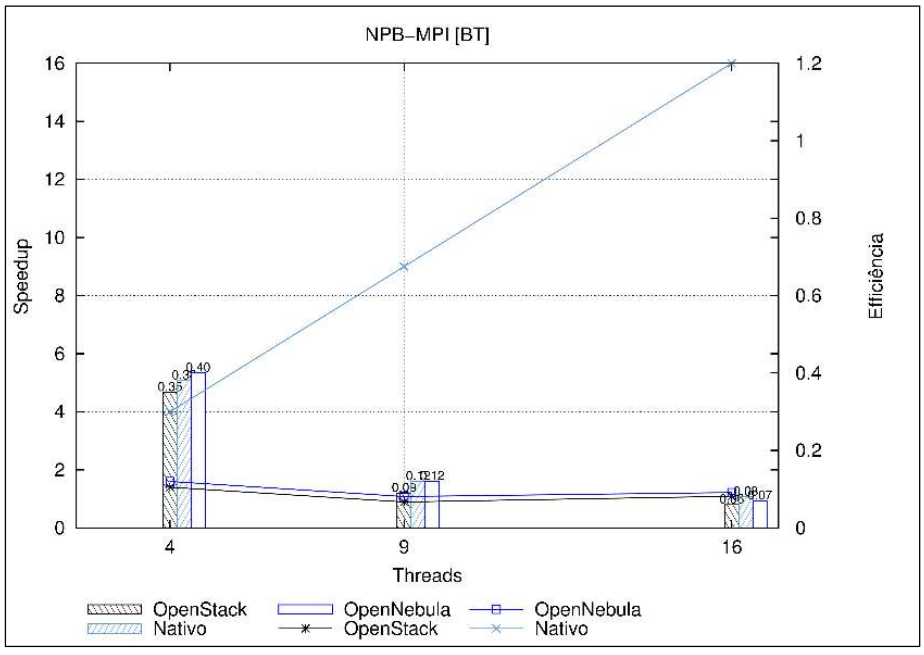
Fonte: Maron, Griebler. 2014

Figura 50: Comparação do tempo médio do NPB-MPI [BT]

A Figura 50 apresenta a comparação do tempo médio entre os ambientes. Nela é possível perceber que o OpenStack tem a maior média nos tempos de execução.

O KVM, é virtualizador usado para virtualizar as instâncias de cada umas das ferramentas, porém no OpenStack, existe o componente "*nova-compute-kvm*", então devido ao programa BT implementar cálculos de matrizes resultando em um maior consumo de memória e processador, acredita-se que possa prejudicar o desempenho do OpenStack.

A Figura 51 apresenta a comparação dos resultados do programa BT do NPB-MPI dos ambientes Nativo, OpenStack e OpenNebula. Percebe-se que ao executar com 4 processos, o ambiente OpenNebula utiliza melhor os recursos computacionais dentre os outros ambientes. Mas a situação se difere com o restante das execuções. Entre as ferramentas a diferença se torna pequena nos resultados de eficiência, mas no *speed-up*, OpenStack e OpenNebula se distância muito do ambiente Nativo.



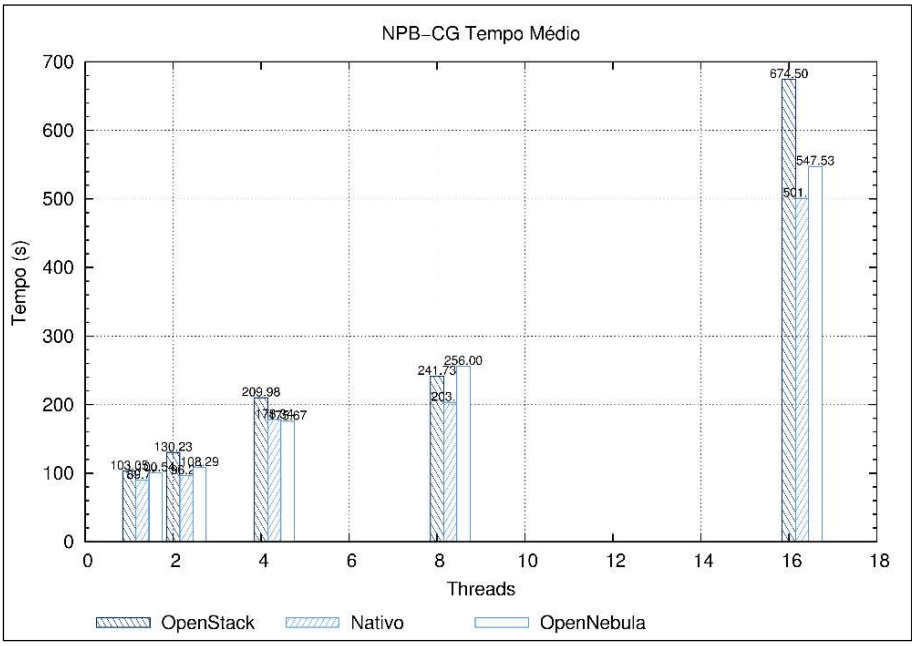
Fonte: Maron, Griebler. 2014

Figura 51: Comparação dos ambientes NPB-MPI [BT].

Porém, entende-se que devido as instâncias serem implementados na forma de virtualização completa. Acontece que ambas as ferramentas aparentam ter

uma perda na eficiência. Acredita-se que devido a camada de virtualização não atender a demanda dos requisitos do programa BT.

A Figura 52 apresenta a comparação do tempo médio de execução do programa CG. Em todos os resultados, a OpenStack teve a maior média de execução. Entretanto, com 4 e 9 processos a ferramenta OpenNebula se manteve próxima ao ambiente Nativo. E com 16 o tempo da ferramenta OpenStack teve um ganho no ficando à frente da ferramenta OpenNebula.



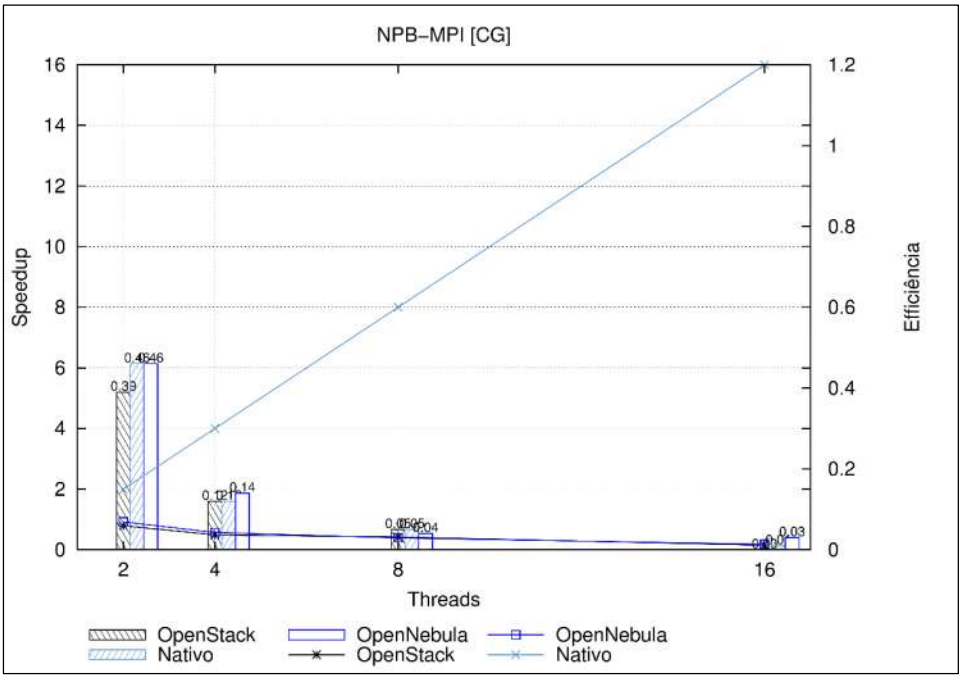
Fonte: Maron, Griebler. 2014

Figura 52: Comparação do tempo médio do NPB-MPI [CG].

Percebe-se que a cada aumento na quantidade de processos, aumentam o tempo de execução em ambos os ambientes. O programa CG implementa cálculos em *grid* exigindo uma grande comunicação entre processador e memória. Com 1 processo, as execuções sem mantém em níveis baixos, porém com o aumento gradual dos processos, aumenta-se a exigência dos recursos de memória e processador. Com 2 processos os ambientes se mantém próximos, porém ao serem implementados com 4 exige-se ainda o recurso de rede, fazendo com que o tempo de execução tenha um salto expressivo.

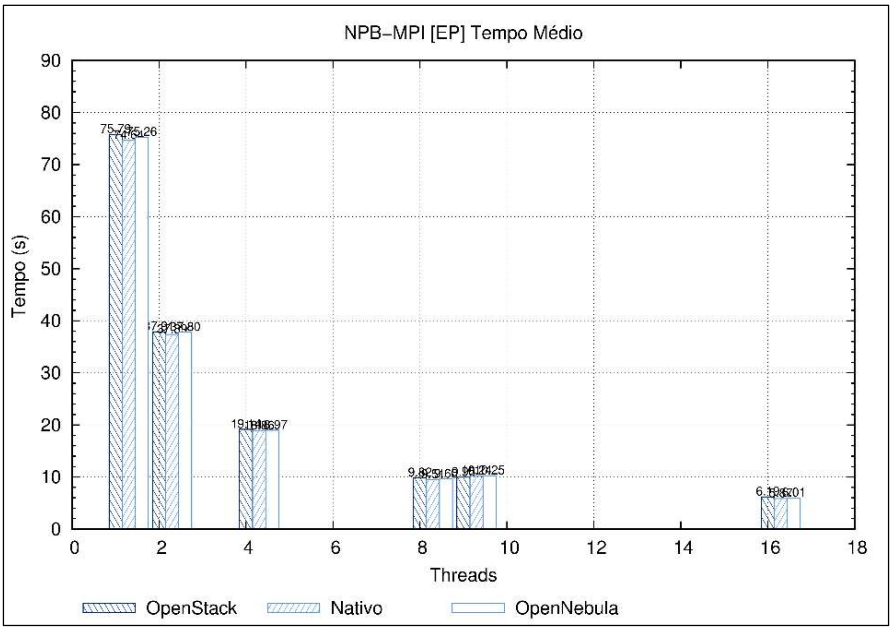
Portanto ao ser executados com 16 processos, os recursos de memória, processador, e principalmente rede estão saturados por não atender suficientemente a quantidade de informações em trânsito na rede, e faz com que os tempos de execução, se elevem. E devido a ferramenta OpenStack não apresentar um bom desempenho na rede, ela se manteve distante dos resultados da ferramenta OpenNebula que teve um bom desempenho de rede.

A Figura 53 representa a comparação de *speed-up* e eficiência do programa CG. Com 2 processos a eficiência do OpenNebula se equipara com o ambiente Nativo. Com 4 e 8 a eficiência de ambas as ferramentas ficam muito próximas no resultado. Mas é com 16 que torna preocupante a utilização dos recursos computacionais. O ganho de desempenho nos ambientes OpenStack e OpenNebula são menores, e se mantem longes do ambiente Nativo. Acredita-se pelo mesmo modo que os resultados de tempo de execução sofreram alterações, o ganho de desempenho e eficiência são prejudicados pelos recursos de memória e processador, e ainda pela rede de comunicação.



Fonte: Maron, Griebler. 2014

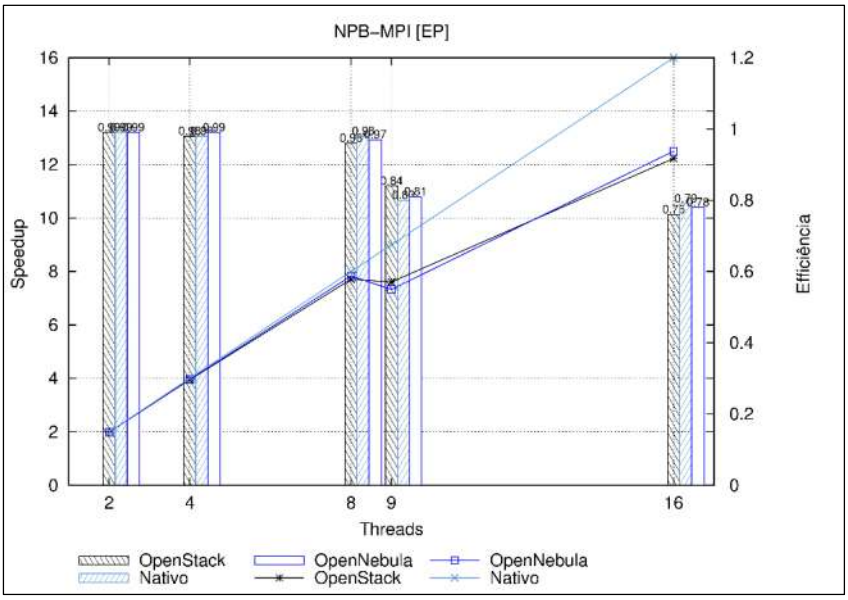
Figura 53: Comparação eficiência e speed-up NPB-MPI [CG].



Fonte: Maron, Griebler. 2014

Figura 54: Comparação tempo médio NPB-MPI [EP]

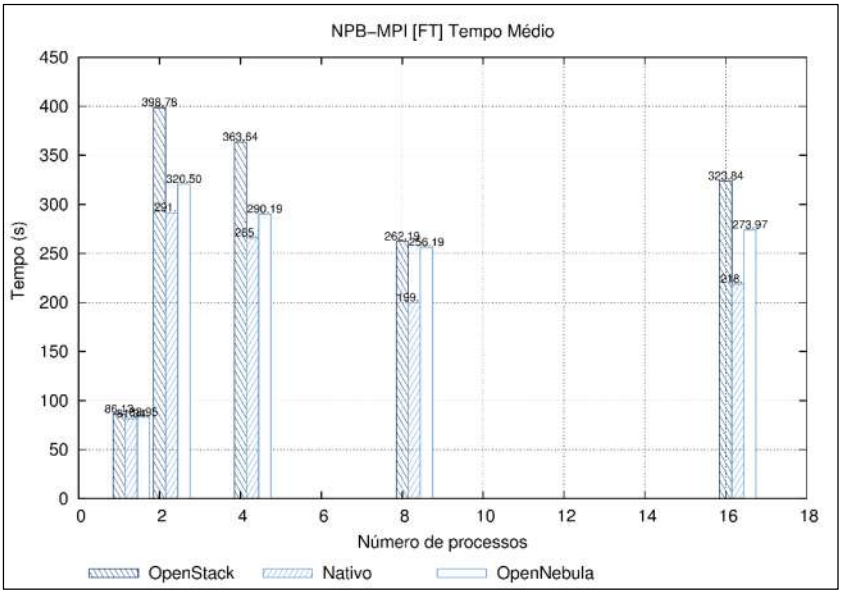
A Figura 54 mostra a comparação dos resultados do programa EP, em todos os ambientes e processos, os resultados se mantem muito próximos. Pois o EP neste caso utiliza cálculos que não são implementados de forma que seja exigida de forma significativa a comunicações entre processos pela rede, fazendo suas execuções inteiramente no processador e exigindo pouco da memória RAM. Durante os testes, ainda por se tratar de uma aplicação distribuída, o uso de rede é mínimo, trazendo um bom desempenho das ferramentas uso de memória e processador, mantendo suas médias próximas nos resultados.



Fonte: Maron, Griebler. 2014

Figura 55: Comparação eficiência e speed-up NPB-MPI [EP].

A Figura 55 mostra a comparação do *speed-up* e eficiência entre os dois ambientes. Pelos resultados, acredita-se que o EP teve um bom desempenho na execução em todos os ambientes. Apenas com 9 processos percebe-se um pequeno ganho de desempenho na ferramenta OpenStack.



Fonte: Maron, Griebler. 2014

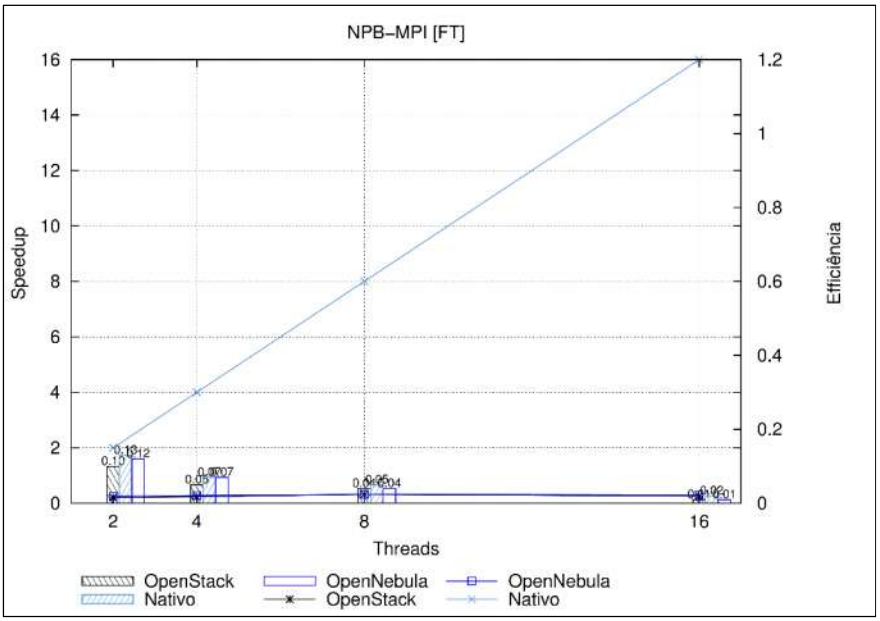
Figura 56: Comparação tempo médio NPB-MPI [FT]

A Figura 56 apresenta a comparação do tempo médio dos resultados obtidos nos ambientes da pesquisa. Percebe-se que em todos os testes, a ferramenta OpenStack tem a maior média do tempo de execução. Nas execuções com 2 e 4 processos, a OpenNebula fica próxima do ambiente Nativo, mas a partir disso, sua média acaba aumentando.

Mas importante lembrar que com 1 processo, ambos se mantem próximos e tem um bom desempenho, mas que ainda torna a ferramenta OpenNebula melhor neste tipo de execução. Isto pode ser justificado pelo alto uso de rede durante as execuções do programa FT. Porém, ao aumentar a quantidade de processos, o fluxo da rede aumenta, e devido os testes de infraestrutura mostrar um desempenho inferior na rede do OpenStack, os resultados do programa FT se justifica por isto, em a ferramenta OpenNebula apresentar um melhor resultado.

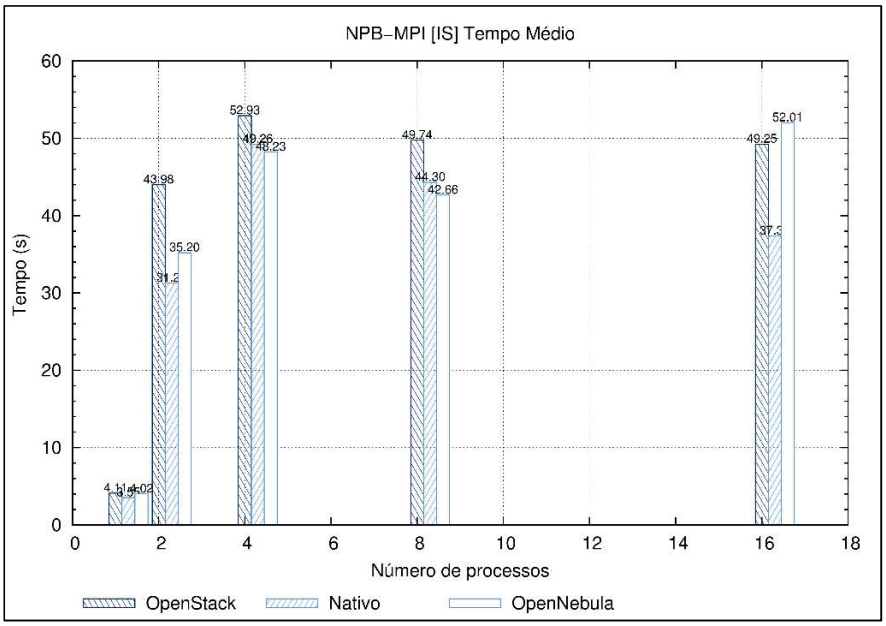
Mas nas execuções de 2 e 4 processos, existem um desempenho irregular da rede no OpenStack, mas nestas execuções os processos estão uniformemente distribuídos entre os nodos. Isso intriga pois mostra que o desempenho de rede além de ser menor que na ferramenta OpenNebula, se torna imprevisível, pois com 8 processos, a OpenStack teve um desempenho mais próximo ao OpenNebula.

A Figura 57 demonstra a comparação dos resultados de *speed-up* e eficiência do programa FT. Os resultados de eficiência mostram que o programa não tem um bom aproveitamento dos recursos computacionais. E é impossível perceber um ganho de desempenho durante suas execuções.



Fonte: Maron, Griebler. 2014

Figura 57: Comparação eficiência e speed-up NPB-MPI [FT]



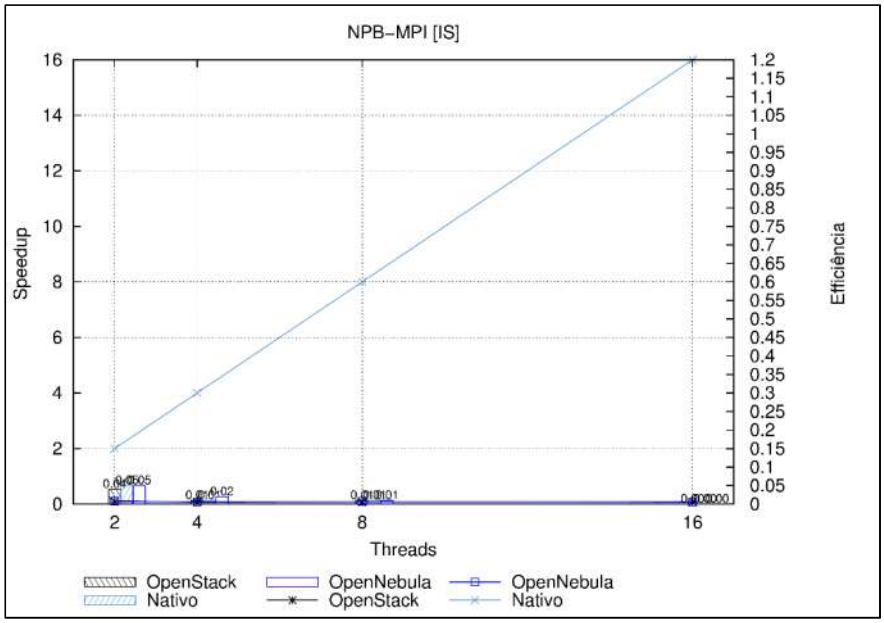
Fonte: Maron, Griebler. 2014

Figura 58: Comparação tempo médio NPB-MPI [IS]

A Figura 58 apresenta a comparação do tempo médio do programa IS. Na imagem percebe-se que nas execuções com 4 e 8 processos, a ferramenta OpenStack tem a maior média no tempo de execução. E a ferramenta OpenNebula, tem as melhores medias, ficando próximo do ambiente Nativo. Entretanto com 16

processos, OpenNebula teve a maior média de tempo, seguido pela ferramenta OpenStack, e mais distante o ambiente Nativo.

Nas execuções do IS ao serem executados com 1 processos, o desempenho entre os ambientes se tornam próximos, mostrando que ambas as ferramentas tendem a ter um bom desempenho do processador, já que o IS não exige tanto da memória RAM. O problema novamente persiste, quando existe o aumento dos processos, que fazem exigir mais dos recursos de rede neste programa, e ainda um pequeno uso nos recursos de armazenamento. Porém ao exigir mais a rede que armazenamento, a ferramenta OpenNebula se manteve melhor que a ferramenta OpenStack, mas onde a situação se inverte ao ser executados com 16 processos, onde acredita-se que devido ao bom desempenho de disco, tornou as médias melhores na ferramenta OpenStack.



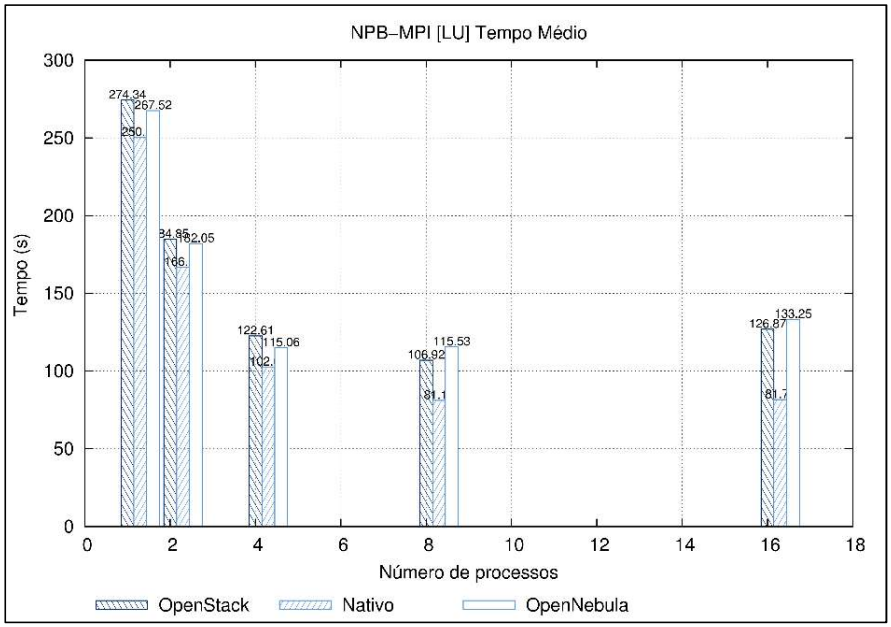
Fonte: Maron, Griebler. 2014

Figura 59: Comparação speed-up e eficiência do NPB-MPI [IS].

A Figura 59 demonstra a comparação dos resultados do programa IS. É percebível que tanto a eficiência, como *speed-up* se tornam quase nulos com 16 processos, e no restante os valores ficam muito próximos entre os ambientes, mostrando um mal desempenho deste programa.

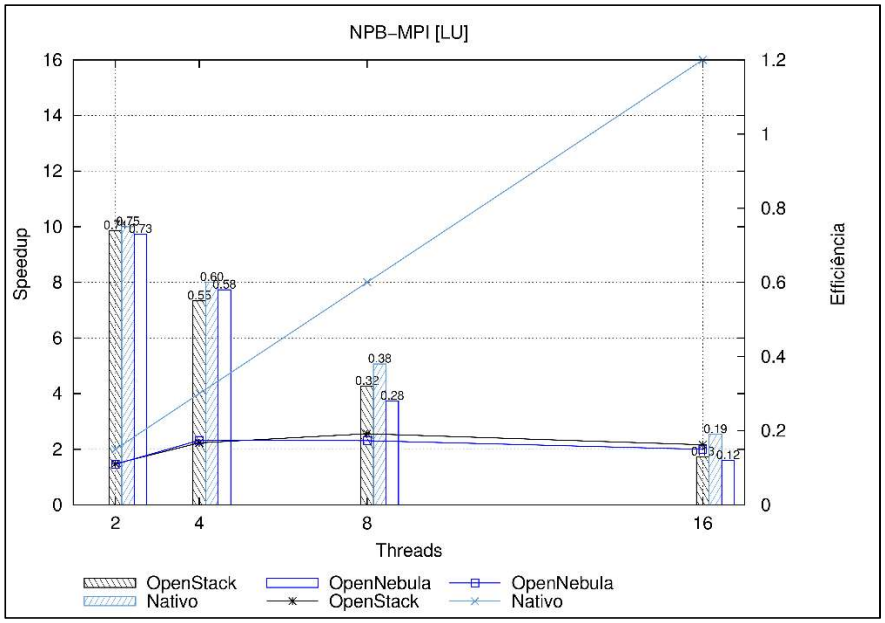
A Figura 60 demonstra a comparação do tempo médio das execuções do programa LU. É possível perceber que com o aumento da quantidade de processos, o tempo de execução diminui gradativamente principalmente no ambiente Nativo. OpenStack e OpenNebula busca acompanhar esse ganho, mas em comparação com as ferramentas, em todas as execuções a OpenNebula tem um melhor desempenho com uma pequena diferença entre as duas. Com exceção apenas, na execução com 2 processos, em que o OpenStack teve o melhor desempenho.

Durante a execução do programa LU, é exigido do sistema principalmente o processador para executar as cargas de trabalho. A arquitetura de virtualização imposta pelo KVM permite que as medias fiquem próximas entre as ferramentas, mas mostrando uma pequena superioridade da ferramenta OpenNebula por causa dos recursos de rede que LU exige durante os seus testes.



Fonte: Maron, Griebler. 2014

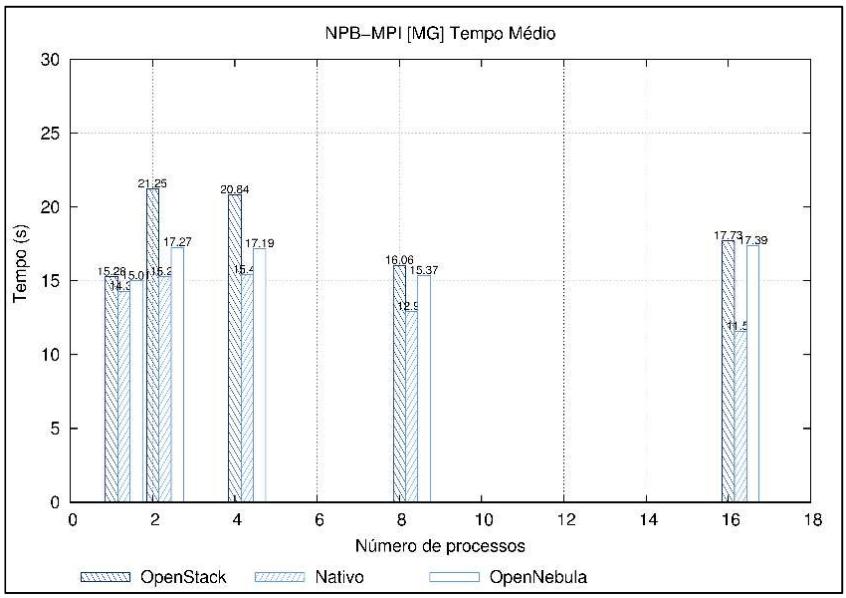
Figura 60: Comparação tempo médio do NPB-MPI [LU]



Fonte: Maron, Griebler. 2014

Figura 61: Comparação speed-up e eficiência do NPB-MPI [LU].

A Figura 61 representa a comparação entre eficiência e *speed-up* do programa LU. Deve-se notar neste gráfico que a eficiência em todas as execuções sempre se mantém próximas em ambos os ambientes. Mas ao passar a utilizar um maior número de processos, a ferramenta OpenStack tende a perder eficiência dos recursos.



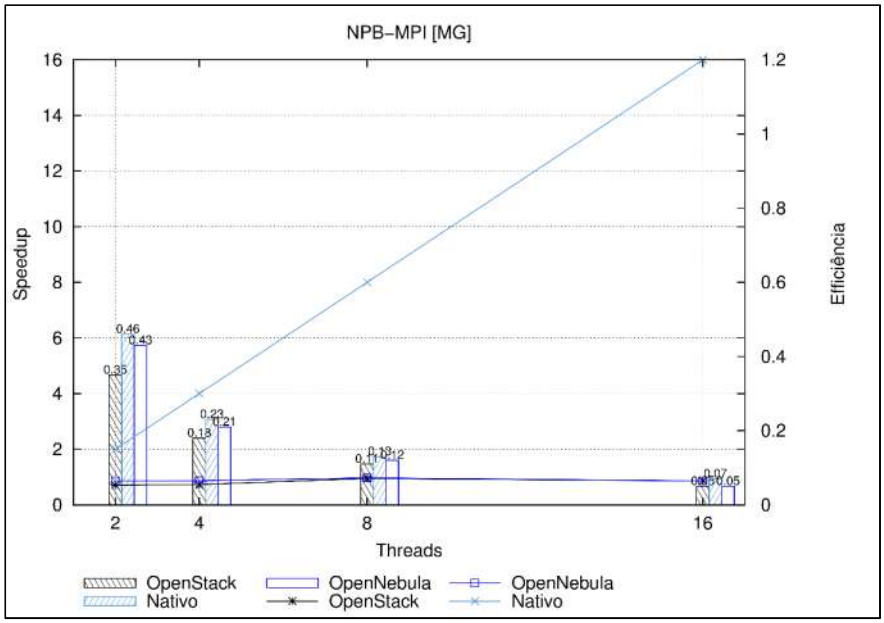
Fonte: Maron, Griebler. 2014

Figura 62: Comparação tempo médio do NPB-MPI [MG]

A comparação do tempo médio das execuções do programa MG, como é demonstrado na Figura 62, nas execuções de 8 e 16 processos, a diferença entre as ferramentas de administração de nuvem é muito pequena, mas ficando consideravelmente longe do tempo médio do ambiente Nativo. Mas é com 4 processos que acontece a grande diferença, OpenStack tem o maior tempo que a ferramenta OpenNebula, que fica mais próxima do Nativo, mais ainda que nas execuções de 8 e 16 processos.

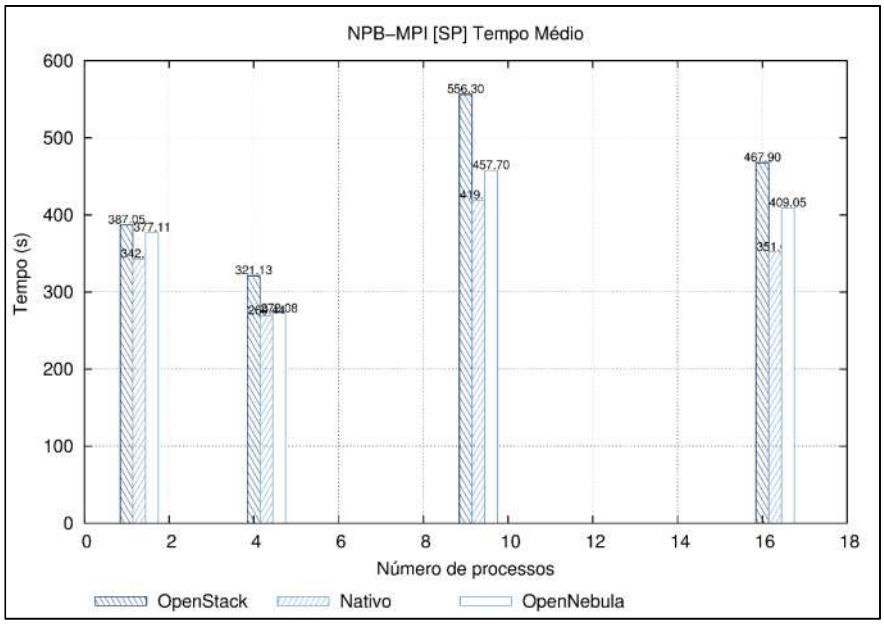
Na execução do programa MG, é exigido recursos de memória e processamento. Quando é executado com um 1 processo, o desempenho das ferramentas se mantém próximos. Mas quando aumenta a quantidade de memória a ferramenta OpenStack torna seu gerenciamento com o recurso de memória pouco eficiente, e juntamente a isso, pelas tarefas serem distribuídas a rede é exigida nestes casos, fazendo com que o tempo médio se eleve devido desempenho inferior da rede OpenStack.

A Figura 63 apresenta a comparação entre os resultados das execuções do programa MG. O nível do *speed-up* se mantém linear nas duas ferramentas, mas longe do ambiente Nativo. O ganho de eficiência nas execuções se mantém próximos, mas a maior diferença está nas execuções com 2 processos, onde a ferramenta OpenNebula apresenta um ganho maior que na ferramenta OpenStack. E contudo, nas execuções com 2 processos, onde se concentra o maior ganho de eficiência em ambos os ambientes, com o aumento da quantidade de processos, a eficiência demonstra diminuir.



Fonte: Maron, Griebler. 2014

Figura 63: Comparação speed-up e eficiência do NPB-MPI [MG].



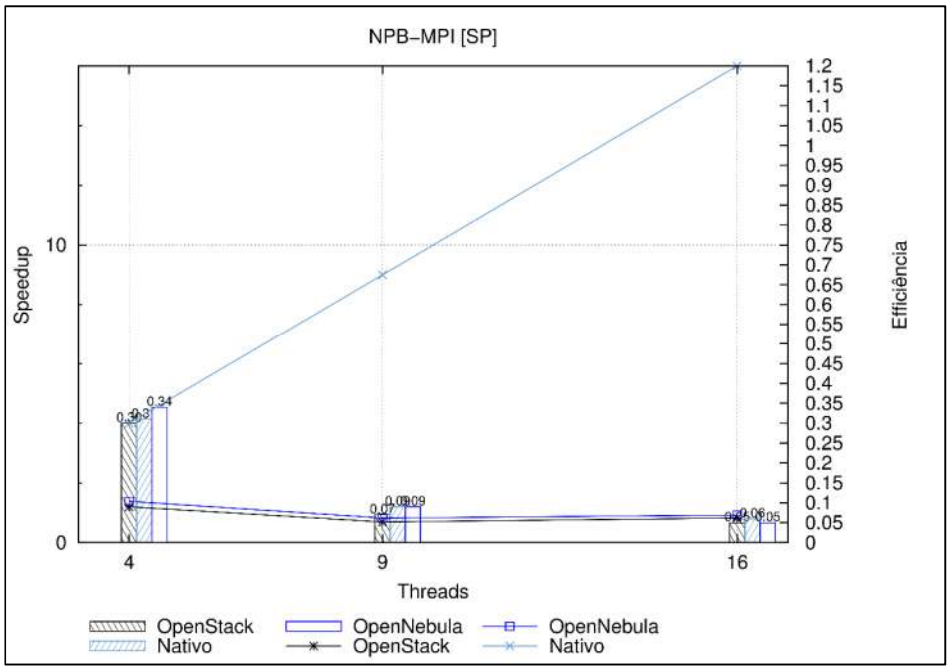
Fonte: Maron, Griebler. 2014

Figura 64: Comparação tempo médio do NPB-MPI [SP].

A comparação do tempo médio das execuções do programa SP como mostra a Figura 64, a ferramenta OpenStack se manteve com as maiores médias de tempo de execução, ficando ainda distante do ambiente Nativo. Nas execuções com 4 processo, OpenNebula ficou quase igualado ao ambiente Nativo. Contudo, ao

aumentar a quantidade de processos, o tempo médio das execuções também aumentam, em todos os ambientes. Neste programa, a ferramenta OpenNebula se manteve mais próxima ao ambiente Nativo.

Dentre os programas da suíte NPB-MPI, o SP é o que mais exige dos recursos de rede do ambiente. Durante sua execução, o nível de utilização dos recursos torna-se alto em comparação aos outros programas, justificando o melhor desempenho da ferramenta OpenNebula. É importante lembrar que durante a execução dos testes do SP, em ambas as ferramentas existe um maior uso das *threads* do ambiente. Porém, novamente a superioridade na ferramenta OpenNebula está devido ao bom desempenho da rede.



Fonte: Maron, Griebler. 2014

Figura 65: Comparação eficiência e speed-up do NPB-MPI [SP].

A Figura 65 demonstra a comparação entre os resultados de *speed-up* e eficiência das execuções do programa SP. No gráfico, os níveis de ganho de desempenho se mantém lineares, mas com um maior ganho apenas nas execuções com 4 processos. Os níveis de *speed-up* se concentram também nas execuções com 4 processos, onde com o aumento, os níveis vão diminuindo mas se mantendo próximos em cada um dos ambientes.

Em todos os gráficos apresentados, apenas nos resultados dos programas EP em que observa-se o real contexto do processamento paralelo. Em cada aumento da quantidade de processos, o tempo de execução dos programas diminui. E consequente a isto, o ganho de desempenho e eficiência também aumentam.

No restante das execuções, percebe-se um comportamento ao contrário e muito irregular, em ambos os ambientes. Acredita-se que o principal problema de todos os resultados instáveis das execuções, estejam diretamente ligados ao desempenho da rede. Aplicações paralelas no padrão MPI, necessitam uma rede preparada e compatível com as execuções que suporte um grande volume de dados e com um *delay* em níveis baixos. E um dos motivos que acredita-se que os resultados da ferramenta OpenStack seja inferior ao OpenNebula, é pelo componente desenvolvido por ela, que torna em alguns tipos de execução uma baixa da eficiência dos recursos computacionais. Também vale lembra que alguns casos, o desempenho se tornam compatíveis devido a implementação do virtualizador KVM em um ambiente Linux. No qual permite um ganho de desempenho, mesmo se tratando de um ambiente virtual.

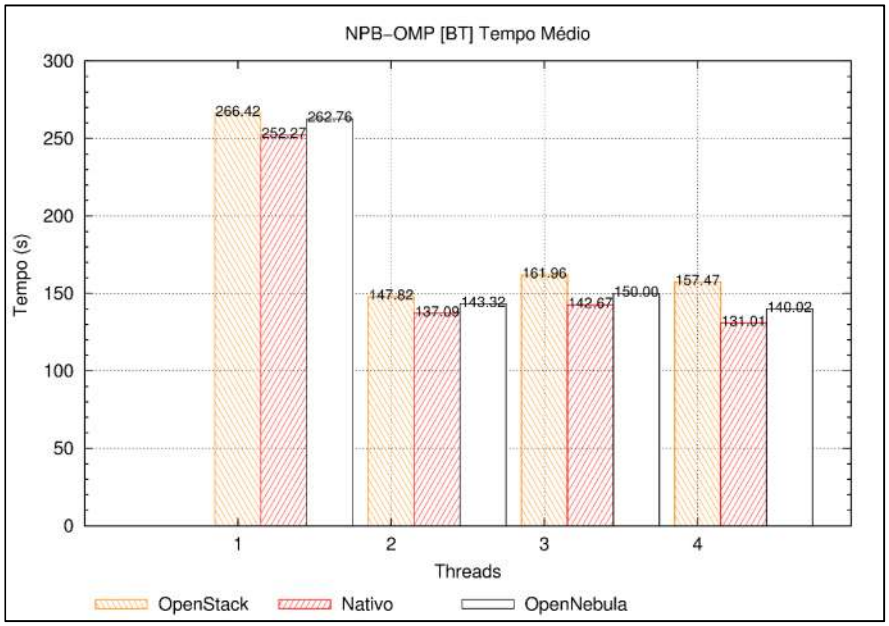
3.3.3.2 OMP

Nesta seção serão apresentados os resultados com as execuções do NPB-OMP em forma de gráficos. Em cada programa da suíte NPB, haverá um gráfico contendo a comparação do tempo médio das execuções em cada um dos ambientes. Cada gráfico terá o mesmo padrão, mudando apenas o programa e os resultados. Os gráficos de tempo médio de execução, serão descriminados na coluna vertical o tempo em segundos, e na horizontal, a quantidade de *threads* em cada execução.

Os gráficos demonstrando o desempenho em cada execução, estará descrito na lateral esquerda, os valores do *speed-up* que serão representados pelas linhas no gráfico. Já os valores eficiência, serão representados pelas caixas no centro do gráfico, definidas pelos valores de eficiência na lateral direta do gráfico. Na vertical será a quantidade de *threads* em cada execução.

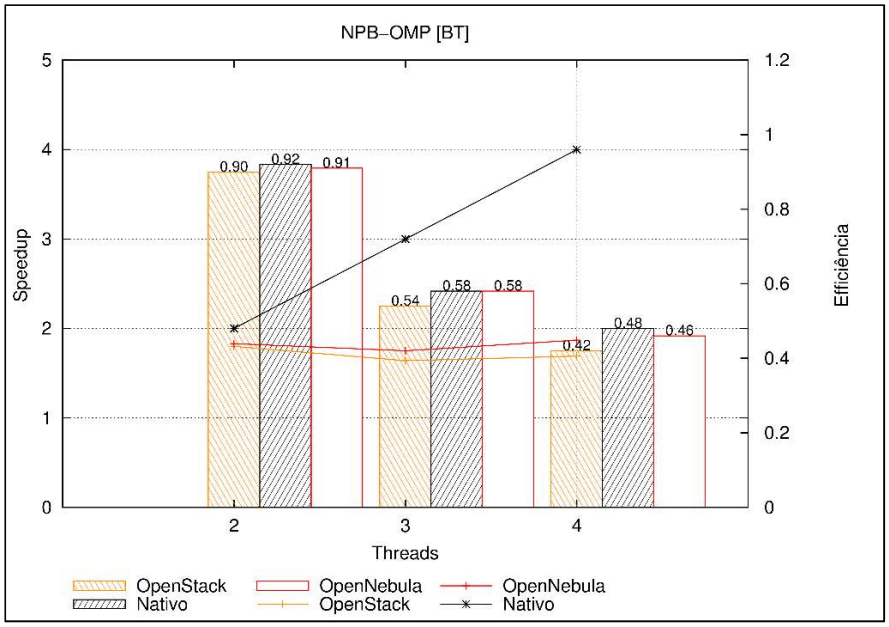
A Figura 66 demonstra o comparativo do tempo de execução do programa BT na suíte NPB-OMP. Observa-se no gráfico, que a ferramenta OpenStack tem as maiores médias de tempo nas execuções em todas as *threads*. Mas todas em níveis muito próximos ao Nativo. É importante ressaltar ainda que existe um ganho no tempo quando o programa passa a executar com 2 *threads,* mas no momento em que passa a ser executado com 3 e 4, o ganho não é tão expressivo.

Nota-se que a ferramenta OpenStack tem um aumento no tempo de execução. Sendo que o BT implementa cálculos que exigem recurso do processador e memória RAM. Portanto, acredita-se que os componentes do OpenStack que são implementados junto ao virtualizador, fazem com o seu tempo de execução degrade. Tornando neste tipo de execução, a ferramenta OpenNebula melhor.



Fonte: Maron, Griebler. 2014

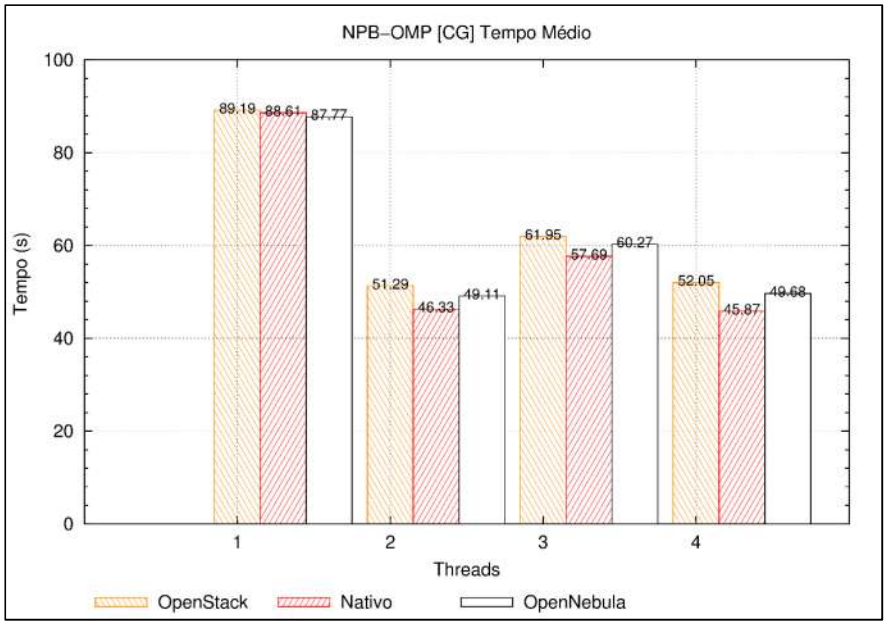
Figura 66: Comparação tempo de execução NPB-OMP [BT]



Fonte: Maron, Griebler. 2014

Figura 67: Comparação eficiência e speed-up do NPB-OMP [BT]

A figura 67 representa a comparação de eficiência e *speed-up* nas execuções do programa BT. Ao analisar o gráfico, percebe-se que os níveis de eficiência estão concentrados nas execuções com 1 e 2 *threads*, juntamente com o ganho de desempenho se mantendo próximo no Nativo. Mas ao executar com 3 e 4, o ganho não se torna muito alto, e ficam muito próximos entre si. Ainda é possível perceber que a ferramenta OpenNebula alcançou a mesma eficiência que o ambiente Nativo executando o programa em 3 *threads.*



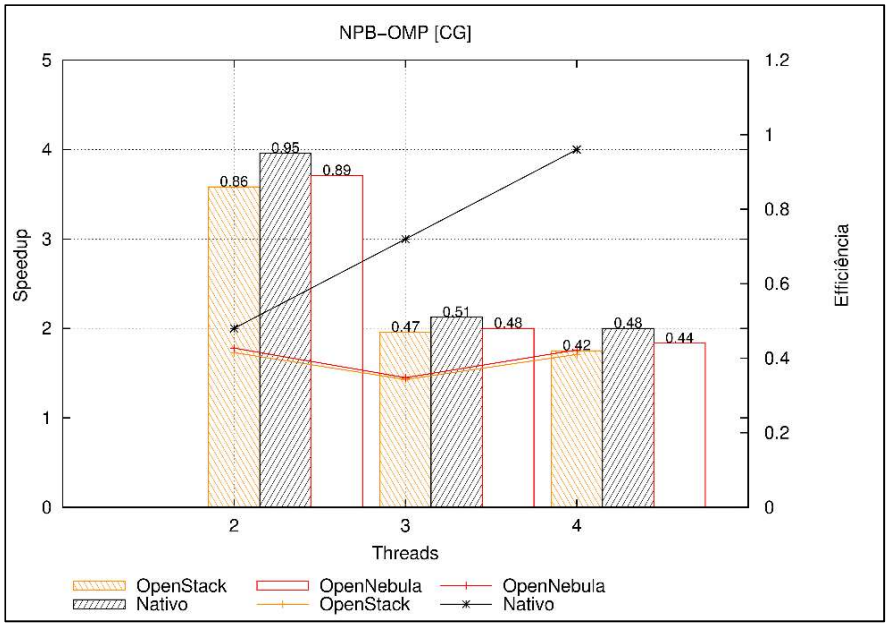
Fonte: Maron, Griebler. 2014

Figura 68: Comparação tempo médio de execução do NPB-OMP [CG].

Como é demonstrada na Figura 68, a comparação entre os tempos de execução do programa CG, com 1 *thread* os tempos quase se igualam, e ao passar a executar com 2 *threads*, existem um ganho expressivo no tempo. Porém, o tempo de execução é maior com 3 *threads*, e com uma pequena diminuição ao executar com 4. É importante ressaltar que ao executar o programa em 1 *thread*, a ferramenta OpenNebula teve a melhor média em comparação ao ambiente Nativo.

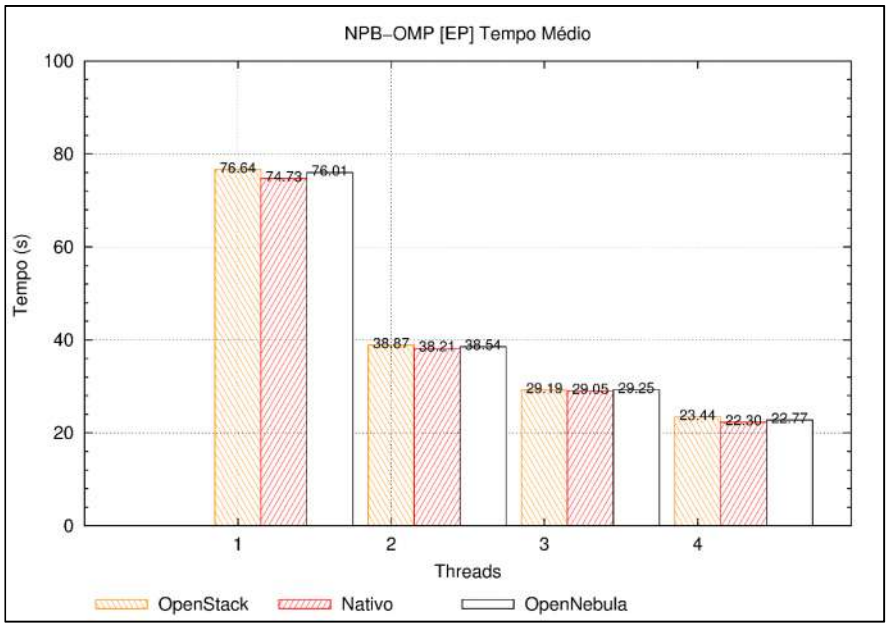
Esse resultado pode representar um melhor desempenho no uso de memória da ferramenta OpenNebula. Pois o CG ao ser executado, necessita de um volume de memória considerável pois suas principais cargas de trabalhos estão relacionadas a vetorização de valores, acesso irregulares de memória.

A Figura 69 representa a ganho de desempenho e a eficiência durante a execução do programa CG. O ganho no desempenho se mantém quando são executados em 1 e 2 *threads*, mas ao passar para 3 e subsequente com 4, o ganho sofre uma oscilação. Isso ocorre também com a eficiência, se mantem em níveis próximos ao máximo, mas ao passar a executarem em 3 e 4, a eficiência diminui.



Fonte: Maron, Griebler. 2014

Figura 69: Comparação da eficiência e speed-up do NPB-OMP [CG].

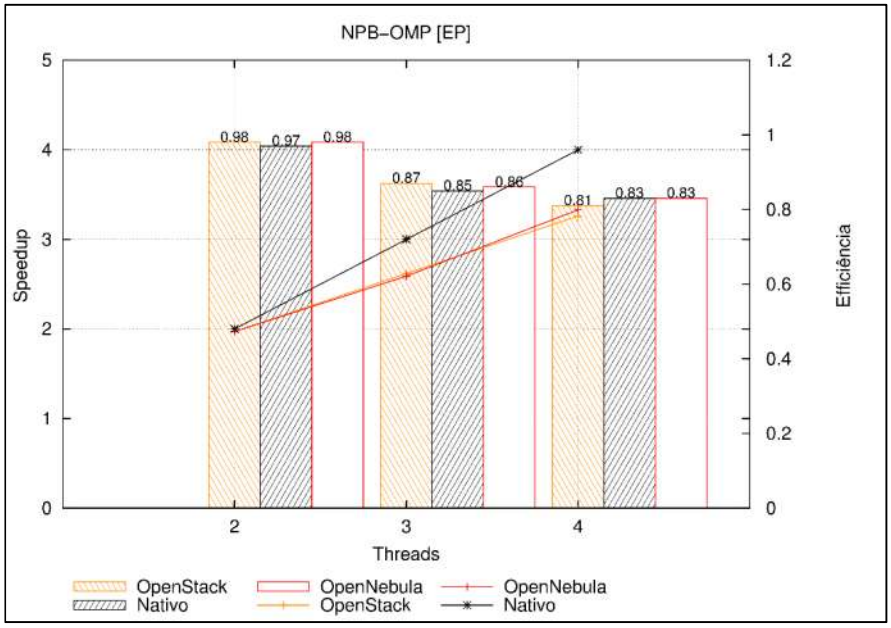


Fonte: Maron, Griebler. 2014

Figura 70: Comparação tempo médio NPB-OMP [EP].

A comparação da execução do programa EP nos ambientes da pesquisa, mostra os resultados na Figura 70, que o ganho no tempo de execução vai aumentando ao momento que se tem a execução em maior número de *threads*. E os resultados das médias se mantém próximas entre os ambientes.

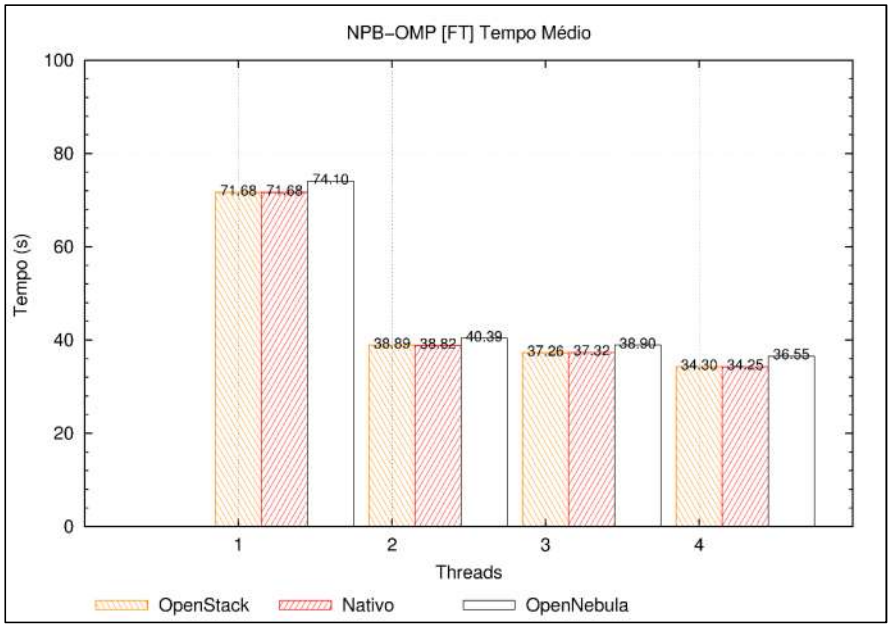
O EP ao ser executado, basicamente implementa cálculos de pontos flutuantes ao processador. Exigindo um mínimo de acesso à memória RAM. Isto mostra que a virtualização usada pelo KVM realiza bem estes tipos de cargas de trabalho, pois as médias se mantiveram muito próximas ao ambiente Nativo.



Fonte: Maron, Griebler. 2014

Figura 71: Comparação eficiência e speed-up do NPB-OMP [EP].

O programa EP mostrou um ganho gradativo no tempo execução ao momento que a quantidade de *threads* foi aumentando. Isto significa, que eficiência e ganho de desempenho seguem em níveis próximos ao recomendado, com pequenas diferenças entre os ambientes.

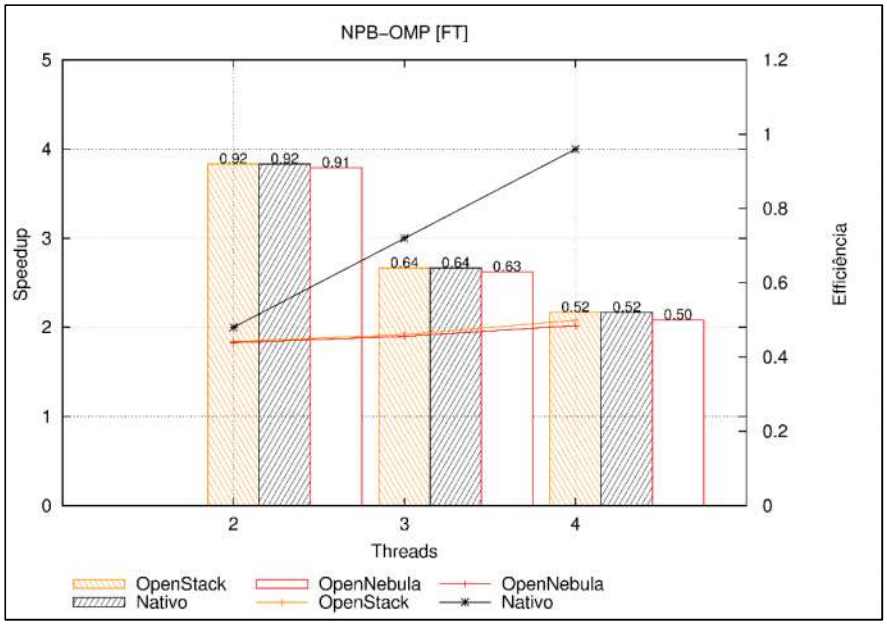


Fonte: Maron, Griebler. 2014

Figura 72: Comparação tempo médio do NPB-OMP [FT]

A Figura 72 coloca os valores das médias de execução do programa FT. Nesses resultados, percebe-se que a ferramenta OpenNebula tem as maiores médias de tempo. Resultados que são diferentes com a ferramenta OpenStack e ambiente Nativo, que em todas as execuções seguindo o aumento das *threads*, possuem resultados muito semelhantes.

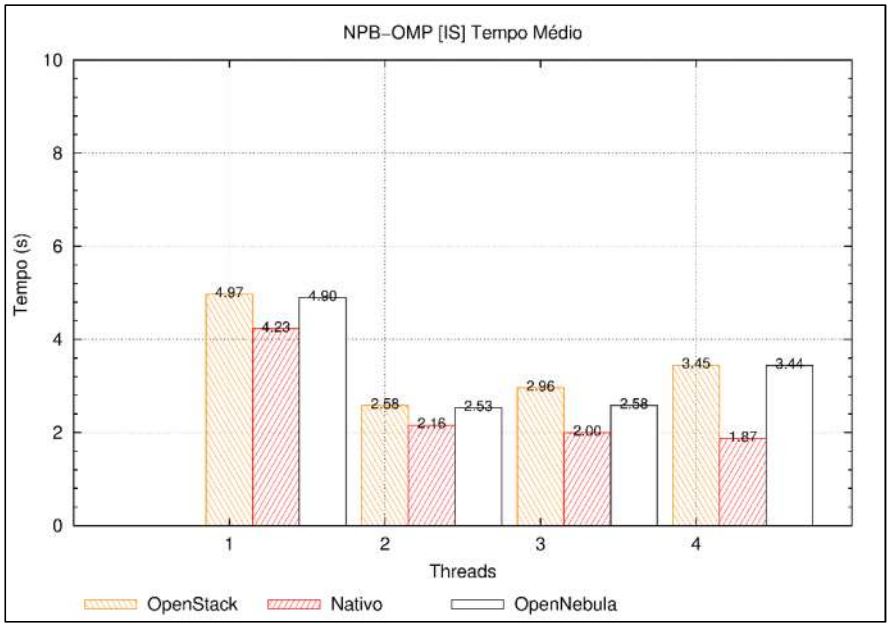
O FT quando executado, utiliza um grande volume de memória RAM, e devido a forma como o programa aplica suas cargas de trabalho, a ferramenta OpenNebula acaba não gerenciando bem os recursos exigidos pelo programa. Nota-se que durante a execução dos testes, o FT faz acessos à unidade de armazenamento, fazendo com que isto aumente seu tempo de execução, devido ao desempenho inferior que a ferramenta OpenNebula alcançou com as operações de disco.



Fonte: Maron, Griebler. 2014

Figura 73: Comparação eficiência e speed-up do NPB-OMP [FT]

Ao executar o programa FT, os níveis de eficiência e *speed-up* das execuções seguem uma diminuição quando aumentam a quantidade de *threads* para 3 e posteriormente à 4, como mostra a Figura 73. Entretanto, os valores de cada ambiente se mantém próximos no decorrer das execuções, com exceção o *speed-up*, onde as ferramentas têm um declínio, se mantendo distantes do ambiente Nativo.



Fonte: Maron, Griebler. 2014

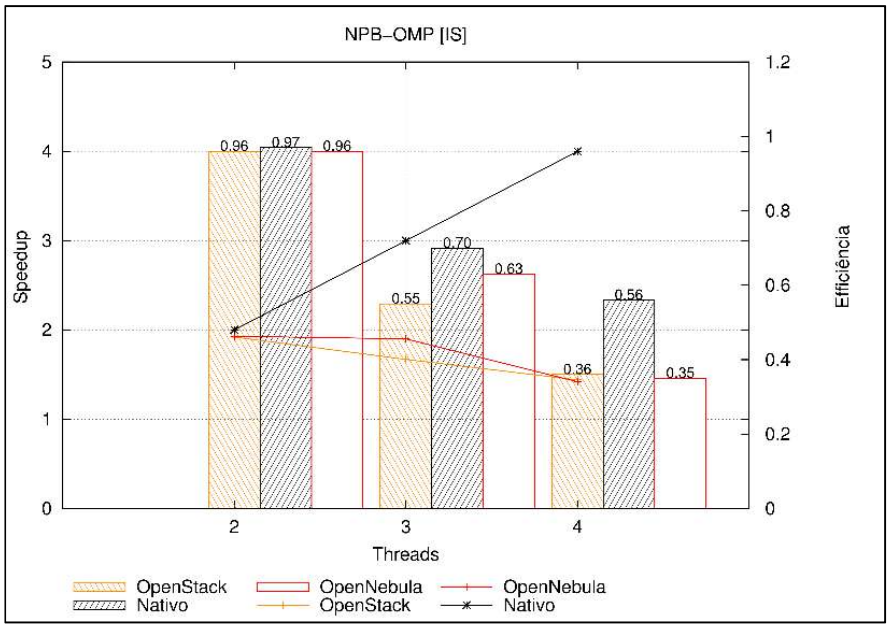
Figura 74: Comparação tempo de execução do NPB-OMP [IS].

Os resultados que a Figura 74 mostra os resultados do programa IS. Durante a execução, suas cargas de trabalho são aplicadas somente ao processador, exigindo um mínimo de recursos da memória RAM. As ferramentas de computação em nuvem apresentam resultados semelhantes entre si, mas mostram que o desempenho em executar estes tipos de cargas de trabalha, fazem com que tenhauma diferença considerável entre as ferramentas e o ambiente Nativo.

Mas contexto que não se repete com as ferramentas OpenStak e OpenNebula, suas médias se tornam um tanto irregulares, principalmente nas execuções com 4 *threads,* aonde espera-se que o tempo médio diminua, acontecendo o contrário, chegando a níveis maiores que as execuções com 2 *threads*. Entretanto, o tempo de execução entre as ferramentas tendem a se manterem iguais.

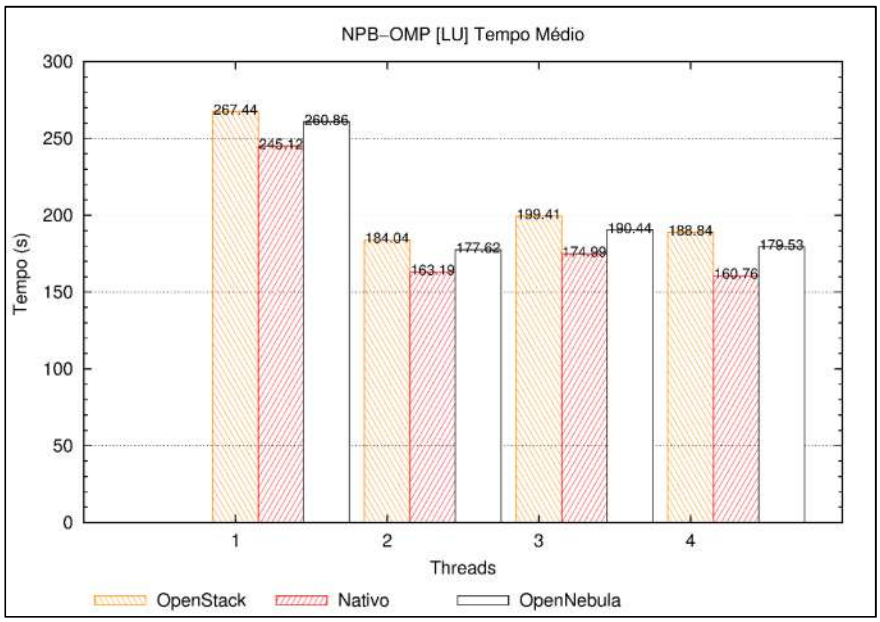
A Figura 75 mostra os níveis de eficiência e *speed-up* aonde nas execuções com 3 e 4 *threads*, os níveis são irregulares entre si, se mantendo iguais apenas nas execuções de 4 *threads* nas ferramentas OpenStack e OpenNebula. É importante lembrar, que a eficiência dos ambientes se mantém muito perto do máximo, quando executados com 1 e 2 *threads*, mas que refletiu de maneira diferente em cada um dos ambientes, pois o tempo de execução não são semelhantes como os níveis de

eficiência. Entretanto, o ganho de *speed-up* nas ferramentas OpenStack e OpenNebula, caem ao serem executados com 3 e 4 *threads.*



Fonte: Maron, Griebler. 2014

Figura 75: Comparação da eficiência e speed-up do NPB-OMP [IS].



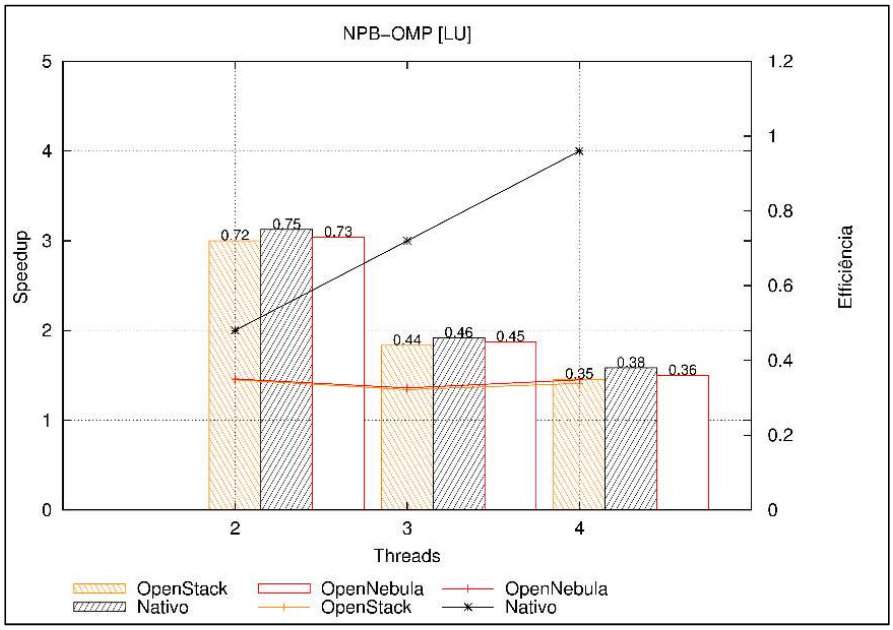
Fonte: Maron, Griebler. 2014

Figura 76: Comparação do tempo médio do NPB-OMP [LU].

A Figura 76 mostra a comparação da média dos tempos de execução do programa LU. No gráfico percebe-se que o ganho no tempo aumento quando o

programa é executado com 2 *threds*, mas ao passar para 3 e 4, seus níveis aumentam e se mantém quase semelhantes com as execuções em 2 *threads.* E dentre as execuções as ferramentas praticamente se igualam nos resultados de tempo.

O programa LU quando executado concentra suas cargas de trabalho principalmente no processador. Porém, nota-se ainda que quando está em execução, realiza algumas funções que exigem a rede. Mesmo se tratando de *benchmark* que paraleliza suas cargas localmente, o desempenho da rede na ferramenta OpenNebula é melhor, justificando assim, eu desempenho sobre a ferramenta OpenStack, que perdas em processo que utilizam a rede.



Fonte: Maron, Griebler. 2014

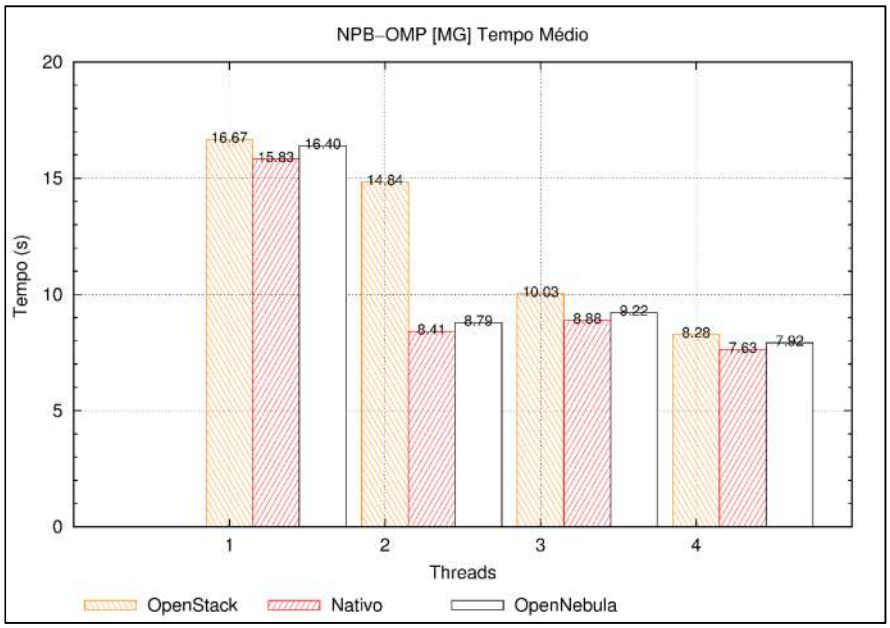
Figura 77: Comparação speed-up e eficiência do NPB-OMP [LU].

A Figura 77 demonstra que a eficiência do programa LU ao executar com 2, 3 e 4 *threads* decaem com o aumento das *threads*. Porém, os ambientes ainda continuam a manter um grau de similaridade em seus valores. O *speed-up* desde as execuções com 1 (uma) *thread*, mantem um nível linear, não mostrando muito ganho durante sua execução.

As execuções do programa MG nos ambientes, mostram a ferramenta OpenStack atingiu tempos médios superiores a todos os outros ambientes. Como

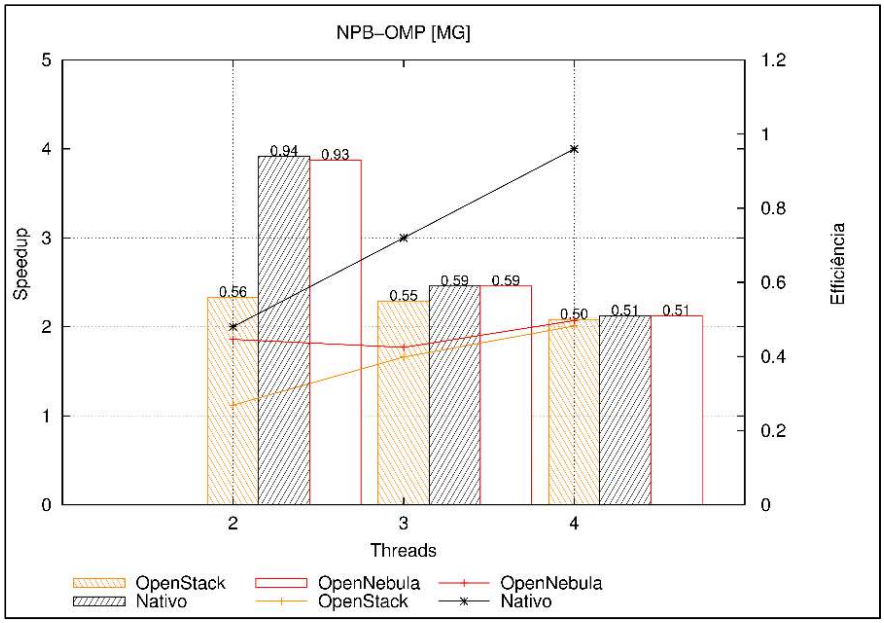
pode ser observado na Figura 78, ao passar a executar com 2 *threads* no ambiente OpenStack, o tempo médio aumenta chegando próximo às médias com 1 (uma) *threads*. Ao aumentar a quantidade de *threads* as médias diminuem, mas ainda se mantém elevadas em comparação aos outros ambientes. Nota-se ainda que a ferramenta OpenNebula se manteve próxima do ambiente Nativo.

Durante a execução do MG, o acesso a memória RAM se torna expressivo, porém, o desempenho alcançado pelo OpenStack durante a execução com 2 *threads* precisa ser melhor investigado, pois após o aumento das *threads*, seu tempo de execução diminui. Neste caso a ferramenta OpenNebula se mostrou melhor em gerenciar os acessos à memória RAM.



Fonte: Maron, Griebler. 2014.

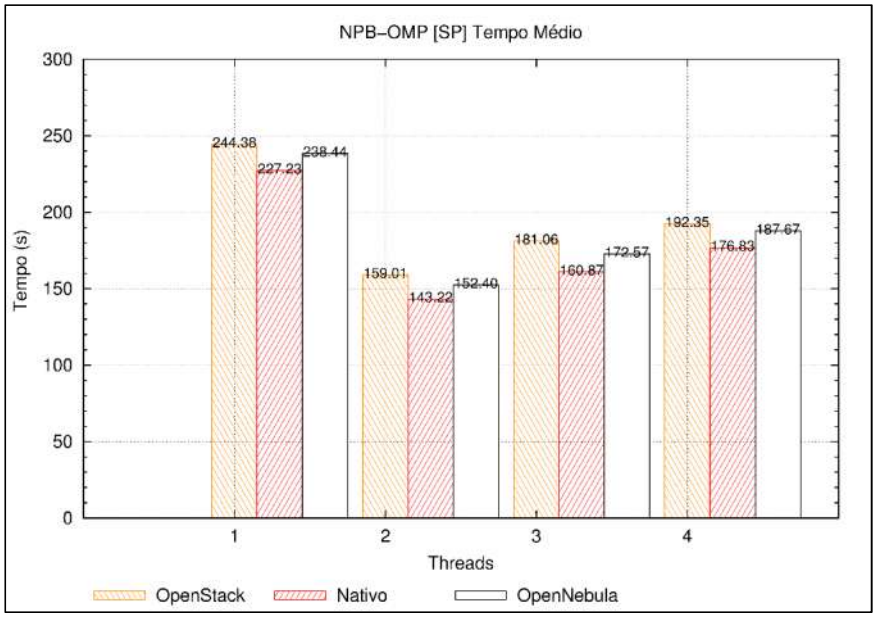
Figura 78: Comparação tempo médio do NPB-OMP [MG].



Fonte: Maron, Griebler. 2014.

Figura 79: Comparação speed-up e eficiência do NPB-OMP [MG]

A Figura 79 pode comprovar o desempenho inferior do ambiente OpenStack na execução do programa MG, a utilização dos recursos a partir das execuções com 2 *threads* se manteve praticamente os mesmos, resultando em um desempenho inferior se comparado aos outros ambientes. Mas a ferramenta OpenNebula e o ambiente Nativo, obtiveram os mesmos valores nas execuções de 3 e 4 *threads.*



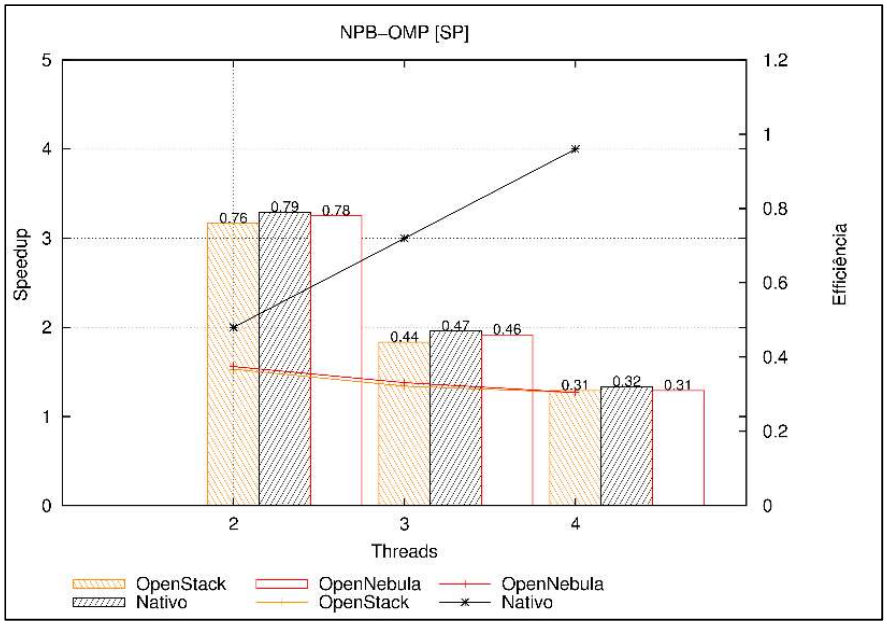
Fonte: Maron, Griebler. 2014

Figura 80: Comparação tempo médio NPB-OMP [SP].

A Figura 80 mostra a média de tempo de execução do programa SP nos ambientes OpenStack, OpenNebula e Nativo. Observa-se que existem ganhos e perdas irregulares no decorrer das execuções. Pois, ao passar a ser executado com 2 *threads* os tempos exibem um ganho. Mas, ao passar para 3 e posteriormente a 4, as médias aumentam em todos os ambientes. Mas ambas as ferramentas se mantêm próximas no resultado.

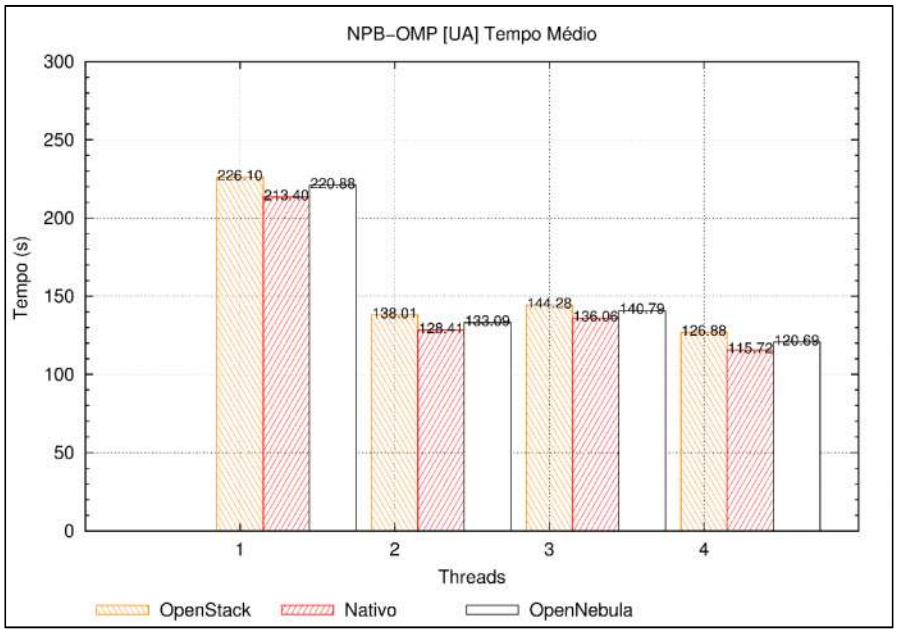
Quando executado, o SP aplica cargas no processador, porém, realizam um nível considerável de acesso a memória RAM. Ainda por se tratar de um programa que paraleliza seu desempenho localmente, durante as execuções percebe-se alguns acessos a rede. Talvez por isso, que a ferramenta OpenNebula alcançou um resultado melhor que a OpenStack.

A Figura 81 mostra a real perda de eficiência e *speed-up* durante a execução do programa SP. Os valores se mantém próximos entre os ambientes, mas com o decorrer do aumento de *threads* os valores vão diminuindo, resultando diretamente no desempenho das execuções do programa.



Fonte: Maron, Griebler. 2014

Figura 81: Comparação eficiência e speed-up do NPB-OMP [SP].



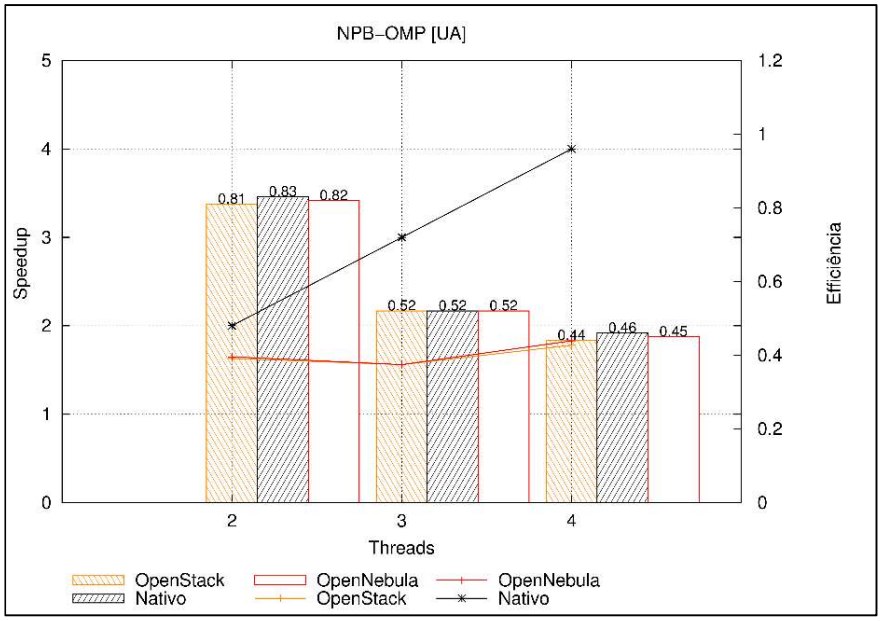
Fonte: Maron, Griebler. 2014

Figura 82: Comparação tempo médio do NPB-OMP [UA].

As execuções do programa UA da suíte NPB-OMP, mostram um gradual ganho no tempo de execução com 2 e 4 *threads* para o ambiente Nativo. Mas na ferramenta OpenNebula, aparentemente não existem ganhos significativos com decorrer do aumento das *threads*. Mas em todas as execuções, OpenStack e

OpenNebula, mantiveram praticamente a todo instante uma média semelhante no tempo de execução, e ainda próximas ao ambiente Nativo.

Isto mostra que ainda por se tratar de um ambiente virtual, o KVM mantém um desempenho perto do Nativo, gerenciando de forma compatível os recursos de memória, aonde o UA concentra suas cargas de trabalho.



Fonte: Maron, Griebler. 2014

Figura 83: Comparativo speed-up e eficiência NPB-OMP [UA].

Como mostra a Figura 83, a eficiência em ambos os ambientes se manteve praticamente idênticas em cada execução das determinadas *threads*. Mas, sofrem um declínio considerável ao passar a executar com 3, e praticamente mantém o mesmo nível quando passam para 4. Mostrando a irregular eficiência na utilização dos recursos para o processamento.

3.4 TESTE DE HIPÓTESES

Nas seções anteriores foram apresentados vários gráficos que demonstravam o comportamento do ambiente computacional, e também o comportamento das aplicações paralelas usadas na pesquisa. Além disso, exibindo uma comparação entre os ambientes OpenStack, OpenNebula e Nativo. Dos

resultados obtidos, foi realizado um mapeamento da execução de cada teste, com o intuito de justificar os resultados obtidos de cada ambiente em cada um dos testes.

Entretanto, nos gráficos é possível ver diferenças entre os ambientes OpenStack e OpenNebula, no que diz respeito a sua infraestrutura e aplicações paralelas. Contudo, essa diferença existente nos resultados das execuções deve ser analisada estatisticamente para apurar se realmente se torna significante nos resultados.

Para isso, o *software* SPSS (*Statistical Package for the Social Sciences*) é frequentemente usado em experimentos de *software* e análise estatísticas para os testes de hipótese. Neste trabalho, será usado o grau de significância (Sig.) para identificar se as amostras obtidas nos resultados são significativamente diferentes (FIELD. 2009).

Os valores repassados para o SPSS para serem analisados foram: os tempos de execução de cada programa das suítes NPB-OMP e NPB-MPI. Os resultados da largura de banda dos testes com o IPERF, repassados em Mbits/s. Os resultados dos valores de *throughput* dos testes com o STREAM, repassados em MBytes/s. Os valores de KFLOPS, executado pelo *benchmark* LINPACK. E os resultados em KBytes, dos testes executados pelo IOzone.

Segundo Triola (2005), para procedimentos experimentais como os envolvidos neste trabalho, usa-se uma margem de 95% de confiabilidade durante a análise estatística com o SPSS. Isto significa que, em uma comparação realizada pelo *software*, se o fator Sig for maior que 0,05, não torna possível afirmar que as médias amostrais são significativamente diferentes.

Durante a apresentação dos resultados com o SPSS, será adotado um esquema de legendas e símbolos para sinalizar qual obteve o melhor resultado. Para cor vermelha e símbolo β (Betha) identificará a ferramenta OpenStack como vantagem nos resultados. Para a ferramenta OpenNebula, será usada a cor azul e o símbolo Ω (Hom) como vantagem nos resultados. Para o ambiente Nativo não haverá representação de cores e símbolos. E quando o Sig. atingir a margem maior que 0,05,

será apresentado o símbolo Δ (Delta) na lateral do valor, identificando que não existe diferença significativa entre os ambintes.

3.4.1 **Primeira hipótese:**

O desempenho da infraestrutura (disco, rede, memória e processamento) virtual é significativamente afetado em relação ao ambiente Nativo‡

A hipótese descritiva foi formalizada para que pudéssemos realizar a análise com o SPSS, pois o intuito é avaliar se o ambiente Nativo é igual, ou, diferente que os ambientes OpenStack e OpenNebula. Sendo assim, para o SPSS as hipóteses foram tratadas da seguinte maneira:

H0: Ambiente Nativo == Ambiente Virtual (OpenStack e OpenNebula)

H1: Ambiente Nativo != Ambiente Virtual (OpenStack e OpenNebula)

O Quadro 5 representa a diferença encontrada durante os testes com o SPSS. Na coluna “Característica” está representado o componente testado. Nas colunas adjacentes (C3 e C4) estão a dispostos a comparação com cada ambiente (OpenStack, OpenNebula e Nativo). E na coluna Sig. estará representado os valores da variável Sig. respeitando a mesma sequência da disposição das colunas dos ambientes (C3 e C4).

‡ Modelos de serviços IaaS trabalha com ferramentas de virtualização. Neste contexto, o ambiente real é composto por memória, disco, rede, e processamento sem a camada de virtualização implantada por ferramentas para este propósito. Sendo de forma genérica, o ambiente real é composto somente por camada física e camada de *software (S.O).*

Característica	Sig C3...|...C4	C3: OpenStack X Nativo (Diferença entre as médias)	C4: OpenNebula X Nativo (Diferença entre as médias)
Processador	0,000 | 0,000	3,45046E8	3,50751E8
Rede	0,000 | 0,000	28,07000	0,55250
Memória			
ADD	0,000 | 0,000	1,00449E7	7,27433E6
COPY	0,000 | 0,000	1,17791E7	8,58201E6
SCALE	0,000 | 0,000	1,30331E7	9,65217E5 (Ω)
TRIAD	0,000 | 0,000	1,08540E7	1,79163E7
Armazenamento			
WRITE	0,009 | 0,000	1,51111E5	3,07730E5
REWRITE	0,000 | 0,000	1,13147E5	2,96457E5
READ	0,031 | 0,000	1,94943E6	4,00799E6
REREAD	0,000 | 0,000	4,83874E5	1,12463E6

Fonte: Maron, Griebler. 2014

Quadro 05: Resultados dos testes estatísticos da infraestrutura.

No apêndice K deste trabalho estão os resultados das análises estatísticas do SPSS aplicado em cada ambiente, contando com os resultados de desvio padrão e cálculos das médias, as amostras pareadas e os testes efetivos dessas amostras.

Sendo assim, os resultados obtidos no Sig. comprovam a rejeição da hipótese nula (H0), pois os ambientes são significativamente diferentes, desta forma, assume a hipótese alternativa (H1).

3.4.2 **Segunda hipótese**

O desempenho de aplicações paralelas (*speed-up,* eficiência, tempo de execução) em um ambiente de nuvem não é significativamente afetado em relação ao ambiente Nativo[§].

[§] Modelos de serviços IaaS trabalha com ferramentas de virtualização. Neste contexto, o ambiente real é composto por memória, disco, rede, e processamento sem a camada de virtualização implantada por ferramentas para este propósito. Sendo de forma genérica, o ambiente real é composto somente por camada física e camada de *software (S.O).*

Portanto para o teste desta hipótese, novamente formalizou-se esta hipótese dividindo elas entre as suítes NPB-MPI e NBP-OMP, para assim poder verificar a significância entre os resultados do ambiente Nativo e do Virtual.

3.4.2.1 NPB-OMP

Os resultados da suíte NPB-MPI foram tratados pelo SPSS da seguinte maneira:

H0: NPB-OMP Nativo == NPB-OMP Nuvem (OpenStack, OpenNebula)

H1: NPB-OMP != NPB-OMP Nuvem (OpenStack, OpenNebula)

A Tabela 02 mostra a diferença dos resultados da análise estatística com as execuções do NPB-OMP. Na coluna "Programa-*thread*" está representado o programa da suíte NPB-OMP testado, seguido pelo número de *threads*. Nas colunas adjacentes (3 e 3) estão a dispostos a comparação com cada ambiente (OpenStack OpenNebula e Nativo). E na coluna Sig. estará representado os valores da variável Sig. respeitando a mesma sequência da disposição dos ambientes (2 e 3).

Tabela 02: Diferença dos resultados estatísticos na análise SPSS dos resultados da suíte NPB-OMP

Programa-*thread*	Sig. C3...|.....C4	C3: OpenStack X Nativo	C4: OpenNebula X Nativo
BT-1	0,000 | 0,000	14,15225	10,48575
BT-2	0,000 | 0,000	10,72450	6,22575
BT-3	0,000 | 0,000	19,28850	7,32375
BT-4	0,000 | 0,000	26,45400	9,00775
SP-1	0,000 | 0,000	17,14975	11,21100
SP-2	0,000 | 0,000	15,79350	9,18675

Programa-*thread*	Sig. C3...|.....C4	C3: OpenStack X Nativo	C4: OpenNebula X Nativo
SP-3	0,000 | 0,000	20,19600	11,70475
SP-4	0,000 | 0,000	15,52625	10,84600
UA-1	0,000 | 0,000	12,69650	7,47125
UA-2	0,000 | 0,000	9,59825	4,68375
UA-3	0,000 | 0,000	8,22625	4,73025
UA-4	0,000 | 0,000	11,15375	4,96825
CG-1	0,005 | 0,003	0,57925	84175
CG-2	0,000 | 0,000	4,96425	2,78300
CG-3	0,000 | 0,000	4,26000	2,58125
CG-4	0,000 | 0,000	6,18000	3,80450
EP-1	0,000 | 0,000	1,91000	1,28625
EP-2	0,000 | 0,000	0,66500	0,33600
EP-3	0,007 | 0,000	0,13750	0,19050
EP-4	0,000 | 0,000	1,13700	0,47075
LU-1	0,000 | 0,000	22,32075	15,74600
LU-2	0,000 | 0,000	20,85550	14,43675
LU-3	0,000 | 0,000	24,42000	15,44800
LU-4	0,000 | 0,000	28,07950	18,76625
MG-1	0,000 | 0,000	0,84575	0,56875
MG-2	0,000 | 0,000	6,43075	0,38000
MG-3	0,000 | 0,000	1,15200	0,34700
MG-4	0,000 | 0,000	0,64475	0,28225
IS-1	0,000 | 0,000	0,74400	0,67625
IS-2	0,000 | 0,000	0,42100	0,36475
IS-3	0,000 | 0,000	0,96150	0,57625
IS-4	0,000 | 0,000	1,57625	1,56700

Fonte: Maron, Griebler. 2014

Os resultados da Tabela 02 confirmam a rejeição da hipótese nula (H0), pois os resultados são significativamente diferentes entre os ambientes Nativo e

OpenStack e Nativo e OpenNebula. Sendo assim, assume-se a hipótese alternativa (H1)

3.4.2.2 NPB-MPI

Os resultados da suíte NPB-MPI foram tratados pelo SPSS da seguinte maneira:

H0: NPB-MPI Nativo == NPB-MPI Nuvem (OpenStack e OpenNebula)

H1: NPB-MPI != NPB-MPI Nuvem (OpenStack e OpenNebula)

Na coluna “Programa-processos” está representado o programa da suíte NPB-MPI testado, seguido pelo número de processos durante a execução. Nas colunas adjacentes (2 e 3) estão a dispostos a comparação com cada ambiente (OpenStack OpenNebula e Nativo). E na coluna Sig. estará representado os valores da variável Sig. respeitando a mesma sequência da disposição dos ambientes (2 e 3).

Tabela 03: Diferença dos resultados estatísticos na análise SPSS dos resultados da suíte NPB-MPI

Programa-processos	Sig. C3...|.....C4	C3: OpenStack X Nativo	C4: OpenNebula X Nativo
MG-1	0,000 | 0,000	,97033	0,70467
MG-2	0,000 | 0,000	5,98500	1,99867
MG-4	0,000 | 0,000	5,48767	1,78033
MG-8	0,000 | 0,000	3,19267	2,44833
MG-16	0,000 | 0,000	6,15800	5,77867
LU-1	0,000 | 0,000	23,74767	17,07367
LU-2	0,000 | 0,000	18,62833	15,22500
LU-4	0,000 | 0,000	20,17567	12,40667
LU-8	0,000 | 0,000	25,82133	34,42333
LU-16	0,000 | 0,000	45,44033	51,72933
IS-1	0,000 | 0,000	0,55200	0,46867

IS-2	0,000 | 0,000	12,61133	3,91500
IS-4	0,000 | 0,000	3,60900	1,08567 (Ω)
IS-8	0,000 | 0,000	5,21100	1,69500 (Ω)
IS-16	0,000 | 0,000	11,37633	14,66167
FT-1	0,062 (Δ) | 0,157 (Δ)	5,71300 (Δ)	1,42100(Δ)
FT-2	0,000 | 0,000	107,31433	28,81467
FT-4	0,000 | 0,000	99,10100	24,37500
FT-8	0,000 | 0,000	62,56133	56,46733
FT-16	0,000 | 0,000	105,20933	55,01200
SP-1	0,000 | 0,000	43,75333	34,36600
SP-4	0,000 | 0,000	51,42033	2,47033
SP-9	0,000 | 0,000	137,63967	38,83700
SP-16	0,000 | 0,000	116,60733	57,46300
CG-1	0,000 | 0,000	13,39567	10,78400
CG-2	0,000 | 0,000	33,95300	11,99367
CG-4	0,000 | 0,000	31,91900	2,65500
CG-8	0,000 | 0,000	39,34733	53,08267
CG-16	0,000 | 0,000	172,77633	46,50267
EP-1	0,000 | 0,002	1,14333	0,56433
EP-2	0,000 | 0,000	0,53067	0,41300
EP-4	0,000 | 0,094(Δ)	0,27000	0,07900(Δ)
EP-8	0,000 | 0,000	0,30833	0,08467
EP-9	0,000 | 0,529 (Δ)	0,27900	0,02067(Δ)
EP-16	0,000 | 0,000	0,32233	0,14533
BT-1	0,000 | 0,000	17,01800	12,34667
BT-4	0,000 | 0,000	27,26000	1,23433 (Ω)
BT-9	0,000 | 0,000	81,93033	21,81567
BT-16	0,000 | 0,000	63,11000	32,37967

Fonte: Maron, Griebler. 2014

Portanto, a Tabela 03 comprova a rejeição da hipótese H0, com exceção de casos específicos como nas execuções do EP-9 e EP4, na comparação com o

OpenNebula vs Nativo, e FT-1 OpenStack vs Nativo. Dessa forma não é possível considerar que os resultados nestes casos foram significativamente diferentes.

3.4.2.3 Terceira hipótese

A terceira hipótese busca tornar evidente se existe alguma diferença nos resultados das aplicações paralelas executadas nas ferramentas OpenStack e OpenNebula, e na própria infraestrutura composta por essas ferramentas. Sendo assim, a terceira hipótese fica:

O desempenho obtido nas aplicações paralelas e na infraestrutura não é significativamente diferente entre as ferramentas OpenStack e OpenNebula na nuvem.

3.4.2.3.1 Análise da Infraestrutura

Para este primeiro momento, as hipóteses analisadas pelo SPSS foram representadas das seguintes maneira:

H0: Infraestrutura OpenStack == Infraestrutura da OpenNebula

H1: Infraestrutura OpenStack != Infraestrutura OpenNebula

O Quadro 06 está descrito dessa forma: coluna “Característica” está representado o componente testado pelo *benchmark*. Nas colunas adjacentes estão representados os valores Sig. obtidos pelo SPSS. E posterior a isto, a coluna de comparação com cada ambiente (OpenStack e OpenNebula).

Característica	Sig.	OpenStack X OpenNebula
Processador	0,000	5,70420E6 (Ω)
Rede	0,000	27,51750 (Ω)
Memória		
ADD	0,000	2,77061E6 (Ω)
COPY	0,000	3,19706E6 (Ω)
SCALE	0,000	1,39983E7 (Ω)
TRIAD	0,000	7,06232E6 (β)
Armazenamento		
WRITE	0,002	1,56618E5 (β)
REWRITE	0,000	1,83311E5 (β)
READ	0,012	2,05856E6 (β)
REREAD	0,000	6,40755E5 (β)

Fonte: Maron, Griebler. 2014

Quadro 06: Resultados dos testes estatísticos da infraestrutura. OpenStack e OpenNebula.

No Apêndice K deste trabalho, se encontram as amostras estatísticas dos resultados dos SPSS, como também o teste efetivo destas amostras para definir seus valores, definido a margem de erro de cada cálculo e desvio padrão dos resultados obtidos nos testes de infraestrutura.

Sendo assim, através do Quadro 06 é possível confirmar a rejeição da hipótese H0, pois os valores Sig. afirmam que os resultados são significativamente diferentes entre as duas ferramentas. Com base nas médias, pode-se demarcar que quais situações uma ferramnte tece melhor desempenho.

3.4.2.3.2 Analise aplicações paralelas NPB-OMP

Para a análise das aplicações paralelas NPB-OMP, as seguintes hipóteses foram geradas para o SPSS:

H0: NPB-OMP OpenStack == NPB-OMP OpenNebula

H0: NPB-OMP OpenStack != NPB-OMP OpenNebula

A Tabela 04 representa os valores obtidos com a execução do SPSS. Na coluna “Programa-*threads*” está representado o programa da suíte NPB-OMP testado, seguido pelo número de *threads* durante a execução. Na coluna adjacente (2) estão representados os valores de Sig. E posteriormente, a coluna da comparação entre os ambientes OpenStack e OpenNebula.

Tabela 04: Diferença dos resultados estatísticos na análise do SPSS dos resultados da suíte NPB-OMP

Programa-*thread*	Sig.	OpenStack X OpenNebula
BT-1	0,000	3,66650 (Ω)
BT-2	0,000	4,49875 Ω)
BT-3	0,000	11,96475 Ω)
BT-4	0,000	17,44625 Ω)
SP-1	0,000	5,93875 (Ω)
SP-2	0,000	6,60675 (Ω)
SP-3	0,000	8,49125 (Ω)
SP-4	0,000	4,68025 (Ω)
UA-1	0,000	5,22525 (Ω)
UA-2	0,000	4,91450 (Ω)
UA-3	0,000	3,49600 (Ω)
UA-4	0,000	6,18550 (Ω)
CG-1	0,000	1,42100 (Ω)
CG-2	0,000	2,18125 (Ω)
CG-3	0,000	1,67875 (Ω)
CG-4	0,000	2,37550 (Ω)
EP-1	0,000	0,62375 (Ω)
EP-2	0,000	0,32900 (Ω)
EP-3	0,356(Δ)	0,05300(Δ)
EP-4	0,000	0,66625 (Ω)

Programa-*thread*	Sig.	OpenStack X OpenNebula
LU-1	0,000	6,57475(Ω)
LU-2	0,000	6,41875(Ω)
LU-3	0,000	8,97200(Ω)
LU-4	0,000	9,31325(Ω)
MG-1	0,000	0,27700(Ω)
MG-2	0,000	6,05075(Ω)
MG-3	0,000	0,80500(Ω)
MG-4	0,000	0,36250(Ω)
IS-1	0,000	0,06775 (Ω)
IS-2	0,000	0,05625 (Ω)
IS-3	0,000	0,38525 β
IS-4	0,094(Δ)	0,00925(Δ)

Fonte: Maron, Griebler. 2014

No Apêndice I estão mais detalhadas as amostras e testes estatísticos aplicado com os resultados da suíte NPB-OMP, mostrando as margens de erros, desvio padrão e cálculo de médias dos resultados estatísticos.

Sendo assim, os resultados dos valores do Sig. comprovam que a hipótese H0 foi rejeitada. Contudo, em apenas uma execução do programa IS-4 e EP-3, em que a hipótese H0 não é rejeitada.

3.4.2.3.3 Resultado das aplicações paralelas NPB-MPI

Para a análise das aplicações paralelas NPB-MPI, as seguintes hipóteses foram geradas para o SPSS:

H0: NPB-MPI OpenStack == NPB-MPI OpenNebula

H0: NPB-MPI OpenStack != NPB-MPI OpenNebula

Portanto, a Tabela 05 traz os resultados do SPSS. Na coluna “Programa-processos” está representado o programa da suíte NPB-MPI testado, seguido pelo

número de processos durante a execução. Nas colunas adjacentes (2) está representado os valores da variável Sig. E posteriormente, estão a dispostos a comparação com cada ambiente (OpenStack e OpenNebula).

Tabela 05: Diferença dos resultados estatísticos na análise SPSS dos resultados da suíte NPB-MPI

Programa-processos	Sig.	OpenStack X OpenNebula
MG-1	0,000	,26567(Ω)
MG-2	0,000	3,98633(Ω)
MG-4	0,000	3,70733(Ω)
MG-8	0,000	,74433(Ω)
MG-16	0,039	,37933(Ω)
LU-1	0,000	6,67400(Ω)
LU-2	0,000	3,40333(Ω)
LU-4	0,000	7,76900(Ω)
LU-8	0,000	-8,60200 β
LU-16	0,000	-6,28900 β
IS-1	0,000	,08333 (Ω)
IS-2	0,000	8,69633 (Ω)
IS-4	0,000	4,69467 (Ω)
IS-8	0,000	6,90600 (Ω)
IS-16	0,000	-3,28533 β
FT-1	0,114(Δ)	4,29200(Δ)
FT-2	0,000	78,49967 (Ω)
FT-4	0,000	74,72600(Ω)
FT-8	0,000	6,09400(Ω)
FT-16	0,000	50,19733(Ω)
SP-1	0,000	9,38733(Ω)
SP-4	0,000	48,95000 β
SP-9	0,000	98,80267(Ω)
SP-16	0,000	59,14433(Ω)

Programa-processos	Sig.	OpenStack X OpenNebula
CG-1	0,000	2,61167(Ω)
CG-2	0,000	21,95933(Ω)
CG-4	0,000	34,57400(Ω)
CG-8	0,000	-13,73533 β
CG-16	0,000	126,27367(Ω)
EP-1	0,001	,57900(Ω)
EP-2	0,151(Δ)	0,11767(Δ)
EP-4	0,000	,19100(Ω)
EP-8	0,000	,22367(Ω)
EP-9	0,000	-,29967 β
EP-16	0,000	,17700(Ω)
BT-1	0,000	4,67133 (Ω)
BT-4	0,000	28,49433(Ω)
BT-9	0,000	60,11467 β
BT-16	0,000	30,73033(Ω)

Fonte: Maron, Griebler. 2014

A Tabela 05 apresentou os resultados estatísticos da análise do SPSS. Informações sobre as amostras dos testes, e propriamente os testes aplicados nestas amostras com as margens de erros e desvio padrão, podem ser vistas no Apêndice I deste trabalho.

Portanto comprova-se pelos resultados da Tabela 05, que a hipótese H0 é rejeitada, pois existem diferenças significantes entre os dois ambientes OpenStack e OpenNebula. Porém, em casos isolados como nas execuções com o EP-2 e FT1 as hipóteses não podem ser descartadas, já que não existem diferenças significativas nos resultados.

Entende-se através da Tabela 05, que a ferramenta OpenNebula tem os melhores resultados com as aplicações paralelas no padrão MPI. Contudo, observa-se um ganho do OpenStack em algumas execuções com maior quantidade de processos.

CONCLUSÃO

A pesquisa realizada neste trabalho buscou avaliar a performance de ferramentas de computação em nuvem e compará-las ao ambiente Nativo, para assim definir se estas são significativamente diferentes. Junto à isto, comparar o resultado entre as próprias ferramentas de administração de nuvem, e buscar traçar um resultado estatísticos e técnico das diferenças entre as duas ferramentas.

O trabalho proporcionou uma vasta experiência com as ferramentas OpenStack e OpenNebula, além ainda, de aprofundar o entendimento sobre conceitos de computação em nuvem, virtualização, testes com *benchmarks* específicos, para avaliação de componentes da infraestrutura e aplicações para computação de alto desempenho, fazendo com que nossos objetivos fossem alcançados.

Com relação às hipóteses descritas no primeiro capítulo que motivaram esta pesquisa, a primeira afirma que o desempenho de uma infraestrutura virtual é significativamente afetado em relação ao Nativo. Esta hipótese não é totalmente verdadeira, pois como demonstra o Quadro 06, no desempenho da função *SCALE*, o resultado favorece o ambiente OpenNebula que é significativamente diferente do Nativo. E o restante dos resultados favoreceram o ambiente Nativo.

A segunda hipótese afirma que o desempenho das aplicações paralelas executadas em um ambiente de nuvem, não é afetado significativamente com relação aos resultados de um ambiente Nativo. Com a execução de *benchmarks* e a avaliação estatística, os resultados mostram que essa hipótese não é totalmente verdadeira. Pois em alguns casos como os do EP-9 e EP-4 do padrão MPI, os resultados entre ambiente Nativo vs OpenNebula e Nativo vs OpenStack não se mostram

significativamente diferente. Já nas execuções do BT-4, IS-4 e IS-8 os resultados estatisticamente foram diferentes ao favor da ferramenta OpenNebula, alcançando um desempenho superior ao ambiente Nativo. E ainda nas execuções dos programas EP-4 e EP-16 na comparação do Nativo vs OpenNebula, os resultados não se mostram significativamente diferentes, valores estes que podem ser comprovados com a Tabela 03. E para as execuções dos programas do padrão OMP, os resultados se mostraram significativamente diferentes, favorável ao ambiente Nativo.

A terceira hipótese busca comparar o desempenho obtido nas aplicações paralelas e infraestrutura. Sendo assim ela afirma que o desempenho não é afetado entre as ferramentas de administração de nuvem OpenStack e OpenNebula. Esta hipótese não é totalmente verdadeira pois com relação a infraestrutura, os resultados nos mostram claramente nas Figuras 36, 37,38, 39, e no Quadro 06 especificamente, que os resultados são significativamente diferentes entre as duas ferramentas. Isto devido ao modo de implantação da ferramenta OpenNebula através de discos LVM em unidades NFS na rede, compartilhadas entre os nodos, onde os resultados mostraram que seu desempenho é reduzido, ficando muito aquém da ferramenta OpenStack que implementa seus discos localmente nos formatos LVM.

No desempenho de processador, e rede, as Figuras 30 e 31, além do Quadro 06 nos provar estatisticamente as diferenças significativas no desempenho destes componentes, favorecendo a ferramenta OpenNebula. Acredita-se que o desempenho inferior da ferramenta OpenStack nestes quesitos, dá-se devido aos seus componentes que gerenciam estes recursos, como OpenVSwitch e Neutron para controle da rede, e *Nova-compute-KVM*, um componente específico da ferramenta para controle da camada de virtualização das instâncias. Onde acredita-se ainda que tenha afetado o desempenho da memória RAM virtual.

É importante descatar que os resultados tiveram um propósisto de comparar o desempenho entre as ferramentas de administração de nuvem (OpenStack e OpenNebula). O desempenho da infraestrutura não podem ser levandos em conta para a implantação em um ambiente empresarial, tanto que principalmente os resultados de disco da ferramenta OpenNebula tiveram um resultado degradante, já que o modo de implantação da ferramenta e a infraestrutura

de rede não ser *gigabit,* influenciou na grande disparidade de desempenho entre as ferramentas neste quesito. Além disso, os desempenhos de speed-up e eficiência na suíte NPB-MPI foram fortemente afetados em ambas as ferramentas usando a rede *megabit*.

Devido ao desempenho superior da ferramenta OpenNebula nos testes de rede, processador e memória, a Tabela 04 e 05 comprova estatisticamente que os resultados das execuções das aplicações paralelas tornam a ferramenta OpenNebula muito mais indicada para este fim que a ferramenta OpenStack. Tornando a hipótese parcialmente falsa, pois as execuções do padrão MPI EP-2 e FT-1 não se mostram significativamente diferentes. Já nas execuções dos programas no padrão MPI BT-9, EP-9, CG-9, SP-4, IS-16, LU-8 e LU-16 se mostram significativamente diferentes, favorecendo a ferramenta OpenStack.

No padrão OMP, as execuções EP-3 e IS-4 não se mostram significativamente diferentes entre os ambientes OpenStack vs OpenNebula. E em execuções como IS-3 os resultados se tornam significativamente diferentes, favorecendo o ambiente OpenStack.

Com todos estes resultados, pretendemos ainda realizar outros trabalhos futuros com a escrita de artigos com os resultados desta pesquisa. De primeira mão, pretendemos realizar novos testes com rede *gigabit*, para determinar resultados mais satisfatórios com a as aplicações paralelas. Além disso aprofundar outros tipos de virtualizadores.

Almejamos ainda executar testes de infraestrutura com diferentes *benchmarks*. Avaliar o desempenho de aplicações de alto desempenho relacionando os resultados com o consumo energético da infraestrutura no geral. Aprofundar mais detalhes de configurações dos componentes de cada ferramenta, visando adquirir um melhor desempenho. E ainda estudar, implantar e comparar os resultados obtidos nestas ferramentas com outras existentes no mercado de computação em nuvem. Buscando ainda com todos os resultados destes trabalhos poder contribuir com a comunidade acadêmica com a escrita artigos.

REFERÊNCIAS

ABRAMS, Rhonda. 2011**. Bringing the Cloud Down to Earth**. Palo Alto : PlanninShop, 2011. p. 168.

ALECRIM, Emerson . 2013**. Cluster: Conceito e características**. *InfoWester.* Março 22, 2013. Disponpivel em: <http://www.infowester.com/cluster.php#> Acessado em: 01 de abril 2014.

BATTISTI, Julio; SANTANA, Fabio. 2009**. Windows Server 2008: Guia de Estudo Completo**. s.l. : Nova Terra, 2009.

Thomé, Bruna; Hentges, Eduardo; Griebler, Dalvan. **Computação em Nuvem: Análise Comparativa de Ferramentas Open Source para IaaS**. Anais da 11ª Escola Regional de Redes de Computadores, 2013.

BONNIE++. 2014. **Introduction**. Disponível em :<http://www.coker.com.au/bonnie++/readme.html> Acessado em 30 Junho 2014

BURNS, Bill. 2010**. KVM in Red Hat Enterprise**. Junho 23, 2010. Disponível em: <http://www.redhat.com/promo/summit/2010/presentations/summit/whats-next/wed/bburns-1130-kvm/kvm-in-rhel6.pdf> Acessado em 21 de Fevereiro 2014.]

CHEVANCE, René J. 2005**. Server Architectures: Multiprocessors, Clusters Parallel Systems, Web Servers, and Storages Solutions**. Burlington : Elsevier, 2005.

CISCO. 2014**. Cloud: What an enterprise must know**. Disponível em : http://www.cisco.com/en/US/solutions/collateral/ns340/ns517/ns224/ns836/ns976/white_paper_c11-617239.html Acessado em: 17 Dezembro de 2013.

CORRADI, Antonio, FANELLI, Mario; FOSCHINI, Luca. 2012. **VM consolidation: A real case based on OpenStack Cloud**. s.l. : Elsevier, 2012. pp. 1-10.

DELL, Michael. 2011**.** Rhonda ABRAMS. **Bringing the cloud down to Earth**. Palo Alto : PlanningShop, 2011.

DONGARRA, Jack. 2007. **FAQ on the Linpack Benchmark and Top500**. Disponpivel em: <http://www.netlib.org/utk/people/JackDongarra/faq-linpack.html#_Toc27885709>. Acessado em 01 de Abril de 2014.

DONGARRA, Jack, LUSZCZEK, Piotr and PETITET, Antonie. 2001. **The Linpack Benchmark: Past, Present and future**. 2001.

EVANGELINOS, Constantinos and HILL, Chris. 2008. **Cloud Computing for parallel Scientific HPC Applications: Feasibility of running Coupled Atmosphere-Ocean Climate Models on Amazon's EC2**. 1-6. 2008.

FIELD, Andy. **Discovering Statístics Using SPSS.** London: SAGE, 2009, 3º Edição. pp. 1-854.

GALISTEU, Renato. 2013**. USP migra para a cloud computing privada**. Disponível em <http://www.itforum365.com.br/noticias/detalhe/3778/usp-migra-para-a-cloud-computing-privada.>. Acessado em Junho 28 de Junho de 2014

GANGLIA. 2014. **What is Ganglia?**. Disponível em <http://ganglia.sourceforge.net/> . Acessado em: 02 de Julho 2014.

GONÇALVES, Charles Ferreira. 2007. **Métricas de Desempenho**. Disponível em <http://homepages.dcc.ufmg.br/~charles/paralela/resenhas_SDP/node5.html> Acessado em: 01 Março 2014.

GOOGLE. 2013**. Google Cloud Platform**. Disponível em: <https://cloud.google.com/>. Acessado em: 30 de Março de 2014.

GOTO, Yasunori. 2011. **Kernel based Virtual Machine Techonology**. *FUJITSU Sci. Tech.* 2011, Vol. 47, pp. 362-368.

GOWDA, Bhaskar. 2009. **A peek into Extended Page Tables**. Disponível em : <https://communities.intel.com/community/itpeernetwork/datastack/blog/2009/06/02/a-peek-into-extended-page-tables>. Acessado em: 02 de Março de 2014.

GRIBLER, Dalvan.

GUPTA, Abhishek and MILOJICIC, Dejan. **Evalution of HPC Applications on Cloud.** *Open Cirrus Summit (OCS).* 6ª, 2011, pp. 22-26.

HWANG, Jinho, et al. **A Component-Based Performance Comparison of Four Hypervisors**. 2013. pp. 1-8.

IBGE**. Pesquisa sobre o Uso das Tecnologias de Informação e Comunicação nas Empresas 2010**. IBGE - Institutos Brasileiro de Geografia e Estatísticas. Disponível em : <http://www.ibge.gov.br/home/estatistica/economia/tic_empresas/2010/default.shtm> . Acessado em: 07 Dezembro 2013.

INTEL. **A Nuven Educacional: A Educação Disponibilizada como um Serviço**. 2010.

IOSUP, Alenxadru, PRODAN, Radu; EPEMA, Dick. **IaaS Cloud Benchmarking: Approaches, Challenges, and Experience.** 2013. p. 8.

IOZONE. **IOZONE File System Benchmark.** 2006. Disponível em: <http://www.iozone.org>. Acessado em: 16 de Abril de 2014

IPERF. **What is Iperf?**. 2014. Disponpivel em: <http://iperf.fr/> Acessado em: 30 Junho de 2014

JACKSON, Kevin. **OpenStack Cloud Computing Cookbook**. Birmingham: Packt Publishing, 2012. pp. 1-318. 978-1-84951-732-4.

JACQUET, Alexandre; CAVASSANA, Henrique. **Computação nas nuvens: Perfil das empresas que adotaram Cloud Computing**. FIAP. São Paulo : s.n., 2012.

KOLYSHKIN, Kirill. **Virtualization in Linux**. 2006.

LANDIS, Cary; BLACHARSKI, Dan. 2013**. Cloud Computing Made Easy**. 2013.

LIBVIRT**. Libvirt FAQ**. 2013. Disponível em: <http://wiki.libvirt.org/page/FAQ>. Acessado em: 02 de Março de 2014.

Linux KVM. **Kernel Based Virtual Machine**. 2013. Disponpivel em: <http://www.linux-kvm.org/page/Main_Page> Acessado em: 21 Fevereiro de 2014.

LOWE, Scotte. **Calculate IOPS in a storage array**. 2010. Disponível em: <http://www.techrepublic.com/blog/the-enterprise-cloud/calculate-iops-in-a-storage-array/> Disponpivel em: 30 de Março de 2014.

MAILLÉ, Eric; MENNECIER, René-Francois. **VMware vSphere 5: Building a Virtual Datacenter**. Crawfordsville : VMware Press, 2013. 13:978-0-321-83221-4.

MARINESCU, Dan C. **Cloud Computing: Theory and Practice**. Waltham : Elsevier, 2013. 978-0-12404-627-6.

MATHEWS, Jeanna, et al. **Executando o Xen: Um Guia Prático para a Arte da Virtualização**. Traduzido por: Eveline Machado MACHADO; Thaís Cristina CASSON. Rio de Janeiro : Alta Books, 2009. 978-85-7608-317-7.

MATTESON, Scott. **How cloud computing will impact the on premise data center**. Tech Republic. 2013. Disponível em: <http://www.techrepublic.com/blog/the-enterprise-cloud/how-cloud-computing-will-impact-the-on-premise-data-center/>. Acessado em: 07 de Dezembro de 2013.

MCCALPIN, Jhon. **The Stream Benchmark.** 1996. Disponível em: <http://www.cs.virginia.edu/~mccalpin/papers/bandwidth/node1.html> Acessado em: 18 de Abril de 2014

MELO, Rosangela, LIBERALQUINO, Diego; SANTOS, Rosiberto**. Geração e modelagem de carga de trabalho (Workloads)**. 2014. Disponpivel em: <www.modcs.org/wp-content/uploads/2008/05/TYPES-OF-WORKLOADS-1.10.pptx>. Acessado em: 01 de Abril de 2014.

MERIAT, Vitor. **Modelos de Seviço na nuvem: IaaS, PaaS e Saas**. 2011. Disponível em: <http://vitormeriat.com.br/2011/07/08/modelos-de-servio-na-nuvem-iaas-paas-e-saas/>. Acessado em 01 de Abril de 2014

MICROSOFT. **What Is Windows Azure.** 2013. Disponpivel em:<http://www.windowsazure.com/pt-br/overview/what-is-windows-azure/> Acessado em: 24 de Março de 2014

NASA. **NAS Parallel Benchmarks**. 2012. Disponpivel em: <https://www.nas.nasa.gov/publications/npb.html>. Acessado em: 01 de Julho de 2014.

NETPIPE**. A Network Protocol Independent Performance Evaluator**. 2009. Disponível em: <http://www.scl.ameslab.gov/netpipe/>. Acessado em: 30 Junho de 2014.

NOVATTE. **How to calculate peak theoretical performance of a CPU-BASED HPC system.** 2013. Dísponível em: <http://www.novatte.com/our-blog/197-how-to-calculate-peak-theoretical-performance-of-a-cpu-based-hpc-system>. Acessado em: Março 03, 2014.

OLTPBENCHMARK. **Main Page**. 2014. Disponível em: <http://oltpbenchmark.com/wiki/index.php?title=Main_Page>. Acessado em: 10 de Abril de 2014

OPENSTACK [A]. **OpenStack Roadmap**. 2014. Disponível em: <https://www.openstack.org/software/roadmap/>. Acessado em: 20 de Maio de 2014.

OPENSTACK [B]**. Open vSwitch concepts**. 2014. Disponpivel em: <http://docs.openstack.org/havana/install-guide/install/apt/content/concepts-neutron.openvswitch.html>. Acessado em: 28 de Junho de 2014.

OPENSTACK [C]. **Chapter 3: Underlying Techonologies.** 2014. Disponível em: <http://docs.openstack.org/grizzly/openstack-compute/install/apt/content/externals.html> Acessado em 28 de Junho de 2014

OPENSTACK [D]. **Welcome to Glance's documentation**. 2014. Disponível em: <http://docs.openstack.org/developer/glance/> Acessado em: 29 Junho de 2014.

OPENSTACK [E]**. Trove**. 2014. Disponível em: <https://wiki.openstack.org/wiki/Trove>. Acessado em: 29 de Junho de 2014.

OPENSTACK [F]**. OpenStack Shared Services**. 2014. Disponível em: <https://www.openstack.org/software/openstack-shared-services/> Acessado em: 29 de Junho de 2014.

PARHAMI, Behrooz. **Introduction to Parallel Processing**. Santa Barbara : Kluwer, 2002.

PEPPLE, Ken. Deploying OpenStack. 1ª ed. Sebastopol : O'Reilly, 2011. pp. 1-86. 978-1-449-31105-6.

QEMU. **QEMU: Open Souce Processor Emulator**. 2013. Disponível em: <http://wiki.qemu.org/Main_Page>. Acessado em: 24 de Março de 2014.

REESE, George**. Cloud Application Architectures:Building Applications and Infrastructure in the Cloud**. Sebastopol : O'Reilly, 2009. p. 206. 978-0-596-15636-7.

REGOLA, Nathan; DUCON, Jean Christophe**. Recommendations for Virtualization Technologies in High Performance Computing**. Indianapolis : IEEE, 2010, pp. 1-8.

ROCHA, Ricardo**. Programação Paralela e Distribuída - Métricas de Desempenho***.* 2007.

SAVILL, John. **Microsoft Virtualization Secrets***.* Indianapolis : Wiley, 2012. p. 554. 978-1-118-29316-4.

SOFTWARE LIVRE BRASIL. **Página Inicial da Comunidade Open Source**.2013. Disponível em: <http://softwarelivre.org/open-source-codigo-aberto>. Acessado em: 07 deDezembro de 2013.

SOSINSKY, Barrie. **Cloud Computing Bible***.* Indianapolis : Wiley, 2011. p. 473. 978-0-470-90356-8.

SOTO, Julio Alba. **OpenNbula: Implantação de uma nuvem privada e orquestração das máquinas virtuais no paradigma da Computação em Nuvem***.* Ceará : s.n., 2011.

SPEC. **The Workload for the SPECweb96 Benchmark**. 2003. Disponível em: <http://www.spec.org/osg/web96/workload.html>. Acessado em: 20 deAbril de 2014.

SUN MICROSYSTEM. **Introduction to Cloud Computing Architecture***.* 2009.

TANENBAUM, Andrew S**. Computer Networks***.* Tradução: D. de Souza VANDENBERG. 4ª. Amsterdam : Campus, 2003. p. 232.

TAURION, Cezar. **Cloud Computing: Computação em nuvem - Transformanco o mundo da Tecnologia da Informação***.* Rio de Janeiro : Basport, 2009. p. 188. 978-85-7542-423-8.

TECHTERMS. **FLOPS**. 2009. Disponível em: <http://www.techterms.com/definition/flops>. Acessado em: 04 de Abril de 2014.

THOMÉ, Bruna Roberta, HENTGES, Eduardo Luís; GRIEBLER, Dalvan. **Análise e comparação de ferramentas open source de computação em nuvem para o modelo de serviço IaaS.** SETREM. Três de Maio : s.n., 2013. p. 194.

TORALDO, Giovani**. OpenNebula 3: Cloud Computing**. Birmingham : Packt, 2013. 978-1-84951-746-1.

VERDI, Fábio Luciano**. Cloud Computing, Data Centers e Governo: desafios e oportunidades**. Workshop de Computação Aplicada em Governo Eletrônica (WCGE). Julho 2010, p. 37.

VIRT-MANAGER. 2013**. Virtual Machine Manager**. 2013. Disponível em: <http://virt-manager.org/>. Acessado em 02 de Março de 2014.

VMWARE. **VMware ESX e VMware ESXi**. 2009. Disponível em: <http://www.vmware.com/files/br/pdf/products/VMW_09Q1_BRO_ESX_ESXi_BR_A4_P6_R2.pdf>. Acessado em: 17 de Dezembro de 2013.

WIKIMEDIA. **Wikimedia Grid Report**. 2014. Disponpivel em: <https://ganglia.wikimedia.org/latest/?r=hour&cs=&ce=&s=by+name&c=&tab=m&vn=&hide-hf=false>. Acessado em: 02 de Julho de 2014.

XAVIER [A], Miguel G., et al. **Performance Evalution of Container-based Virtualization for High Performance Computing Environments**. Euromicro International Conference on Parallel, Distributed, and Network-Based Processing. 2013, 21º, p. 8.

XAVIER [B], Miguel G., et al**. Towards better manageability of database clusters on cloud computing platforms.** 2013. p. 6.

XAVIER [C], Miguel G., NEVES, Marcelo V.; DE ROSE, Cesar A. F.; **A Performance Comparison of Container-based Virtualization Systems for MapReduce Clusters**. 2013. p. 8.

XEN Project. **History**. 2013. Disponível em: <http://www.xenproject.org/about/history.html>. Acessado em: 27 deDezembro de 2013.

ZIA, Ashraf and KHAN, Muhammad Naeem Ahmad. **Identifying Key Challenges in Performance Issues in Cloud Computing**. I.J Modern Education and Computer Science. 2012, pp. 59-68.

APÊNDICES

Apêndice A: Instalação e configuração do OpenStack Havana

Apêndice B: Instalação e configuração do OpenNebula

Apêndice C: Script para execução do *benchmark* NPB-MPI

Apêndice C: Script para execução do *benchmark* NPB-OMP

Apêndice E: Script para execução dos *benchmarks* de avaliação do ambiente

Apêndice F: Logs dos *benchmarks* do ambiente.

Apêndice G: Logs dos *benchmarks* das aplicações paralelas

Apêndice H: Artigo ERAD 2014

Apêndice I: Testes estatístico das aplicações paralelas – NPB-OMP

Apêndice J: Testes estatístico das aplicações paralelas – NPB-MPI

Apêndice K: Testes estatísticos para infraestrutura.

APÊNDICE A. INSTALAÇÃO E CONFIGURAÇÃO DO OPENSTACK HAVANA

Nesta seção serão detalhados os comandos e configurações utilizados no andamento da instalação da ferramenta OpenNebula Havana em todos os nodos da infraestrutura. Para realizar estes procedimentos, foram usados documentos oficiais da distribuição OpenStack Havana.

A.1 Preparação e configuração do frontend

Na infraestrutura foi utilizado o sistema operacional Ubuntu 12.04. Com o sistema operacional instalado, adiciona-se os repositórios para a ferramenta OpenStack da versão Havana, utilizando os comandos a seguir comandos:

```
apt-get install ubuntu-cloud-keyring python-software-properties
software-properties-common python-keyring

echo deb http://ubuntu-cloud.archive.canonical.com/ubuntu precise-
proposed/havana main >> /etc/apt/sources.list.d/havana.list
```

Realizado a inserção dos repositórios, é necessário instalar as atualizações do sistema usando os seguintes comandos:

```
apt-get -y update && apt-get -y upgrade && apt-get -y dist-upgrade
```

A.1.1 Instalação do MySQL e RabbitMQ

Instalação do gerenciador do banco de dados MySQL e configurando as permissões de acesso ao banco. Instalação do *middleware* RabbitMQ. E instalação do protocolo de sincronização de relógios.

Instação do MySQL:

```
apt-get install -y mysql-server python-mysqldb

sed -i 's/127.0.0.1/0.0.0.0/g' /etc/mysql/my.cnf
service mysql restart
```

Instalação do RabbitMQ:

```
apt-get install -y rabbitmq-server
```

Instalação do *Network Protocol Time:*

```
apt-get install -y ntp
```

A.1.2 Criação dos componentes no banco de dados usando o MySQL

Cada componente da ferramenta OpenStack necessita de acesso ao banco de dados. Para isso, foi utilizado um *script* a seguir para automatizar essa criação de tabelas e usuários no bando de dados.

```
#!/bin/sh
mysql -u root -p << EOF
CREATE DATABASE keystone;
GRANT ALL ON keystone.* TO 'keystone'@'%' IDENTIFIED BY
'password_access';
GRANT ALL ON keystone.* TO 'keystone'@'localhost' IDENTIFIED
BY 'password_access';
GRANT ALL ON keystone.* TO 'keystone'@'10.10.10.10' IDENTIFIED
BY 'password_access';
GRANT ALL ON keystone.* TO 'keystone'@'192.168.1.10'
IDENTIFIED BY 'password_access';
FLUSH PRIVILEGES;
CREATE DATABASE glance;
GRANT ALL ON glance.* TO 'glance'@'%' IDENTIFIED BY
'password_access';
GRANT ALL ON glance.* TO 'glance'@'localhost' IDENTIFIED BY
'password_access';
GRANT ALL ON glance.* TO 'glance'@'10.10.10.10' IDENTIFIED BY
'password_access';
GRANT ALL ON glance.* TO 'glance'@'192.168.1.10' IDENTIFIED
BY 'password_access';
FLUSH PRIVILEGES;
```

Continuação do *script.*

```
CREATE DATABASE neutron;
GRANT ALL ON neutron.* TO 'neutron'@'%' IDENTIFIED BY
'password_access';
GRANT ALL ON neutron.* TO 'neutron'@'localhost' IDENTIFIED BY
'password_access';
GRANT ALL ON neutron.* TO 'neutron'@'10.10.10.10' IDENTIFIED
BY 'password_access';
GRANT ALL ON neutron.* TO 'neutron'@'192.168.1.10' IDENTIFIED
BY 'password_access';
FLUSH PRIVILEGES;
CREATE DATABASE nova;
GRANT ALL ON nova.* TO 'nova'@'%' IDENTIFIED BY
'password_access';
GRANT ALL ON nova.* TO 'nova'@'localhost' IDENTIFIED BY
'password_access';
GRANT ALL ON nova.* TO 'nova'@'10.10.10.10' IDENTIFIED BY
'password_access';
GRANT ALL ON nova.* TO 'nova'@'192.168.1.10' IDENTIFIED BY
'password_access';
FLUSH PRIVILEGES;
CREATE DATABASE cinder;
GRANT ALL ON cinder.* TO 'cinder'@'%' IDENTIFIED BY
'password_access';
GRANT ALL ON cinder.* TO 'cinder'@'localhost' IDENTIFIED BY
'password_access';
GRANT ALL ON cinder.* TO 'cinder'@'10.10.10.10' IDENTIFIED BY
'password_access';
GRANT ALL ON cinder.* TO 'cinder'@'192.168.1.10' IDENTIFIED
BY 'password_access';
FLUSH PRIVILEGES;
EOF
```

A.1.3 Instalação configuração de componentes de rede para o OpenStack.

Instalação do serviço de vLAN e *bridge*, e configuração para habilitar o redirecionamento de pacotes.

```
apt-get install -y vlan bridge-utils
sed -i 's/#net.ipv4.ip_forward=1/net.ipv4.ip_forward=1/'/etc/sysctl.conf
sysctl net.ipv4.ip_forward=1
```

Configuração dos endereços IPs. Está configuração será temporária, posteriormente será novamente alterado os arquivos para configuração de outros serviços.

```
auto eth0
iface eth0 inet static
      address 10.10.10.10
      netmask 255.255.255.0
auto eth1
iface eth1 inet static
        address 192.168.1.10
        netmask 255.255.255.0
        gateway 192.168.1.1
        dns-nameservers 8.8.8.8
```

Nesta configuração, a interface eth0 representa a rede do *cluster* dos nodos. A interface eth1 representa a interface que tem acesso externo, conexão com a internet.

A.1.4 Instalação do componente Keystone

Instalação do componente

```
apt-get install –y keystone
```

Configuração do arquivo /etc/keystone/keystone.conf para adicionar os parâmetros corretos de acesso ao banco de dados.

```
connection = mysql://keystone:password access@10.10.10.10/keystone
```

No terminal, remover o banco de dados criado automaticamente ao instalar o serviço. Reiniciar o serviço. E após conectar ao bando de dados aonde o serviço insere as tabelas necessárias

```
rm /var/lib/keystone/keystone.db

service keystone restart

keystone-manage db_sync
```

Com o keystone executando, o próximo passo é criar os usuário e serviços que serão utilizados pelos outros componentes do OpenStack em conjunto com o keystone. Para isto, foi utilizado dois *scripts* para automatizar o processo e torna-lo mais rápido. Segue o script com a criação dos itens básico para o keystone.

```
#!/bin/sh
HOST_IP=10.10.10.10
ADMIN_PASSWORD=${ADMIN_PASSWORD:-password_access}
SERVICE_PASSWORD=${SERVICE_PASSWORD:-password_access}
export SERVICE_TOKEN="ADMIN"
export SERVICE_ENDPOINT="http://${HOST_IP}:35357/v2.0"
SERVICE_TENANT_NAME=${SERVICE_TENANT_NAME:-service}
get_id () {
    echo `$@ | awk '/ id / { print $4 }'`
}
# Tenants
ADMIN_TENANT=$(get_id keystone tenant-create --name=admin)
SERVICE_TENANT=$(get_id keystone tenant-create --name=$SERVICE_TENANT_NAME)
# Users
ADMIN_USER=$(get_id      keystone      user-create      --name=admin      --
pass="$ADMIN_PASSWORD" --email=admin@domain.com)
# Roles
ADMIN_ROLE=$(get_id keystone role-create --name=admin)
KEYSTONEADMIN_ROLE=$(get_id keystone role-create --name=KeystoneAdmin)
KEYSTONESERVICE_ROLE=$(get_id        keystone        role-create        --
name=KeystoneServiceAdmin)
# Add Roles to Users in Tenants
keystone user-role-add --user-id $ADMIN_USER --role-id $ADMIN_ROLE --tenant-
id $ADMIN_TENANT
```

Continuação do *script*

```
# The Member role is used by Horizon and Swift
MEMBER_ROLE=$(get_id keystone role-create --name=Member)

# Configure service users/roles
NOVA_USER=$(get_id      keystone      user-create      --name=nova      --
pass="$SERVICE_PASSWORD"        --tenant-id        $SERVICE_TENANT        --
email=nova@domain.com)
keystone user-role-add --tenant-id $SERVICE_TENANT --user-id $NOVA_USER --
role-id $ADMIN_ROLE

GLANCE_USER=$(get_id     keystone     user-create     --name=glance     --
pass="$SERVICE_PASSWORD"        --tenant-id        $SERVICE_TENANT        --
email=glance@domain.com)
keystone user-role-add --tenant-id $SERVICE_TENANT --user-id $GLANCE_USER -
-role-id $ADMIN_ROLE

NEUTRON_USER=$(get_id     keystone     user-create     --name=neutron     --
pass="$SERVICE_PASSWORD"        --tenant-id        $SERVICE_TENANT        --
email=neutron@domain.com)
keystone user-role-add --tenant-id $SERVICE_TENANT --user-id $NEUTRON_USER
--role-id $ADMIN_ROLE

CINDER_USER=$(get_id     keystone     user-create     --name=cinder     --
pass="$SERVICE_PASSWORD"        --tenant-id        $SERVICE_TENANT        --
email=cinder@domain.com)
keystone user-role-add --tenant-id $SERVICE_TENANT --user-id $CINDER_USER -
-role-id $ADMIN_ROLE

SWIFT_USER=$(get_id      keystone      user-create      --name=swift      --
pass="$SERVICE_PASSWORD"        --tenant-id        $SERVICE_TENANT        --
email=swift@domain.com)
keystone user-role-add --tenant-id $SERVICE_TENANT --user-id $SWIFT_USER -
-role-id $ADMIN ROLE
```

Após a criação dos usuários, inquilinos, e função de acordo como o keystone exige, executou-se um outro script para criação do restante dos componentes, onde cria-se os serviços e atribui estes aos usuários, inquilinos e funções criadas no *script* anterior.

```
#!/bin/sh
# Host address
HOST_IP=10.10.10.10
EXT_HOST_IP=192.168.1.10
# MySQL definitions
MYSQL_USER=keystone
MYSQL_DATABASE=keystone
MYSQL_HOST=$HOST_IP
MYSQL_PASSWORD=password_access
# Keystone definitions
KEYSTONE_REGION=RegionOne
export SERVICE_TOKEN=ADMIN
export SERVICE_ENDPOINT="http://${HOST_IP}:35357/v2.0"
while getopts "u:D:p:m:K:R:E:T:vh" opt; do
  case $opt in
    u)
      MYSQL_USER=$OPTARG
      ;;
    D)
      MYSQL_DATABASE=$OPTARG
      ;;
    p)
      MYSQL_PASSWORD=$OPTARG
      ;;
    m)
      MYSQL_HOST=$OPTARG
      ;;
    K)
      MASTER=$OPTARG
      ;;
    R)
      KEYSTONE_REGION=$OPTARG
      ;;
    E)
      export SERVICE_ENDPOINT=$OPTARG
```

Continuação do *script*.

```
    T)
      export SERVICE_TOKEN=$OPTARG
      ;;
    v)
      set -x
      ;;
    h)
      cat <<EOF
Usage: $0 [-m mysql_hostname] [-u mysql_username] [-D mysql_database] [-p
mysql_password]
       [-K keystone_master ] [ -R keystone_region ] [ -E
keystone_endpoint_url ]
       [ -T keystone_token ]

Add -v for verbose mode, -h to display this message.
EOF
      exit 0
      ;;
    \?)
      echo "Unknown option -$OPTARG" >&2
      exit 1
      ;;
    :)
      echo "Option -$OPTARG requires an argument" >&2
      exit 1
      ;;
  esac
done

if [ -z "$KEYSTONE_REGION" ]; then
  echo "Keystone region not set. Please set with -R option or set
KEYSTONE_REGION variable." >&2
  missing_args="true"
fi

if [ -z "$SERVICE_TOKEN" ]; then
  echo "Keystone service token not set. Please set with -T option or set
SERVICE_TOKEN variable." >&2
```

Continuação do *script.*

```
 missing_args="true"
fi
if [ -z "$SERVICE_ENDPOINT" ]; then
  echo "Keystone service endpoint not set. Please set with -E option or
set SERVICE_ENDPOINT variable." >&2
  missing_args="true"
fi
if [ -z "$MYSQL_PASSWORD" ]; then
  echo "MySQL password not set. Please set with -p option or set
MYSQL_PASSWORD variable." >&2
  missing_args="true"
fi
if [ -n "$missing_args" ]; then
  exit 1
fi
keystone service-create --name nova --type compute --description
'OpenStack Compute Service'
keystone service-create --name cinder --type volume --description
'OpenStack Volume Service'
keystone service-create --name glance --type image --description
'OpenStack Image Service'
keystone service-create --name keystone --type identity --description
'OpenStack Identity'
keystone service-create --name ec2 --type ec2 --description 'OpenStack EC2
service'
keystone service-create --name neutron --type network --description
'OpenStack Networking service'
keystone service-create --name swift --type object-store --description
'OpenStack Object Storage service'
create_endpoint () {
  case $1 in
    compute)
    keystone endpoint-create --region $KEYSTONE_REGION --service-id $2 --
publicurl 'http://'"$EXT_HOST_IP"':8774/v2/$(tenant_id)s' --adminurl
'http://'"$HOST_IP"':8774/v2/$(tenant_id)s' --internalurl
'http://'"$HOST_IP"':8774/v2/$(tenant_id)s'
    ;;
    volume)
```

Continuação do *script*

```
    keystone endpoint-create --region $KEYSTONE_REGION --service-id $2 --
publicurl 'http://'"$EXT_HOST_IP"':8776/v1/$(tenant_id)s' --adminurl
'http://'"$HOST_IP"':8776/v1/$(tenant_id)s' --internalurl
'http://'"$HOST_IP"':8776/v1/$(tenant_id)s'
    ;;
    image)
    keystone endpoint-create --region $KEYSTONE_REGION --service-id $2 --
publicurl 'http://'"$EXT_HOST_IP"':9292/v2' --adminurl
'http://'"$HOST_IP"':9292/v2' --internalurl 'http://'"$HOST_IP"':9292/v2'
    ;;
    identity)
    keystone endpoint-create --region $KEYSTONE_REGION --service-id $2 --
publicurl 'http://'"$EXT_HOST_IP"':5000/v2.0' --adminurl
'http://'"$HOST_IP"':35357/v2.0' --internalurl
'http://'"$HOST_IP"':5000/v2.0'
    ;;
    ec2)
    keystone endpoint-create --region $KEYSTONE_REGION --service-id $2 --
publicurl 'http://'"$EXT_HOST_IP"':8773/services/Cloud' --adminurl
'http://'"$HOST_IP"':8773/services/Admin' --internalurl
'http://'"$HOST_IP"':8773/services/Cloud'
    ;;
    network)
    keystone endpoint-create --region $KEYSTONE_REGION --service-id $2 --
publicurl 'http://'"$EXT_HOST_IP"':9696/' --adminurl
'http://'"$HOST_IP"':9696/' --internalurl 'http://'"$HOST_IP"':9696/'
    ;;
    object-store)
    keystone endpoint-create --region $KEYSTONE_REGION --service-id $2 --
publicurl 'http://'"$EXT_HOST_IP"':8080/v1/AUTH_%(tenant_id)s' --adminurl
'http://'"$HOST_IP"':8080/' --internalurl
'http://'"$HOST_IP"':8080/v1/AUTH_%(tenant_id)s'
    ;;
  esac
}
```

Continuação do *script.*

```
for i in compute volume image object-store identity ec2 network; do
  id=`mysql -h "$MYSQL_HOST" -u "$MYSQL_USER" -p"$MYSQL_PASSWORD"
"$MYSQL_DATABASE" -ss -e "SELECT id FROM service WHERE type='"$i"';"` ||
exit 1
  create_endpoint $i $id
done
```

Com o Keystone devidamente configurado, é importante adicionar algumas credenciais para uso do componente. Estas credenciais são em forma de variáveis de ambiente, sem isso pode ocorrer alguns erros de permissões e autenticações. Cria-se um arquivo com o seguinte conteúdo:

```
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=password_access
export OS_AUTH_URL="http://192.168.1.1:5000/v2.0/"
```

Com o arquivo criado, foi inserido no arquivo /root/.bashrc as linhas necessárias para que na inicialização do sistema operacional, este arquivo criado seja carreado para o ambiente do sistema. A configuração foi feita da seguinte forma no arquivo /root/.bashrc:

```
source /keystone_source
```

A.1.5 Instalação do componente Glance

Para instalar o componente Glance, utilizou-se o seguinte comando:

```
apt-get install -y glance
```

Com o componente instalado, os arquivos /etc/glance/glance-api-paste.ini e /etc/glance-registry-paste.ini devem ser editados com os seguintes parâmetros na seção “filter:authtoken” :

```
[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory
auth_host = 10.10.10.10
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = glance
admin_password = password_access
```

Nos arquivos /etc/glance/glance-api-paste.ini e /etc/glance/glance-registry-paste.ini, é necessário adicionar os seguintes parâmetros na seção “DEFAULT”:

```
[DEFAULT]
sql_connection = mysql://glance:openstacktest@10.10.10.51/glance

[keystone_authtoken]
auth_host = 10.10.10.10
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = glance
admin_password = openstacktest

[paste_deploy]
flavor = keystone
```

Remover o banco de dados criado no momento da instalação do serviço. Reiniciar os serviços glance-api e glance-registry. E sincronização do serviço com o banco de dados para criação das tabelas MySQL e reiniciar os serviços.

```
rm /var/lib/glance/glance.sqlite

service glance-api restart; service glance-registry restart

glance-manage db_sync

service glance-registry restart; service glance-api restart
```

Com o serviço instalado, adiciona-se as imagens que serão utilizadas nas máquinas virtuais instanciadas na infraestrutura OpenStack. Para isso, usou os seguintes comandos:

```
glance image-create --name myFirstImage --is-public true --container-
format bare --disk-format qcow2 --location
https://launchpad.net/cirros/trunk/0.3.0/+download/cirros-0.3.0-x86_64-
disk.img

wget http://cloud-images.ubuntu.com/precise/current/precise-server-
cloudimg-amd64-disk1.img

glance add name="Ubuntu 12.04 cloudimg amd64" is_public=true
container_format=ovf disk_format=qcow2 < ./precise-server-cloudimg-amd64-
disk1.img
```

A.1.6 Configuração do componente Neutron

Neutron é o componente do OpenStack que administra as redes. Mas com ele são necessários outros serviços para criação de interfaces virtuais, redes virtuais entre outros. Primeiramente, se instala o OpenVSwitch e se adiciona as interfaces virtuais.

```
apt-get install -y openvswitch-controller openvswitch-switch openvswitch-
datapath-dkms

service openvswitch-switch restart

#Adicionando a interface br-int
ovs-vsctl add-br br-int

#Adicionando a interface br-ex
ovs-vsctl add-br br-ex
```

A interface br-int é usada para integração das máquinas virtuais. Já a interface br-ex é usada para que as máquinas virtuais tenham acesso a internet.

Ainda é preciso alterar o arquivo /etc/network/interfaces, para configurar as interfaces de redes, e vincula-las as interfaces *bridges* virtuais criadas anteriormente. Para isso segue-se o exemplo:

```
auto eth0
iface eth0 inet static
        address 10.10.10.10
        netmask 255.255.255.0

auto eth1
iface eth1 inet manual
        up ifconfig $IFACE 0.0.0.0 up
        up ip link set $IFACE promisc on
        down ip link set $IFACE promisc off
        down ifconfig $IFACE down

auto br-ex
iface br-ex inet static
        address 192.168.1.10
        netmask 255.255.255.0
        gateway 192.168.1.1
        dns-nameservers 8.8.8.8
```

A interface física eth1 que obtem o acesso à internet, a partir desta configuração para o sistema OpenStack será identifica pela interface br-ex. Mas ainda é preciso configurar o sistema operacional e o serviço para efetivamente a interface

br-ex entrar em atividade. Executa-se o seguinte comando e reinicia o sistema operacional.

```
ovs-vsctl add-port br-ex eth1

#Reiniciar o Sistema Operacional
reboot
```

As etapas descritas até o momento são apenas para definição das interfaces virtuais e seus serviços. Na infraestrutura de componentes do OpenStack, Neutron define uma cadeia de componentes que estão ligadas à este componente e ao serviço de rede. Então é instala-se toda a cadeia de componentes do Neutron usando o seguinte comando:

```
apt-get install -y neutron-server neutron-plugin-openvswitch neutron-
plugin-openvswitch-agent dnsmasq neutron-dhcp-agent neutron-l3-agent
neutron-metadata-agent
```

Com os componentes instalados, reinicia-se todos os componentes e se edita o arquivo /etc/neutron/api-paste.ini na seção “filter:authtoken”.

```
[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory
auth_host = 10.10.10.10
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = neutron
admin_password = password_access
```

Seguindo, se edita o arquivo /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini nas seções “DATABASE”, “OVS” e “SECURITYGROUP”.

```
[DATABASE]
sql_connection=mysql://neutron:password_access@10.10.10.51/neutron
[OVS]
tenant_network_type = gre
enable_tunneling = True
tunnel_id_ranges = 1:1000
integration_bridge = br-int
tunnel_bridge = br-tun
local_ip = 10.10.10.10
[SECURITYGROUP]
firewall_driver =
neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
```

Ainda é preciso editar o arquivo /etc/neutron/metada_agent.ini, alterando os parâmetros que se encontram no arquivo e adequando para a infraestrutura local.

```
auth_url = http://10.10.10.10:35357/v2.0
auth_region = RegionOne
admin_tenant_name = service
admin_user = neutron
admin_password = password_access

nova_metadata_ip = 10.10.10.10

nova_metadata_port = 8775

metadata_proxy_shared_secret = helloOpenStack
```

O arquivo /etc/neutron/neutron.conf também deve ser editado, nas seções “keystone_authtoken” e “DATABASE” e adequar os parâmetros de acordo com a infraestrutura.

```
rabbit_host = 10.10.10.10
[keystone_authtoken]
auth_host = 10.10.10.10
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = neutron
admin_password = password_access
signing_dir = /var/lib/neutron/keystone-signing
[DATABASE]
connection = mysql://neutron:password_access@10.10.10.10/neutron
```

Edita-se ainda o arquivo /etc/neutron/l3_agent.ini, na seção “DEFAULT”.

```
[DEFAULT]
interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver
use_namespaces = True
external_network_bridge = br-ex
signing_dir = /var/cache/neutron
admin_tenant_name = service
admin_user = neutron
admin_password = password_access
auth_url = http://10.10.10.10:35357/v2.0
l3_agent_manager = neutron.agent.l3_agent.L3NATAgentWithStateReport
root_helper = sudo neutron-rootwrap /etc/neutron/rootwrap.conf
interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver
```

O arquivo /etc/neutron/dhcp_agent.ini deve ser alterado na seção “DEFAULT”.

```
[DEFAULT]
interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
use_namespaces = True
signing_dir = /var/cache/neutron
admin_tenant_name = service
admin_user = neutron
admin_password = password_access
auth_url = http://10.10.10.10:35357/v2.0
dhcp_agent_manager = neutron.agent.dhcp_agent.DhcpAgentWithStateReport
root_helper = sudo neutron-rootwrap /etc/neutron/rootwrap.conf
state_path = /var/lib/neutron
```

Após todas as edições completas, remove-se o banco de dados criado no momento da instalação do pacote, e reinicia-se todos os componentes do Neuton. Após é importante verificar se todos os serviços estão ativos.

```
rm /var/lib/neutron/neutron.sqlite

cd /etc/init.d/; for i in $( ls neutron-* ); do sudo service $i restart;
cd /root/; done
service dnsmasq restart
```

Após isto, é importante verificar se todos os componentes do Neutron estão executando normalmente. Para isto a forma simples é executando o comando para listar os serviços, e os mesmos devem ser seguido de um rosto sorridente. A Figura 84 demonstra o resultado do comando neutron agent-list executado no teminal do sistema operacional

```
+--------------------------------------+--------------------+----------------+-------+----------------+
| id                                   | agent_type         | host           | alive | admin_state_up |
+--------------------------------------+--------------------+----------------+-------+----------------+
| 26d7b754-b3b8-4ca4-822b-732d644a0761 | Open vSwitch agent | openstackfront | :-)   | True           |
| 64958dc4-79bf-4d0c-abd6-c58977d6e216 | L3 agent           | openstackfront | :-)   | True           |
| 72488675-4733-4600-9d16-ddf50242ccc8 | DHCP agent         | openstackfront | :-)   | True           |
| 875d2869-f4a9-43b6-b337-689da8551cd1 | Open vSwitch agent | openstack1     | :-)   | True           |
| b46211ee-9c6c-4e20-8b34-b397d2e64687 | Open vSwitch agent | openstack2     | :-)   | True           |
| e460e3ad-52d5-488c-b754-be99da20d50f | Open vSwitch agent | openstack3     | :-)   | True           |
+--------------------------------------+--------------------+----------------+-------+----------------+
```

Fonte: Maron, Griebler. 2014

Figura 84: Serviços ligados ao Neutron executando normalmente.

A.1.7 Instalação do *hypervisor* KVM e os componentes Nova

O *frontend* também pode fazer parte da infraestrutura e ceder recursos de processamento para as máquinas virtuais que serão alocadas. Para isto é necessário instalar o KVM e alguns componentes e serviços que se fazem necessários, usando os seguintes comandos:

```
apt-get install -y kvm libvirt-bin pm-utils
```

Com os components instalados, é necessário editar o arquivo /etc/libvirt/qemu.conf, na seção “cgroup_device_acl”, com os seguintes parâmetros:

```
cgroup_device_acl = [
"/dev/null", "/dev/full", "/dev/zero",
"/dev/random", "/dev/urandom",
"/dev/ptmx", "/dev/kvm", "/dev/kqemu",
"/dev/rtc", "/dev/hpet","/dev/net/tun"
]
```

Após a configuração deletar as redes virtuais criadas automaticamente durante a instalação do KVM.

```
virsh net-destroy default
virsh net-undefine default
```

Editar o arquivo /etc/libvirt/libvirt.conf os seguintes parâmetros

```
listen_tls = 0
listen_tcp = 1
auth_tcp = "none"
```

Editar o arquivo /etc/init/libvirt-bin.conf as seguintes variáveis:

```
libvirtd_opts="-d -l"
```

Após estas configurações e instalações, é necessário instalar os componentes ligados ao serviço Nova, que na infraestrutura dos componentes OpenStack representado vários outros componentes que estão diretamente ligados a administração das instâncias. Para instalar todos os componentes, é imprescindível executar os seguintes comandos:

```
apt-get install -y nova-api nova-cert novnc nova-consoleauth nova-
scheduler nova-novncproxy nova-doc nova-conductor nova-compute-kvm
```

Após a instalação, é importante editar os parâmetros na seção “filter:authtoken” no arquivo /etc/nova/api-paste.ini.

```
[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory
auth_host = 10.10.10.10
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = nova
admin_password = password_access
signing_dirname = /tmp/keystone-signing-nova
# Workaround for https://bugs.launchpad.net/nova/+bug/1154809
auth_version = v2.0
```

Editar o arquivo /etc/nova/nova.conf e adicionar os seguintes parâmetros:

```
[DEFAULT]
logdir=/var/log/nova
state_path=/var/lib/nova
lock_path=/run/lock/nova
verbose=True
api_paste_config=/etc/nova/api-paste.ini
compute_scheduler_driver=nova.scheduler.simple.SimpleScheduler
rabbit_host=10.10.10.10
nova_url=http://10.10.10.10:8774/v1.1/
sql_connection=mysql://nova:password_access@10.10.10.10/nova
root_helper=sudo nova-rootwrap /etc/nova/rootwrap.conf
# Auth
use_deprecated_auth=false
auth_strategy=keystone
# Imaging service
glance_api_servers=10.10.10.10:9292
image_service=nova.image.glance.GlanceImageService
# Vnc configuration
novnc_enabled=true
novncproxy_base_url=http://192.168.1.210:6080/vnc_auto.html
novncproxy_port=6080
vncserver_proxyclient_address=10.10.10.10
vncserver_listen=0.0.0.0
# Network settings
network_api_class=nova.network.neutronv2.api.API
neutron_url=http://10.10.10.10:9696
neutron_auth_strategy=keystone
neutron_admin_tenant_name=service
neutron_admin_username=neutron
neutron_admin_password=password_access
neutron_admin_auth_url=http://10.10.10.10:35357/v2.0
libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtHybridOVSBridgeDriver
linuxnet_interface_driver=nova.network.linux_net.LinuxOVSInterfaceDriver
```

Continuação do arquivo /etc/nova/nova.conf.

```
#Metadata
service_neutron_metadata_proxy = True
neutron_metadata_proxy_shared_secret = helloOpenStack
metadata_host = 10.10.10.10
metadata_listen = 0.0.0.0
metadata_listen_port = 8775
# Compute #
compute_driver=libvirt.LibvirtDriver
# Cinder #
volume_api_class=nova.volume.cinder.API
osapi_volume_listen_port=5900
cinder_catalog_info=volume:cinder:internalURL
```

Após, editar o arquivo /etc/nova/nova-compute.conf na seção “DEFAULT” e adicionar os seguintes parâmetros.

```
[DEFAULT]
libvirt_type=kvm
libvirt_ovs_bridge=br-int
libvirt_vif_type=ethernet
libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtHybridOVSBridgeDriver
libvirt_use_virtio_for_bridges=True
```

Após todas as edições, é preciso remover o banco de dados criado no momento da instalação. Reiniciar todos os serviços vinculado ao Nova. E finalmente sincronizar com o banco de dados.

```
#Remover o banco de dados padrão
rm /var/lib/nova/nova.sqlite

#Sincronizar com o banco
nova-manage db sync

#Reiniciar os serviços
cd /etc/init.d/; for i in $( ls nova-* ); do sudo service $i restart; cd
/root/;done
```

Após reiniciar os serviços, é importante verificar se os serviços estão rodando normalmente. Para isto, executa-se o comando nova-manage service list, e o resultado deverá conter a relação dos serviços com o desenho semelhante à rostos sorrindo, indicando que os serviços estão em execução. A Figura 85 demonstra o resultado do comendo executado no terminal do sistema operacional.

```
Binary           Host                                 Zone             Status     State Updated_At
nova-consoleauth openstackfront                       internal         enabled    :-)   2014-07-15 02:14:56
nova-cert        openstackfront                       internal         enabled    :-)   2014-07-15 02:14:52
nova-conductor   openstackfront                       internal         enabled    :-)   2014-07-15 02:14:53
nova-scheduler   openstackfront                       internal         enabled    :-)   2014-07-15 02:14:51
nova-compute     openstackfront                       nova             enabled    :-)   2014-07-15 02:15:00
nova-compute     openstack1                           nova             enabled    :-)   2014-07-15 02:14:53
nova-compute     openstack2                           nova             enabled    :-)   2014-07-15 02:14:54
nova-compute     openstack3                           nova             enabled    :-)   2014-07-15 02:14:57
```

Fonte: Maron, Griebler. 2014.

Figura 85: Serviços ligados ao Nova executando normalmente.

A.1.8 Configuração do componente Cinder

Na infraestrutura de componentes do OpenStack Cinder é representado de forma geral por vários outros componentes. Os comandos à seguir são necessários para instalar todos estes componentes.

```
apt-get install -y cinder-api cinder-scheduler cinder-volume iscsitarget
open-iscsi iscsitarget-dkms
```

Configurar o serviço de ISCSI, e após, iniciar os serviços.

```
#Configurar o serviço
sed -i 's/false/true/g' /etc/default/iscsitarget

#Iniciar os serviços
service iscsitarget start
service open-iscsi start
```

Configurar o arquivo /etc/cinder/api-paste.ini com os seguintes parâmetros na seção “filter:authtoken”.

```
[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory
service_protocol = http
service_host = 192.168.1.10
service_port = 5000
auth_host = 10.10.10.10
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = cinder
admin_password = password_access
```

Configura o arquivo /etc/cinder/cinder.conf na seção “DEFAULT” com os parâmetros seguintes.

```
[DEFAULT]
rootwrap_config=/etc/cinder/rootwrap.conf
sql_connection = mysql://cinder:password_access@10.10.10.10/cinder
api_paste_config = /etc/cinder/api-paste.ini
iscsi_helper=ietadm
volume_name_template = volume-%s
volume_group = cinder-volumes
verbose = True
auth_strategy = keystone
#osapi_volume_listen_port=5900
```

Remover o banco de dados criado automaticamente durante a instalação do componente. E após isso, sincronizar com o banco de dados.

```
#Excluir o banco de dados padrão
rm /var/lib/cinder/cinder.sqlite

#Sincronizar com o banco de dados MySQL
cinder-manage db sync
```

O Cinder necessita ser configurado para que crie as partições LVM dos discos criados na infraestrutura OpenStack. Então é importante que no *frontend* seja dedicado uma partição exclusiva para que isto seja feito. Para seguir com as configurações foi realizado o procedimento da seguinte forma:

```
pvcreate /dev/sda3
vgcreate cinder-volumes /dev/sda3
```

E após isto, reiniciado os serviços referente ao Cinder.

```
cd /etc/init.d/; for i in $( ls cinder-* ); do sudo service $i restart; cd
/root/; done
```

A.1.9 Instalação do componente Horizon (Interface WEB)

Horizon é a interface gráfica da ferramenta OpenStack que o torna acessível pelos navegadores. Para instalar este componente é necessário executar os seguintes comandos.

```
apt-get -y install openstack-dashboard memcached
```

De acordo com a documentação, foi recomendado remover um tema que é instalado juntamente com o Horizon, para isto executou-se o seguinte comando, que posteriormente sera necessário reiniciar o serviço Apache:

```
#Remover pacote
dpkg --purge openstack-dashboard-ubuntu-theme

#Reiniciar Apache
service apache2 restart; service memcached restart
```

A.2. Preparação e configuração do node

Com o sistema operacional Ubuntu 12.04 instalado, primeiramente adiciona-se os repositórios de atualizações do OpenStack Havana.

```
apt-get install ubuntu-cloud-keyring python-software-properties software-
properties-common python-keyring
echo deb http://ubuntu-cloud.archive.canonical.com/ubuntu precise-
proposed/havana main >> /etc/apt/sources.list.d/havana.list
```

Com os repositórios instalados, executa-se os commandos para instalar as atualizações. Após terminado o processo, é preciso reiniciar o sistema operacional.

```
apt-get update
apt-get upgrade
```

Seguindo com a instalação, instala-se o NTP, para sincronização de horário entre nodos e *frontend*. As a seguir as linhas devem ser comentadas no arquivo /etc/ntp.conf, e adicionar o *frontend* como servidor de sincronização.

```
#server 0.ubuntu.pool.ntp.org
#server 1.ubuntu.pool.ntp.org
#server 2.ubuntu.pool.ntp.org
#server 3.ubuntu.pool.ntp.org
#ntp.ubuntu.com
server 10.10.10.10
```

Após instala-se os serviços para rede:

```
apt-get install -y vlan bridge-utils
```

Após instalado, se habilita o redirecionamento de pacotes no sistema operacional, habilitando as seguintes linhas no arquivo /etc/sysctl.conf. Após alteração é necessário digira a mesma linha inserida no arquivo no terminal do sistema operacional, para que a alteração seja gravada e não se perca no momento da inicialização do sistema operacional

```
#Linha inserida no arquivo /etc/sysctl.conf
net.ipv4.ip_forward=1

#Comando que dever ser inserido no terminal do Sistema
net.ipv4.ip_forward=1
```

A.2.1 Configuração da rede

A interface de configuração deve ser configurada com a mesma rede que será usada nas configurações do OpenStack. Portanto, o endereçamento usado foi o seguinte:

```
auto eth0
 iface eth0 inet static
        address 10.10.10.11
        netmask 255.255.255.0
        gateway 10.10.10.10
```

A.2.2 Instalação e configuração do KVM

A seguir serão detalhadas as configurações para instalação do *hypervisor*. Para instalação utilizou-se os comandos a seguir:

```
apt-get install -y kvm libvirt-bin pm-utils
```

No momento da instalação o KVM cria automaticamente interfaces de redes virtuais, então é importante e remoção desta interface com os seguintes comandos. E após isso, reiniciar os serviços:

```
virsh net-destroy default
virsh net-undefine default
service dbus restart && service libvirt-bin restart
```

A.2.3 Instalação e configuração do OpenVSwitch

Para criação das redes e interfaces virtuais que serão utilizadas nas máquinas virtuais é necessário instalar o OpenVSwitch, para isto, foram usados os seguintes comandos. Após instalação reinicia-se o serviço:

```
#Instalar os serviços
apt-get install -y openvswitch-controller openvswitch-switch openvswitch-
datapath-dkms

#Reiniciar o serviço
/etc/init.d/openvswitch-switch restart
```

Com os serviços rodando, é necessário adicionar as interfaces virtuais que são integradas as máquinas virtuais. Usou-se o seguinte comando:

```
ovs-vsctl add-br br-int
```

A.2.3.1 Instalação do Neutron

Nos nodos somente é instalado o agente *OpenVSwitch* do Neutron, portanto para configuração e instalação foram usados os seguintes comandos

```
apt-get -y install neutron-plugin-openvswitch-agent
```

Após, configurar o arquivo /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini, nas seções “DATABASE”, “OVS”, “SECURITYGROUP”.

```
[DATABASE]
connection = mysql://neutron:openstacktest@10.10.10.10/neutron

[OVS]
tenant_network_type = gre
tunnel_id_ranges = 1:1000
integration_bridge = br-int
tunnel_bridge = br-tun
local_ip = 10.10.10.12
enable_tunneling = True

 [SECURITYGROUP]
firewall_driver =
neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
```

Configurar também o arquivo /etc/neutron/neutron.conf, nas seções “keystone_authtoken”, “DATABASE”.

```
rabbit_host = 10.10.10.10

[keystone_authtoken]
auth_host = 10.10.10.10
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = neutron
admin_password = password_access
signing_dir = /var/lib/neutron/keystone-signing

[DATABASE]
connection = mysql://neutron:password_access@10.10.10.10/neutron
```

Após alteração reiniciar os serviços.

```
service neutron-plugin-openvswitch-agent restart
```

A.2.4 Instalação e configuração do Nova

Nova é o conjunto de componentes da infraestrutura OpenStack que gerencia recursos ligados às instâncias da nuvem, gerenciando também as próprias instâncias. Para instalar no node, foi executado os seguintes comandos.

```
apt-get install -y nova-compute-kvm
```

Após instalação modificar as seções “filter:authtoken” no arquivo /etc/nova/api-paste.ini com os seguintes parâmetros:

```
[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory
auth_host = 10.10.10.10
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = nova
admin_password = openstack
signing_dirname = /tmp/keystone-signing-nova
# Workaround for https://bugs.launchpad.net/nova/+bug/1154809
auth_version = v2.0
```

Após, editar o arquivo /etc/nova/nova-compute na seção “DEFAULT” com os seguintes parâmetros.

```
[DEFAULT]
libvirt_type=kvm
compute_driver=libvirt.LibvirtDriver
libvirt_ovs_bridge=br-int
libvirt_vif_type=ethernet
libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtHybridOVSBridgeDriver
libvirt_use_virtio_for_bridges=True
```

Seguindo com as configurações, é importante editar o arquivo /etc/nova/nova.conf, adicionando e alterando os parâmetros de configuração citados a seguir.

```
[DEFAULT]
logdir=/var/log/nova
state_path=/var/lib/nova
lock_path=/run/lock/nova
verbose=True
api_paste_config=/etc/nova/api-paste.ini
compute_scheduler_driver=nova.scheduler.simple.SimpleScheduler
rabbit_host=10.10.10.51
nova_url=http://10.10.10.51:8774/v1.1/
sql_connection=mysql://nova:openstacktest@10.10.10.51/nova
root_helper=sudo nova-rootwrap /etc/nova/rootwrap.conf

# Auth
use_deprecated_auth=false
auth_strategy=keystone

# Imaging service
glance_api_servers=10.10.10.51:9292
image_service=nova.image.glance.GlanceImageService

# Vnc configuration
novnc_enabled=true
novncproxy_base_url=http://192.168.1.251:6080/vnc_auto.html
novncproxy_port=6080
vncserver_proxyclient_address=10.10.10.52
vncserver_listen=0.0.0.0

# Network settings
network_api_class=nova.network.neutronv2.api.API
neutron_url=http://10.10.10.51:9696
neutron_auth_strategy=keystone
neutron_admin_tenant_name=service
neutron_admin_username=neutron
neutron_admin_password=openstacktest
neutron_admin_auth_url=http://10.10.10.51:35357/v2.0
libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtHybridOVSBridgeDriver
linuxnet_interface_driver=nova.network.linux_net.LinuxOVSInterfaceDriver
#If you want Neutron + Nova Security groups
firewall_driver=nova.virt.firewall.NoopFirewallDriver
security_group_api=neutron
```

Continuação do arquivo

```
#Metadata
service_neutron_metadata_proxy = True
neutron_metadata_proxy_shared_secret = helloOpenStack

# Compute #
compute_driver=libvirt.LibvirtDriver

# Cinder #
volume_api_class=nova.volume.cinder.API
osapi_volume_listen_port=5900
cinder_catalog_info=volume:cinder:internalURL
```

A.3 Criando uma instância no OpenStack

Com o sistema configurado e em funcionamento, existem dois meios de criar a administrar os recursos da infraestrutura OpenStack. Através da linha de comando é uma das alternativas, porém exige do administrador um conhecimento mais técnico e detalhado de todos os componentes que constituem o OpenStack. E outra alternativa é por meio da interface gráfica (*Dasboard* Horizon*)* da ferramenta, que possui uma imagem mais atraente e funcional.

Para a criação das instâncias é preciso tanto pelo terminal como pela interface gráfica, criar as redes, sub-redes, roteadores e links virtuais para as instâncias.

A.3.1 Criando redes virtuais e instancias pelo terminal do sistema operacional

Iniciando o processo de criação das redes, primeiramente se cria as novas redes:

```
neutron net-create --tenant-id $coloque_a_id_do_projeto_criado
nome_da_rede
```

Criar as sub-nets com endereços de rede e gateways, e definir para qual usuário será disponível a rede

```
neutron subnet-create --tenant-id $coloque_id_do_projeto_criado
nome_do_projeto 50.50.1.0/24 --dns_nameservers list=true 192.168.1.1
```

Criar o roteador da rede virtual.

```
neutron router-create --tenant-id $coloque_id_do_projeto_criado
nome_do_roteador_do_projeto
```

Os comandos a seguir, serviram para definir ao roteador qual serviço de *layer* 3 ele deverá usar no como principal no serviço da rede.

```
#Digite no terminal, e copie a ID do serviço l3-agent
neutron agent-list

#Defina ao roteador qual é o serviço para conectar
neutron l3-agent-router-add $ID_l3_agente nome_do_roteador_do_projeto
```

Após, adicionar as subnets ao roteador criado para o projeto.

```
neutron router-interface-add $coloque_ID_do_projeto_criado
$coloque_id_da_subnet_criada
```

Os passos anteriores serviram para criar as redes internas, os roteadores principais e definir algumas configurações. Agora é preciso criar as redes externas para a infraestrutura. Os comandos usados foram os seguintes.

Criar a rede externa

```
neutron net-create --tenant-id $ coloque_ID_do_projeto_criado
nome_da_rede_externa --router:external=True
```

Criar as subnets, gateway e a faixa de IPs para disponibilizar IPs flutuantes às máquinas virtuais terem acesso à internet.

```
neutron subnet-create --tenant-id $coloque_id_do_usuário_criado --
allocation-pool start=192.168.1.52,end=192.168.1.76 --gateway
192.168.1.251 nome_rede_externa 192.168.1.0/24 --enable_dhcp=False
```

Após criar a rede, conecta-la ao roteador criado anteriormente.

```
neutron router-gateway-set $coloque_a_ID_do_roteador_
$coloque_ID_da_rede_externa
```

Após o processo de criação das redes e roteadores, é necessário criar os certificados de acessos que serão inseridos polos serviços do OpenStack nas instâncias criadas. Estes certificados permitem o acesso via SSH às máquinas. Os comandos usados foram os seguintes:

```
#Criar a chave do usuário local no sistema operacional
ssh-keygen -t rsa

#Adicionar a chave criada ao serviço Nova.
nova keypair-add --pub_key ~/.ssh/id_rsa.pub mykey

#Para listar as chaves existentes no serviço Nova
nova keypair-list
```

Com as chaves criadas, é importante criar algumas regras de segurança que também serão inseridas no momento da criação de uma instância. Neste caso, foram criadas as regras para permitir que as instâncias respondam ao comando *ping* e permitir acesso via SSH. Os comandos usados foram os seguintes:

```
nova --no-cache secgroup-add-rule default icmp -1 -1 0.0.0.0/0

nova --no-cache secgroup-add-rule default tcp 22 22 0.0.0.0/0
```

Com os processos realizados até esta etapa, agora cria-se a instância na infraestrutura OpenStack.

```
nova --no-cache boot --image $ID_da_imagem_do_SO --flavor 1
$nome_maquina_virtual
```

Para acessar a instância criada, pode se utilizar o modo clássico de acesso com o commando ssh+ip_da_máquina. Porém, em alguns casos deve-se usar o recurso netns do sistema operacional Linux. Portanto, o acesso pode se tornar um pouco mais complexo na utilização de comandos. O procedimento de acesso com o netns foi o seguinte:

```
#Listar as redes e roteadores internos
Ip netns
```

O resultado será as redes com suas respectivas identificações, que em cada implantação será diferente. Portanto para fins de exemplo, será usado a identifica da rede da infraestrutura criada.

```
#Resultado
qdhcp-6d9da13e-263b-4162-8d10-beed97497723

#Comando para acesso via ssh

ip netns exec qdhcp-6d9da13e-263b-4162-8d10-beed97497723 ssh
usuario@IP_da_maquina
```

A.3.2 Criando instancias e redes virtuais pela Dashboard (Horizon) do OpenStack

A interface gráfica proporciona facilitar visualmente a interação entre a ferramenta e o usuário. Portanto para criar as redes e instâncias são seguidos os seguintes passos:

Acessar o endereço e autenticar com usuário na *Dashboard.*

Fonte: Maron, Griebler. 2014

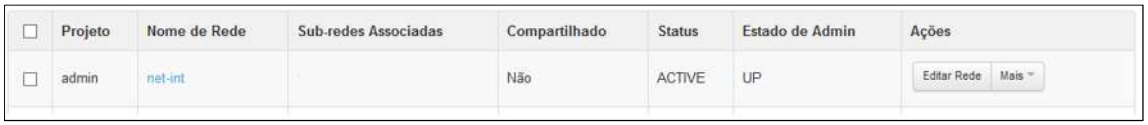
Figura 86: Autenticação Dashboard

No painel à esquerda, ir até Redes e clicar no ícone Criar Rede.

Fonte: Maron, Griebler. 2014

Figura 87: Painel Dashboard

Com a rede criada, deve-se acessar ela clicando sobre o nome da rede, e criar as subredes.



Fonte: Maron, Griebler. 2014

Figura 88: Criando subrede na Dashboard.

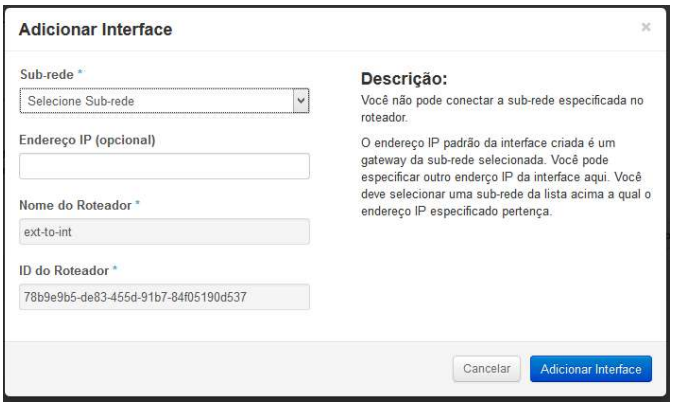
Fonte: Maron, Griebler. 2014.

Figura 89: Editando a sub-rede

Na Figura 89, mostra o painel para editar as subredes criadas, nele adiciona-se a faixa de endereços de rede, o *gateway* da rede. Na aba detalhes da

sub-rede configura-se os endereços de DNS, habilitar o DHCP para as instâncias e rotas de *host.*

Para criar os roteadores, no painel da Figura 87, clica-se no menu Roteadores e após isso clicar na opção Criar Roteador, aonde na primeira etapa será pedido apenas o nome do roteador. Com o roteador criado, é necessário adicionar as interfaces para que ele se conecte nas redes e subredes criadas anteriormente. Para isto, é necessário acessar o roteador criado e selecionar a opção Adicionar Interface.

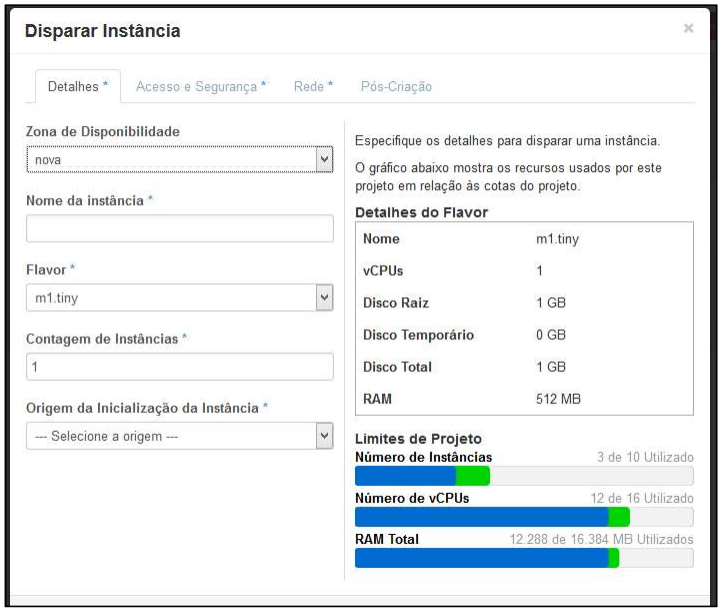


Fonte: Maron, Griebler. 2014.

Figura 90: Adicionando interface ao Roteador

A Figura 90 mostra os detalhes para a criação da porta das redes e subredes ao roteador. Na imagem tem a opção de escolha das sub-redes, e opcionalmente definir um endereço IP à ela, porém se o recurso DHCP estiver habilitar não há necessidade desta configuração.

Com as redes criadas, pode-se partir para a criação das instâncias. Para isto, clica-se na aba Projeto, como pode ser visto na Figura 32, e após isso no menu Instâncias, onde deverá ser selecionado o menu Disparar Instância. Ao clicar neste menu será aberta uma nova janela com as opções de Detalhes técnicos para criação da instância, origem da criação (Imagem ISO, Volume, Snapshot). Acesso e Segurança aonde se defini as chaves de públicas para acesso via SSH pelo terminal do sistema operacional. Define-se a Rede que será usada na Instância. E na aba Pós-Criação é possível definir alguns *scripts* que serão executados após a criação da máquina. A Figura 91 mostra alguns detalhes.



Fonte: Maron, Griebler. 2014.

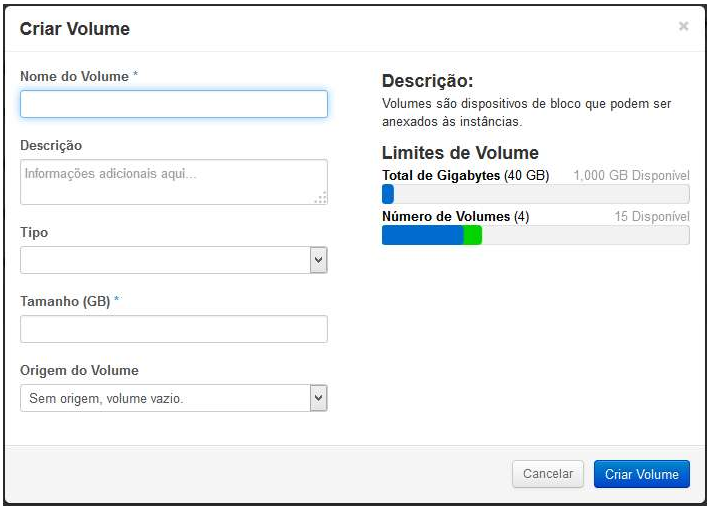
Figura 91: Disparar uma Instância.

É possível perceber também nesta etapa, os limites de recursos do projeto criado. Em uma forma bem intuitiva, ele mostra os números de instâncias, vCPUs e memória RAM em uso e ainda disponíveis.

A.4 Criação de volumes pela *Dashboard*

A possibilidade de criação de blocos de armazenamento permite que os volumes criados sejam anexados às instâncias criadas. Estes blocos de armazenamento podem ser unidades vazias para a instância, ou até mesmo blocos com o sistema operacional já incluso.

Para criação dos volumes é necessário clicar na aba projetos e depois em volumes, como mostra a Figura 92, e nesta página clicar em Cria Volume. Nesta opção ser aberto uma nova janela aonde terá as definições do volume a ser criado.



Fonte: Maron, Griebler. 2014.

Figura 92: Criando Volume.

A Figura 92, mostra as opções disponíveis no momento da criação. Aonde existe o campo dedicado ao nome do volume, descrição e capacidade em *GigaBytes*. O campo “Tipo” refere-se à uma organização dos volumes criados, não exerce nennuma influência em seu modo de operação. Por exemplo, levando em conta a capacidade, pode-se criar vários volumes, então define-se grupos como Backup, Sistema Operacional, para facilitar na organização e administração.

Já a “Origem do Volume” é que define o seu modo de operação em uma Instância. Nele tem a opção de “Sem origem, volume vazio”, que será criado um volume sem nenhum tipo de dados, sendo um armazenamento livre para a instância criada. E ainda tem a opção Imagem aonde será escolhida uma imagem de algum sistema operacional para popular o volume, simulando um caso real de um computador clássico com o sistema operacional já incluso.

APÊNDICE B. INSTALAÇÃO E CONFIGURAÇÃO DO OPENNEBULA

Nas seções seguintes, serão detalhados os procedimentos usados para a instalação e configuração da ferramenta OpenNebula. Para realizar estes procedimentos, foram usados documentos oficiais da distribuição OpenNebula adequando as configurações para a infraestrutura desejada para a realização dos testes.

B.1 Configuração do *frontend*

Para iniciar a configuração do *frontend* foi instalado o serviço para configuração das interfaces de rede *bridges.* Para isto foi usado o comando:

```
apt-get install bridge-utils
```

Após a instalação foi configurado o arquivo /etc/networking/interfaces para definição dos endereços de redes e interfaces *bridges.* Após a alteração do arquivo, foi preciso reinicializar o sistema operacional

```
auto eth0
iface eth0 inet static
        address 192.168.1.11
        netmask 255.255.255.0
        gateway 192.168.1.1
        dns-nameservers 8.8.8.8
auto br0
iface br0 inet static
      address 11.11.11.10
      netmask 255.255.255.0
      network 11.11.11.0
      broadcast 11.11.11.255
            bridge_ports eth1
            bridge_fd 9
            bridge_hello 2
            bridge_maxage 12
            bridge stp off
```

Na configuração usada, a interface eth0 é destinada ao acesso à internet. A interface br0, é a configuração para rede do restante dos nodos.

Após as definições de rede e a reinicialização foi criado o grupo usado pelo OpenNebula. Também foi criado o usuário que será usado pela ferramenta, e após criado uma senha de autenticação. Os comandos usados foram os seguintes:

```
#Criação do grupo
groupadd –g 10000 oneadmin

#Criação do usuário
useradd –u 10000 –m oneadmin –d /var/lib/one –s /bin/bash -g 10000 oneadmin

#Alteração de senha
passwd oneadmin
```

Após a criação dos grupos e usuários, foi instalado o NFS, o sistema de arquivos para a rede, no qual compartilha os diretórios usados pelos *nodes* e o *frontend*. É necessário também a criação de chaves SSH para o acesso sem senha aos outros nodos.

```
#Instalação do pacote NFS
apt-get install nfs-kernel-server

#Configuração do arquivo /etc/exports para compartilhar a unidade do
#frontend
/var/lib/one
192.168.1.0/24(rw,sync,no_subtree_check,no_root_squash,anonuid=10000,anong
id=10000)
```

Para criar as chaves SSH e habilitar as configurações para acesso sem senha.

```
#Logar com o usuário oneadmin
su -l oneadmin
#Executar o commando para gerar a chave pública
ssh-keygen

#Criar o arquivo que autoriza o acesso
cat ~/.ssh/id_rsa.pub > ~/.ssh/authorized_keys
```

Prosseguindo, a instalação das dependências do OpenNebula necessárias no sistema operacional para prosseguir com a instalação dos componentes do OpenNebula.

```
apt-get install libsqlite3-dev libxmlrpc-c3-dev g++ ruby libopenssl-ruby
libssl-dev ruby-dev libxml2-dev libmysqlclient-dev libmysql++-dev
libsqlite3-ruby libexpat1-dev rake rubygems libxml-parser-ruby1.8 libxslt1-
dev genisoimage scons
```

Para seguir a instalação das dependências, é preciso instalar a versão mais recente do Ruby.

```
apt-get install python-software-properties
apt-add-repository ppa:brightbox/ruby-ng
apt-get update
apt-get install build-essential ruby rubygems ruby-switch
apt-get install ruby1.9.1-full
ruby-switch --set ruby1.9.1
```

Continuar a instalação do restante das dependências e o gerenciador do banco de dados MySQL.

```
gem install nokogiri rake xmlparser
apt-get install mysql-server
```

Após instalar o gerenciador do banco de dados, é necessário criar o banco, usuários e definir as permissões de acesso. Para isso, foi usado os seguintes comandos

```
#Logar no Gerenciandor do banco de dados
mysql –uroot –psenha

#Criar os usuário oneadmin
CREATE USER ‘oneadmin’@’localhost’ IDENTIFIED BY ‘oneadmin’;

#Criar o banco de dados
CREATE DATABASE OpenNebula;

#Definindo as permissões para acesso completo ao banco criado
GRANT ALL PRIVILEGES ON OpenNebula. * TO ‘oneadmin’ IDENTIFIED BY ‘oneadmin’
quit;
```

Com o ambiente preparado, foi realizado o download da última versão estável disponível para instalação. O arquivo foi descompactado e alterado o suporte ao MySQL invés de sqlite. Dentro do diretório dos arquivos baixados da versão 4.6.2-1 do OpenNebula, executou-se o comando.

```
scons sqlite=no mysql=yes
```

Após o procedimento realizado definitivamente o sistema foi instalado. Para isto, dentro do diretório de instalação, executou-se o seguinte comando:

```
./install.sh -u oneadmin -g oneadmin -d /var/lib/one
```

Após a instalação, é importante definir as variáveis de ambiente para que seja utilizada no OpenNebula. O arquivo criado consta com o local de instalação do OpenNebula, arquivo de autenticação e o caminho para alguns recursos necessário para o funcionamento da ferramenta.

```
export ONE_LOCATION=/var/lib/one
export ONE_AUTH=$ONE_LOCATION/.one/one_auth
export ONE_XMLRPC=http://localhost:2633/RPC2
export
PATH=$ONE_LOCATION/bin:/usr/local/bin:/var/lib/gems/1.8/bin/:/var/lib/gems
/1.8/:$PATH
```

O arquivo criado para as variáveis de ambiente é importante que seja adicionado ao arquivo bash_profile do usuário oneadmin. Fazendo isso, toda vez que o sistema é inicializado e o usuário é logado, esse arquivo é carregado automaticamente.

```
source ~/.bash profile
```

Foi preciso ainda criar o diretório para armazenar o arquivo contendo as credenciais de *login* do usuário oneadmin.

```
su - oneadmin
mkdir ~/.one
echo “oneadmin:senha” ~/.one/one_auth
```

Com o sistema instalado no ambiente, é necessário a alteração de alguns arquivos para que seu funcionamento esteja completo. O arquivo alterado foi

/var/lib/one/etc/onde.conf, comentado a linha que deve omitir a utilização do sqlite pelo OpenNebula. E descomentadas as linhas que indique a utilização do MySQL.

```
#Linha a ser comentada
DB = [ backend = "sqlite" ]

#Linhas que devem ser descomentadas
DB = [ backend = "mysql",
server = "localhost",
port = 0,
user = "oneadmin",
passwd = "oneadmin",
db_name = "OpenNebula" ]
```

Uma das configurações recomendadas é a ativação da tolerância a falhas. Para ativar esta função, foi necessário alterar o arquivo /var/lib/one/etc/onde.conf na seção “Fault Tolerance Hooks” e descomentar as linhas apartir dele.

```
HOST_HOOK = [
name = "error",
on = "ERROR",
command = "ft/host_error.rb",
arguments = "$ID -r",
remote = "no" ]

VM_HOOK = [
name = "on_failure_recreate",
on = "FAILED",
command = "/usr/bin/env onevm delete --recreate",
arguments = "$ID" ]
```

B.1.1 Instalação e configuração do Sunstone

O pacote Sunstone, já é instalado no momento da primeira instalação do arquivo OpenNebula. Porém ainda é preciso instalar alguns pacotes para seu correto funcionamento.

```
apt-get install rails thin
gem install json sinatra thin
```

Foi executado este comando para que se evite o erro no formato da data no Ruby.

```
sudo sed -i 's/ 00:00:00.000000000Z//' /var/lib/gems/1.8/specifications/*
```

Para que o Sunstone seja acessível foi editado o arquivo /etc/var/lib/one/etc/sunstone-server.conf para definir em qual endereço IP o Susnstone será possível acessar sua interface gráfica. Para isto foi adicionado o seguinte IP

```
:host: 192.168.x.x
```

B.2 Instalação no node

Para iniciar a instalação do node, foi necessário configurar os pacotes que permite a criação de redes virtuais e se conceda a comunicação entre o restante dos nodos. Ao instalar o pacote, o arquivo /etc/network/interfaces foi editado, e adequado o endereçamento de rede de acordo com o ambiente.

```
#Instalação do pacote
apt-get install bridge-utils

#Configuração da interface de rede
auto br0
iface br0 inet static
        address 11.11.11.11
        netmask 255.255.255.0
        gateway 11.11.11.10
        network 11.11.11.0
        bridge_ports eth2
        bridge_fd 9
        bridge_hello 2
        bridge_maxage 12
        bridge_stp off
```

Após isto, foram criados o usuário e o grupo oneadmin para uso no OpenNebula.

```
#Criação do grupo ondeadmin
groupadd –g 10000 oneadmin

#Adicionando o usuário
useradd –u 10000 –m oneadmin –d /var/lib/one –s /bin/bash -g 10000 oneadmin
```

Ao prosseguir com a instalação, é necessário instalar o serviço de NFS para montagem do diretório compartilhado. Aonde após a instalação do serviço, foi configurado o diretório compartilhado para no /etc/rc.local para que quando no momento em que o sistema fosse iniciado, a unidade seja montada automaticamente.

```
#Instalação do serviço NFS
apt-get install nfs-common

Editando o arquivo /etc/rc.local
mount -t nfs 11.11.11.10:/var/lib/one /var/lib/one
```

B.1.2 Instalação e configuração do *hypervisor*

Após as configurações demonstradas até aqui, foi realizada a instalação dos pacotes das ferramentas de virtualização.

```
apt-get install qemu-kvm libvirt-bin ubuntu-vm-builder
```

Com a instalação terminada, são adicionados ao grupo libvirt e kvm o usuário root e o usuário oneadmin.

```
adduser `id -un` libvirtd
adduser oneadmin libvirtd
adduser `id -un` kvm
adduser oneadmin kvm
```

Ao concluir esta estapa, os arquivos /etc/libvirt/libvirtd.conf e /etc/libvirt/qemu.conf deve ter algumas linhas de configurações descomentada. As linhas desabilitam e habilitam funções ligadas ao KVM, e aplique algumas permissões aos usuários do sistema.

```
listen_tls = 0
listen_tcp = 1
mdns_adv = 0
unix_sock_group = “oneadmin”
unix_sock_rw_perms = “0777”
vnc_listen = “0.0.0.0”
user = “oneadmin”
group = “oneadmin”
dynamic_ownership = 0
```

Reinicialize o sistema do libvirt após realizar as configurações.

```
sudo service libvirt-bin restart
```

Após é importante configurar o acesso aos usuários do grupo oneadmin.

```
chown :oneadmin /var/run/libvirt/libvirt-sock
```

Com todas as configurações realizadas, o Opennebula pode ser executado, iniciando a todos os componentes instalados.

```
one start
sunstone-server start
```

O node configurado ainda não faz parte completamente do *cluster* OpenNebula. Para adicionar os *hosts* é necessário executar o comando onehost com a identificação do *hostnamen* do ambiente do sistema operacional.

```
onehost create nomedohost --im kvm --vm kvm --net dummy
```

APÊNDICE C. SCRIPT PARA EXECUÇÃO DO BENCHMARK NPB-MPI

```
#!/bin/bash

BTB_MPI="/home/wcp/NPB3.3/NPB3.3-MPI/bin/bt.B"
CGB_MPI="/home/wcp/NPB3.3/NPB3.3-MPI/bin/cg.B"
DTB_MPI="/home/wcp/NPB3.3/NPB3.3-MPI/bin/dt.B"
EPB_MPI="/home/wcp/NPB3.3/NPB3.3-MPI/bin/ep.B"
FTB_MPI="/home/wcp/NPB3.3/NPB3.3-MPI/bin/ft.B"
ISB_MPI="/home/wcp/NPB3.3/NPB3.3-MPI/bin/is.B"
LUB_MPI="/home/wcp/NPB3.3/NPB3.3-MPI/bin/lu.B"
MGB_MPI="/home/wcp/NPB3.3/NPB3.3-MPI/bin/mg.B"
SPB_MPI="/home/wcp/NPB3.3/NPB3.3-MPI/bin/sp.B"

array=( "exec1" "exec2" "exec4" "exec8" "exec9" "exec16" )
exec1()
{
PROCESS="1"
arrayexec1=("$SPB_MPI.$PROCESS"   "$BTB_MPI.$PROCESS"   "$CGB_MPI.$PROCESS"
"$EPB_MPI.$PROCESS"         "$FTB_MPI.$PROCESS"         "$ISB_MPI.$PROCESS"
"$LUB_MPI.$PROCESS"
"$MGB_MPI.$PROCESS" )
for((CONT1=1;CONT1<41;CONT1++))
        do
                for((AUX1=0;AUX1<${#arrayexec1[@]};AUX1++))
                do
                  LOG1=(`echo ${arrayexec1[$AUX1]}| awk -F/ '{print $7}'`)
                        echo Bencmark: $LOG1
                        echo Contador: $CONT1
                  echo "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
" >> /home/wcp/logs/$LOG1.log
                  echo   Start   in   `date   +%H:%M:%S--%d/%m/%Y`   2>&1>>
/home/wcp/logs/$LOG1.log
                  mpirun    -np    $PROCESS    -machinefile    /etc/nodefile
${arrayexec1[$AUX1]} >> /home/wcp/logs/$LOG1.log
                        echo  Finish  in  `date  +%H:%M:%S--%d/%m/%Y`  2>&1>>
/home/wcp/logs/$LOG1.log
                  echo "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
" >> /home/wcp/logs/$LOG1.log
                  sleep 1;
```

Continuação do *script.*

```
                done
        done
}
exec2()
{
PROCESS="2"
arrayexec2=( "$CGB_MPI.$PROCESS" "$EPB_MPI.$PROCESS" "$FTB_MPI.$PROCESS"
"$ISB_MPI.$PROCESS" "$LUB_MPI.$PROCESS" "$MGB_MPI.$PROCESS" )
for((CONT2=1;CONT2<41;CONT2++))
        do
                for((AUX2=0;AUX2<${#arrayexec2[@]};AUX2++))
                do
                 LOG2=(`echo ${arrayexec2[$AUX2]}| awk -F/ '{print $7}'`)
                        echo Bencmark: $LOG2
                        echo Contador: $CONT2
                        echo "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
/-/-/-/-" >> /home/wcp/logs/$LOG2.log
                 echo Start in `date +%H:%M:%S--%d/%m/%Y` 2>&1>>
/home/wcp/logs/$LOG2.log
                 mpirun -np $PROCESS -machinefile /etc/nodefile
${arrayexec2[$AUX2]} >> /home/wcp/logs/$LOG2.log
                        echo Finish in `date +%H:%M:%S--%d/%m/%Y` 2>&1>>
/home/wcp/logs/$LOG2.log
                        echo "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
/-/-/-/-" >> /home/wcp/logs/$LOG2.log
                 sleep 1;
                done
        done
}

exec4()
{
PROCESS="4"
arrayexec4=( "$SPB_MPI.$PROCESS" "$BTB_MPI.$PROCESS" "$CGB_MPI.$PROCESS"
"$EPB_MPI.$PROCESS" "$FTB_MPI.$PROCESS" "$ISB_MPI.$PROCESS"
"$LUB_MPI.$PROCESS" "$MGB_MPI.$PROCESS" )
```

Continuação do *script.*

```
for((CONT4=1;CONT4<41;CONT4++))
        do
                for((AUX4=0;AUX4<${#arrayexec4[@]};AUX4++))
                do
                  LOG4=(`echo ${arrayexec4[$AUX4]}| awk -F/ '{print $7}'`)
                        echo Bencmark: $LOG4
                        echo Contador: $CONT4
                  echo "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
" >> /home/wcp/logs/$LOG4.log
                        echo Start in `date +%H:%M:%S--%d/%m/%Y` 2>&1>>
/home/wcp/logs/$LOG4.log
                        mpirun -np $PROCESS -machinefile /etc/nodefile
${arrayexec4[$AUX4]} >> /home/wcp/logs/$LOG4.log
                        echo Finish in `date +%H:%M:%S--%d/%m/%Y` 2>&1>>
/home/wcp/logs/$LOG4.log
                        echo "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
/-/-/-/-" >> /home/wcp/logs/$LOG4.log
                  sleep 1;
                done
        done
}
exec8()
{
PROCESS="8"
arrayexec8=( "$CGB_MPI.$PROCESS" "$EPB_MPI.$PROCESS" "$FTB_MPI.$PROCESS"
"$ISB_MPI.$PROCESS" "$LUB_MPI.$PROCESS" "$MGB_MPI.$PROCESS" )

for((CONT8=1;CONT8<41;CONT8++))
        do
                for((AUX8=0;AUX8<${#arrayexec8[@]};AUX8++))
                do
                  LOG8=(`echo ${arrayexec8[$AUX8]}| awk -F/ '{print $7}'`)
                  echo "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
" >> /home/wcp/logs/$LOG8.log
                        echo Start in `date +%H:%M:%S--%d/%m/%Y` 2>&1>>
/home/wcp/logs/$LOG8.log
                        echo Bencmark: $LOG8
                        echo Contador: $CONT8
```

Continuação do *script.*

```
                        mpirun   -np   $PROCESS   -machinefile   /etc/nodefile
${arrayexec8[$AUX8]}  >> /home/wcp/logs/$LOG8.log
                        echo  Finish  in  `date  +%H:%M:%S--%d/%m/%Y`  2>&1>>
/home/wcp/logs/$LOG8.log
                        echo   "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
/-/-/-/-" >> /home/wcp/logs/$LOG8.log
                  sleep 1;
                done
        done
}
exec9()
{
PROCESS="9"
arrayexec9=( "$SPB_MPI.$PROCESS" "$BTB_MPI.$PROCESS" "$EPB_MPI.$PROCESS" )

for((CONT9=1;CONT9<41;CONT9++))
        do
                for((AUX9=0;AUX9<${#arrayexec9[@]};AUX9++))
                do
                  LOG9=(`echo ${arrayexec9[$AUX9]}| awk -F/ '{print $7}'`)
                        echo   "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
/-/-/-/-" >> /home/wcp/logs/$LOG9.log
                        echo  Start  in  `date  +%H:%M:%S--%d/%m/%Y`  2>&1>>
/home/wcp/logs/$LOG9.log
                  echo Bencmark: $LOG9
                        echo Contador: $CONT9
                        mpirun   -np   $PROCESS   -machinefile   /etc/nodefile
${arrayexec9[$AUX9]} >> /home/wcp/logs/$LOG9.log
                  echo   Finish   in   `date   +%H:%M:%S--%d/%m/%Y`   2>&1>>
/home/wcp/logs/$LOG9.log
                        echo   "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
/-/-/-/-" >> /home/wcp/logs/$LOG9.log
                        sleep 1;
                done
        done
}
exec16()
{
```

Continuação do *script.*

```
PROCESS="16"
arrayexec16=( "$BTB_MPI.$PROCESS" "$CGB_MPI.$PROCESS" "$FTB_MPI.$PROCESS"
"$ISB_MPI.$PROCESS" "$LUB_MPI.$PROCESS" "$MGB_MPI.$PROCESS"
"$SPB_MPI.$PROCESS" "$EPB_MPI.$PROCESS" )
for((CONT16=1;CONT16<41;CONT16++))
        do
                for((AUX16=0;AUX16<${#arrayexec16[@]};AUX16++))
                do
                  LOG16=(`echo ${arrayexec16[$AUX16]}| awk -F/ '{print
$7}'`)
                        echo "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
/-/-/-/-" >> /home/wcp/logs/$LOG16.log
                  echo Start in `date +%H:%M:%S--%d/%m/%Y` 2>&1>>
/home/wcp/logs/$LOG16.log
                  echo Bencmark: $LOG16
                        echo Contador: $CONT16
                        mpirun -np $PROCESS -machinefile /etc/nodefile
${arrayexec16[$AUX16]} >> /home/wcp/logs/$LOG16.log
                  echo Finish in `date +%H:%M:%S--%d/%m/%Y` 2>&1>>
/home/wcp/logs/$LOG16.log
                        echo "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
/-/-/-/-" >> /home/wcp/logs/$LOG16.log
                        sleep 1;
                done
        done
}
for((i=0;i<${#array[@]};i++))
do
      ${array[$i]}
      echo "============================================="
      echo "##############################################"
      echo "#                                    "
      echo "#             END ${array[$i]}               "
      echo "#                                    "
      echo "##############################################"
      echo "============================================="
sleep 1;
done
```

APÊNDICE D. SCRIPT PARA EXECUÇÃO DO BENCHMARK NPB-OMP

```
#!/bin/bash
#Carlos Alberto Franco Maron; Dalvan Griebler
#Script para execucao dos benchmarks OpenMP.
# Start in 23-05-2014
CONT=1
BTB_OMP="/home/wcp/NPB3.3/NPB3.3-OMP/bin/bt.B"
CGB_OMP="/home/wcp/NPB3.3/NPB3.3-OMP/bin/cg.B"
DCB_OMP="/home/wcp/NPB3.3/NPB3.3-OMP/bin/dc.B"
EPB_OMP="/home/wcp/NPB3.3/NPB3.3-OMP/bin/ep.B"
FTB_OMP="/home/wcp/NPB3.3/NPB3.3-OMP/bin/ft.B"
ISB_OMP="/home/wcp/NPB3.3/NPB3.3-OMP/bin/is.B"
LUB_OMP="/home/wcp/NPB3.3/NPB3.3-OMP/bin/lu.B"
MGB_OMP="/home/wcp/NPB3.3/NPB3.3-OMP/bin/mg.B"
SPB_OMP="/home/wcp/NPB3.3/NPB3.3-OMP/bin/sp.B"
UAB_OMP="/home/wcp/NPB3.3/NPB3.3-OMP/bin/ua.B"

array=( "btb_omp" "cgb_omp" "dcb_omp" "epb_omp" "ftb_omp" "isb_omp"
"lub_omp" "mgb_omp" "spb_omp" "uab_omp" )

btb_omp()
{
export OMP_NUM_THREADS=$THREADS
echo        "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-"        >>
/home/wcp/logs/bt.B.$OMP_NUM_THREADS.log
echo       Start       in       `date       +%H:%M:%S--%d/%m/%Y`       2>&1>>
/home/wcp/logs/bt.B.$OMP_NUM_THREADS.log
$BTB_OMP >> /home/wcp/logs/bt.B.$OMP_NUM_THREADS.log
echo       Finish       in       `date       +%H:%M:%S--%d/%m/%Y`       2>&1>>
/home/wcp/logs/bt.B.$OMP_NUM_THREADS.log
echo        "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-"        >>
/home/wcp/logs/bt.B.$OMP_NUM_THREADS.log
}
cgb_omp()
{
export OMP_NUM_THREADS=$THREADS
echo        "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-"        >>
/home/wcp/logs/cg.B.$OMP_NUM_THREADS.log
echo       Start       in       `date       +%H:%M:%S--%d/%m/%Y`       2>&1>>
```

Continuação do *script.*

```
/home/wcp/logs/cg.B.$OMP_NUM_THREADS.log
$CGB_OMP >> /home/wcp/logs/cg.B.$OMP_NUM_THREADS.log
echo      Finish      in      `date      +%H:%M:%S--%d/%m/%Y`      2>&1>>
/home/wcp/logs/cg.B.$OMP_NUM_THREADS.log
echo       "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-"       >>
/home/wcp/logs/cg.B.$OMP_NUM_THREADS.log
}
dcb_omp()
{
export OMP_NUM_THREADS=$THREADS
echo       "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-"       >>
/home/wcp/logs/dc.B.$OMP_NUM_THREADS.log
echo      Start      in      `date      +%H:%M:%S--%d/%m/%Y`      2>&1>>
/home/wcp/logs/dc.B.$OMP_NUM_THREADS.log
$DCB_OMP >> /home/wcp/logs/dc.B.$OMP_NUM_THREADS.log
echo      Finish      in      `date      +%H:%M:%S--%d/%m/%Y`      2>&1>>
/home/wcp/logs/cg.B.$OMP_NUM_THREADS.log
echo       "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-"       >>
/home/wcp/logs/dc.B.$OMP_NUM_THREADS.log
}
epb_omp()
{
export OMP_NUM_THREADS=$THREADS
echo       "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-"       >>
/home/wcp/logs/ep.B.$OMP_NUM_THREADS.log
echo      Start      in      `date      +%H:%M:%S--%d/%m/%Y`      2>&1>>
/home/wcp/logs/ep.B.$OMP_NUM_THREADS.log
$EPB_OMP >> /home/wcp/logs/ep.B.$OMP_NUM_THREADS.log
echo      Finish      in      `date      +%H:%M:%S--%d/%m/%Y`      2>&1>>
/home/wcp/logs/ep.B.$OMP_NUM_THREADS.log
echo       "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-"       >>
/home/wcp/logs/ep.B.$OMP_NUM_THREADS.log
}
ftb_omp()
{
export OMP_NUM_THREADS=$THREADS
echo       "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-"       >>
/home/wcp/logs/ft.B.$OMP_NUM_THREADS.log
```

Continuação do *script.*

```
echo       Start       in       `date       +%H:%M:%S--%d/%m/%Y`       2>&1>>
/home/wcp/logs/ft.B.$OMP_NUM_THREADS.log
$FTB_OMP >> /home/wcp/logs/ft.B.$OMP_NUM_THREADS.log
echo       Finish      in       `date       +%H:%M:%S--%d/%m/%Y`       2>&1>>
/home/wcp/logs/ft.B.$OMP_NUM_THREADS.log
echo        "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-"        >>
/home/wcp/logs/ft.B.$OMP_NUM_THREADS.log
}
isb_omp()
{
export OMP_NUM_THREADS=$THREADS
echo        "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-"        >>
/home/wcp/logs/is.B.$OMP_NUM_THREADS.log
echo       Start       in       `date       +%H:%M:%S--%d/%m/%Y`       2>&1>>
/home/wcp/logs/is.B.$OMP_NUM_THREADS.log
$ISB_OMP >> /home/wcp/logs/is.B.$OMP_NUM_THREADS.log
echo       Finish      in       `date       +%H:%M:%S--%d/%m/%Y`       2>&1>>
/home/wcp/logs/is.B.$OMP_NUM_THREADS.log
echo        "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-"        >>
/home/wcp/logs/is.B.$OMP_NUM_THREADS.log
}
lub_omp()
{
export OMP_NUM_THREADS=$THREADS
echo        "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-"        >>
/home/wcp/logs/lu.B.$OMP_NUM_THREADS.log
echo       Start       in       `date       +%H:%M:%S--%d/%m/%Y`       2>&1>>
/home/wcp/logs/lu.B..$OMP_NUM_THREADS.log
$LUB_OMP >> /home/wcp/logs/lu.B.$OMP_NUM_THREADS.log
echo       Finish      in       `date       +%H:%M:%S--%d/%m/%Y`       2>&1>>
/home/wcp/logs/lu.B.$OMP_NUM_THREADS.log
echo        "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-"        >>
/home/wcp/logs/lu.B.$OMP_NUM_THREADS.log
}
mgb_omp()
{
export OMP_NUM_THREADS=$THREADS
echo        "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-"        >>
```

Continuação do *script.*

```
/home/wcp/logs/mg.B.$OMP_NUM_THREADS.log
echo       Start       in       `date       +%H:%M:%S--%d/%m/%Y`       2>&1>>
/home/wcp/logs/mg.B.$OMP_NUM_THREADS.log
$MGB_OMP >> /home/wcp/logs/mg.B.$OMP_NUM_THREADS.log
echo       Finish      in       `date       +%H:%M:%S--%d/%m/%Y`       2>&1>>
/home/wcp/logs/mg.B.$OMP_NUM_THREADS.log
echo        "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-"        >>
/home/wcp/logs/mg.B.$OMP_NUM_THREADS.log
}
spb_omp()
{
export OMP_NUM_THREADS=$THREADS
echo        "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-"        >>
/home/wcp/logs/sp.B.$OMP_NUM_THREADS.log
echo       Start       in       `date       +%H:%M:%S--%d/%m/%Y`       2>&1>>
/home/wcp/logs/sp.B.$OMP_NUM_THREADS.log
$SPB_OMP >> /home/wcp/logs/sp.B.$OMP_NUM_THREADS.log
echo       Finish      in       `date       +%H:%M:%S--%d/%m/%Y`       2>&1>>
/home/wcp/logs/sp.B.$OMP_NUM_THREADS.log
echo        "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-"        >>
/home/wcp/logs/sp.B.$OMP_NUM_THREADS.log
}
uab_omp()
{
export OMP_NUM_THREADS=$THREADS
echo        "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-"        >>
/home/wcp/logs/ua.B.$OMP_NUM_THREADS.log
echo       Start       in       `date       +%H:%M:%S--%d/%m/%Y`       2>&1>>
/home/wcp/logs/ua.B.$OMP_NUM_THREADS.log
$UAB_OMP >> /home/wcp/logs/ua.B.$OMP_NUM_THREADS.log
echo       Finish      in       `date       +%H:%M:%S--%d/%m/%Y`       2>&1>>
/home/wcp/logs/ua.B.$OMP_NUM_THREADS.log
echo        "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-"        >>
/home/wcp/logs/ua.B.$OMP_NUM_THREADS.log
}
```

Continuação do *script.*

```
for((THREADS=1;THREADS<5;THREADS++))
do
      for((CONT=1;CONT<41;CONT++))
      do
            for((AUX=0;AUX<${#array[@]};AUX++))
            do
                  echo ${array[$AUX]}
                  echo $CONT
                  echo $THREADS
                  ${array[$AUX]}
                  sleep 1;
            done
      done
done
```

APÊNDICE E. SCRIPT PARA EXECUÇÃO DOS BENCHMARKS DE AVALIAÇAO DO AMBIENTE

```
#Carlos Alberto Franco Maron; Dalvan Griebler
#Script para execucao dos benchmarks para avaliacao de Disco, rede, memoria, processador.
#Benchmarks selecionados: Iozone(Disco), Stream (memoria), Rede(iperf), Processador(codigo_linpack)
# Start in 16-07-2014
CONT=1
for((CONT=1;CONT<41;CONT++))
do
echo "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-" >> /home/wcp/logs/linpack.log
        echo Start in `date +%H:%M:%S--%d/%m/%Y` 2>&1>> /home/wcp/logs/linpack.log
        swapoff -a; swapon -a
        echo $CONT
        echo "LINPACK"
        /home/wcp/benchmarks/linpack >> /home/wcp/logs/linpack.log
        echo "OK!"
```

Continuação do script.

```
        echo     Finish     in     `date     +%H:%M:%S--%d/%m/%Y`     2>&1>>
/home/wcp/logs/linpack.log
        echo     "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-"     >>
/home/wcp/logs/linpack.log
        echo""
        echo     "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-"     >>
/home/wcp/logs/iperf.log
        echo     Start     in     `date     +%H:%M:%S--%d/%m/%Y`     2>&1>>
/home/wcp/logs/iperf.log
        swapoff -a; swapon -a
        echo $CONT
        echo "IPERF"
        iperf -c 10.10.10.10 &>> /home/wcp/logs/iperf.log
        echo "OK"
        echo     Finish     in     `date     +%H:%M:%S--%d/%m/%Y`     2>&1>>
/home/wcp/logs/iperf.log
        echo     "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-"     >>
/home/wcp/logs/iperf.log
        echo""
        echo     "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-"     >>
/home/wcp/logs/iozone.log
        echo     Start     in     `date     +%H:%M:%S--%d/%m/%Y`     2>&1>>
/home/wcp/logs/iozone.log
        swapoff -a; swapon -a
        echo $CONT
        echo "IOZONE"
        iozone -i 0 -i 1 -s#m 100  &>> /home/wcp/logs/iozone.log
        echo "OK"
        echo     Finish     in     `date     +%H:%M:%S--%d/%m/%Y`     2>&1>>
/home/wcp/logs/iozone.log
        echo     "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-"     >>
/home/wcp/logs/iozone.log
        echo""
        echo     "-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-"     >>
/home/wcp/logs/stream.log
done
```

APÊNDICE F. LOGS DOS BENCHMARKS DO AMBIENTE

A seguir será apresentado os resultados obtidos nos logs dos *benchmarks* de testes no ambiente, como o IOzone, STREAM, IPERF, LINPACK.

F.1 Log IOzone

```
-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
Start in 17:19:46--17/07/2014
      Iozone: Performance Test of File I/O
              Version $Revision: 3.397 $
            Compiled for 64 bit mode.
            Build: linux-AMD64
      Contributors:William Norcott, Don Capps, Isom Crawford, Kirby Collins
                   Al Slater, Scott Rhine, Mike Wisner, Ken Goss
                   Steve Landherr, Brad Smith, Mark Kelly, Dr. Alain CYR,
                   Randy Dunlap, Mark Montague, Dan Million, Gavin Brebner,
                   Jean-Marc Zucconi, Jeff Blomberg, Benny Halevy, Dave Boone,
                   Erik Habbinga, Kris Strecker, Walter Wong, Joshua Root,
                   Fabrice Bacchella, Zhenghua Xue, Qin Li, Darren Sawyer.
                   Ben England.
      Run began: Thu Jul 17 17:19:46 2014
      File size set to 524288 KB
      Command line used: iozone -i 0 -i 1 -s#m 100
      Output is in Kbytes/sec
      Time Resolution = 0.000001 seconds.
      Processor cache size set to 1024 Kbytes.
      Processor cache line size set to 32 bytes.
      File stride size set to 17 * record size.
                     random  random    bkwd   record   stride
KB     reclen   write   rewrite  read     reread    read   write    read  rewrite     read
524288    4     1565707 1868999  5346599  5225122
fwrite frewrite   fread  freread
iozone test complete.
Finish in 17:21:40--17/07/2014
```

F.2 Log IPERF

```
-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
Start in 16:08:05--30/07/2014
------------------------------------------------------------
Client connecting to 11.11.11.13, TCP port 5001
TCP window size: 22.9 KByte (default)
------------------------------------------------------------
[  3] local 11.11.11.12 port 55090 connected with 11.11.11.13 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   113 MBytes  94.4 Mbits/sec
Finish in 16:08:15--30/07/2014
-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
```

F.3 Log LINPACK

```
-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
Start in 17:19:05--17/07/2014
Memory required:  125047K.
LINPACK benchmark, Double precision.
Machine precision:  15 digits.
Array size 4000 X 4000.
Average rolled and unrolled performance:

    Reps Time(s) DGEFA   DGESL  OVERHEAD    KFLOPS
----------------------------------------------------
       1  29.70  99.46%   0.13%   0.40%  361144.918
Finish in 17:19:35--17/07/2014
-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
```

F.3 Log STREAM

```
-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
Start in 17:21:41--17/07/2014
-------------------------------------------------------------
STREAM version $Revision: 5.9 $
-------------------------------------------------------------
This system uses 8 bytes per DOUBLE PRECISION word.
-------------------------------------------------------------
Array size = 2000000, Offset = 0
Total memory required = 45.8 MB.
Each test is run 10 times, but only
the *best* time for each is used.
-------------------------------------------------------------
Printing one line per active thread....
-------------------------------------------------------------
Your clock granularity/precision appears to be 1 microseconds.
Each test below will take on the order of 2630 microseconds.
   (= 2630 clock ticks)
Increase the size of the arrays if this shows that
you are not getting at least 20 clock ticks per test.
-------------------------------------------------------------
WARNING -- The above is only a rough guideline.
For best results, please be sure you know the
precision of your system timer.
-------------------------------------------------------------
Function      Rate (MB/s)   Avg time     Min time     Max time
Copy:        8385.4635       0.0043       0.0038       0.0045
Scale:       8024.0167       0.0043       0.0040       0.0045
Add:         9090.8784       0.0056       0.0053       0.0060
Triad:       8933.5548       0.0059       0.0054       0.0066
-------------------------------------------------------------
Solution Validates
-------------------------------------------------------------
Finish in 17:21:41--17/07/2014
```

APÊNDICE G. LOGS DAS APLICAÇÕES PARALELAS

G.1 Log programa BT

```
-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
Start in 16:40:29--23/07/2014
 NAS Parallel Benchmarks (NPB3.3-OMP) - BT Benchmark
 No input file inputbt.data. Using compiled defaults
 Size:  102x 102x 102
 Iterations:  200       dt:   0.0003000
 Number of available threads:     1
 Time step    1
 Time step   20
 Time step   40
 Time step   60
 Time step   80
 Time step  100
 Time step  120
 Time step  140
 Time step  160
 Time step  180
 Time step  200
 Verification being performed for class B
 accuracy setting for epsilon =  0.1000000000000E-07
 Comparison of RMS-norms of residual
           1 0.1423359722929E+04 0.1423359722929E+04 0.7348240154546E-14
           2 0.9933052259015E+02 0.9933052259015E+02 0.4864255696212E-14
           3 0.3564602564454E+03 0.3564602564454E+03 0.3029860801665E-14
           4 0.3248544795908E+03 0.3248544795908E+03 0.2554725169291E-13
           5 0.3270754125466E+04 0.3270754125466E+04 0.7646892227092E-14
 Comparison of RMS-norms of solution error
           1 0.5296984714094E+02 0.5296984714094E+02 0.4024229485897E-15
           2 0.4463289611567E+01 0.4463289611567E+01 0.1989963674771E-15
           3 0.1312257334221E+02 0.1312257334221E+02 0.4060995034457E-15
           4 0.1200692532356E+02 0.1200692532356E+02 0.2958887128106E-15
           5 0.1245957615104E+03 0.1245957615104E+03 0.2281113665977E-15
 Verification Successful
```

Continuação do LOG

```
 BT Benchmark Completed.
 Class           =                        B
 Size            =            102x 102x 102
 Iterations      =                      200
 Time in seconds =                   251.79
 Total threads   =                        1
 Avail threads   =                        1
 Mop/s total     =                  2788.79
 Mop/s/thread    =                  2788.79
 Operation type  =           floating point
 Verification    =               SUCCESSFUL
 Version         =                      3.3
 Compile date    =              17 Jun 2014
 Compile options:
    F77          = gfortran
    FLINK        = $(F77)
    F_LIB        = (none)
    F_INC        = (none)
    FFLAGS       = -O -fopenmp -mcmodel=medium
    FLINKFLAGS   = -O -fopenmp
    RAND         = (none)
 Please send all errors/feedbacks to:
 NPB Development Team
 npb@nas.nasa.gov
Finish in 16:44:43--23/07/2014
-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
```

G.2 Log programa MG

```
-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
Start in 16:53:10--23/07/2014
 NAS Parallel Benchmarks (NPB3.3-OMP) - MG Benchmark
 No input file. Using compiled defaults
 Size:  256x 256x 256  (class B)
 Iterations:                     20
 Number of available threads:     1
 Initialization time:           1.559 seconds
  iter   1
  iter   5
  iter  10
  iter  15
  iter  20
 Benchmark completed
 VERIFICATION SUCCESSFUL
 L2 Norm is  0.1800564401355E-05
 Error is    0.6633011597529E-13
 MG Benchmark Completed.
 Class           =                        B
 Size            =            256x 256x 256
 Iterations      =                       20
 Time in seconds =                    15.83
 Total threads   =                        1
 Avail threads   =                        1
 Mop/s total     =                  1229.65
 Mop/s/thread    =                  1229.65
 Operation type  =           floating point
 Verification    =               SUCCESSFUL
 Version         =                      3.3
 Compile date    =              17 Jun 2014
 Compile options:
    F77          = gfortran
    FLINK        = $(F77)
    F_LIB        = (none)
    F_INC        = (none)
    FFLAGS       = -O -fopenmp -mcmodel=medium
    FLINKFLAGS   = -O -fopenmp
    RAND         = randi8
Finish in 16:53:27--23/07/2014
```

G.3 Log programa SP

```
-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
Start in 16:53:27--23/07/2014
 NAS Parallel Benchmarks (NPB3.3-OMP) - SP Benchmark
 No input file inputsp.data. Using compiled defaults
 Size:  102x 102x 102
 Iterations:  400    dt:    0.0010000
 Number of available threads:     1
 Time step    1
 Time step   20
 Time step   40
 Time step   60
 Time step   80
 Time step  100
 Time step  120
 Time step  140
 Time step  160
 Time step  180
 Time step  200
 Time step  220
 Time step  240
 Time step  260
 Time step  280
 Time step  300
 Time step  320
 Time step  340
 Time step  360
 Time step  380
 Time step  400
 Verification being performed for class B
 accuracy setting for epsilon =  0.1000000000000E-07
 Comparison of RMS-norms of residual
           1 0.6903293579998E+02 0.6903293579998E+02 0.5166989484339E-13
           2 0.3095134488084E+02 0.3095134488084E+02 0.7667559347018E-13
           3 0.4103336647017E+02 0.4103336647017E+02 0.1040704726243E-12
           4 0.3864769009604E+02 0.3864769009604E+02 0.3824106658660E-13
           5 0.5643482272596E+02 0.5643482272596E+02 0.5640544795653E-13
 Comparison of RMS-norms of solution error
           1 0.9810006190188E-02 0.9810006190188E-02 0.4049453876961E-13
```

Continuação do LOG

```
           2 0.1022827905670E-02 0.1022827905670E-02 0.3983496873339E-12
           3 0.1720597911692E-02 0.1720597911692E-02 0.1511054263145E-12
           4 0.1694479428231E-02 0.1694479428231E-02 0.8522719542593E-13
           5 0.1847456263981E-01 0.1847456263981E-01 0.7755970298692E-13
 Verification Successful
 SP Benchmark Completed.
 Class           =                        B
 Size            =            102x 102x 102
 Iterations      =                      400
 Time in seconds =                   227.75
 Total threads   =                        1
 Avail threads   =                        1
 Mop/s total     =                  1558.78
 Mop/s/thread    =                  1558.78
 Operation type  =           floating point
 Verification    =               SUCCESSFUL
 Version         =                      3.3
 Compile date    =              17 Jun 2014
 Compile options:
    F77          = gfortran
    FLINK        = $(F77)
    F_LIB        = (none)
    F_INC        = (none)
    FFLAGS       = -O -fopenmp -mcmodel=medium
    FLINKFLAGS   = -O -fopenmp
    RAND         = (none)
 Please send all errors/feedbacks to:
 NPB Development Team
 npb@nas.nasa.gov
Finish in 16:57:17--23/07/2014
-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
```

G.4 Log programa UA

```
-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
Start in 16:57:17--23/07/2014
 NAS Parallel Benchmarks (NPB3.3-OMP) - UA Benchmark
 No input file inputua.data. Using compiled defaults
 Levels of refinement:              7
 Adaptation frequency:              5
 Time steps:                      200    dt:    0.312500E-03
 CG iterations:                    10
 Heat source radius:           0.0760
 Number of available threads:       1
Step    0: elements refined, merged, total:  1156      0   8093
Step    5: elements refined, merged, total:    67    504   8121
Step   10: elements refined, merged, total:    50    880   7701
Step   15: elements refined, merged, total:   102    280   8170
Step   20: elements refined, merged, total:    62    536   8135
Step   25: elements refined, merged, total:    58    680   7946
Step   30: elements refined, merged, total:    37    704   7589
Step   35: elements refined, merged, total:   128    416   8121
Step   40: elements refined, merged, total:    68    544   8121
Step   45: elements refined, merged, total:    39    616   7855
Step   50: elements refined, merged, total:    97    408   8177
Step   55: elements refined, merged, total:    69    448   8268
Step   60: elements refined, merged, total:    45    880   7813
Step   65: elements refined, merged, total:    76    200   8170
Step   70: elements refined, merged, total:    67    568   8142
Step   75: elements refined, merged, total:    62    624   8030
Step   80: elements refined, merged, total:    59    472   8030
Step   85: elements refined, merged, total:    83    448   8219
Step   90: elements refined, merged, total:    58    624   8079
Step   95: elements refined, merged, total:    30    776   7610
Step  100: elements refined, merged, total:   115    344   8114
Step  105: elements refined, merged, total:    77    480   8233
Step  110: elements refined, merged, total:    48    752   7911
Step  115: elements refined, merged, total:    84    264   8268
Step  120: elements refined, merged, total:    57    536   8198
Step  125: elements refined, merged, total:    62    624   8086
Step  130: elements refined, merged, total:    37    512   7897
Step  135: elements refined, merged, total:    97    416   8212
```

Continuação do LOG UA

```
Step  140: elements refined, merged, total:    59    616   8086
Step  145: elements refined, merged, total:    39    776   7680
Step  150: elements refined, merged, total:   107    408   8072
Step  155: elements refined, merged, total:    67    480   8121
Step  160: elements refined, merged, total:    43   1144   7421
Step  165: elements refined, merged, total:   116    200   8058
Step  170: elements refined, merged, total:    76    552   8107
Step  175: elements refined, merged, total:    66    560   8079
Step  180: elements refined, merged, total:    55    424   8093
Step  185: elements refined, merged, total:    81    464   8254
Step  190: elements refined, merged, total:    67    576   8219
Step  195: elements refined, merged, total:    30    672   7841
 Verification being performed for class B
 accuracy setting for epsilon =  0.1000000000000E-07
 Comparison of temperature integrals
          0.4507561922901E-04 0.4507561922901E-04 0.5306684819777E-13
 Verification Successful
 UA Benchmark Completed.
 Class           =                        B
 Size            =                        7
 Iterations      =                      200
 Time in seconds =                   213.52
 Total threads   =                        1
 Avail threads   =                        1
 Mop/s total     =                    10.34
 Mop/s/thread    =                    10.34
 Operation type  =     coll. point advanced
 Verification    =               SUCCESSFUL
 Version         =                      3.3
 Compile date    =              17 Jun 2014
 Compile options:
    F77          = gfortran
    FLINK        = $(F77)
    F_LIB        = (none)
    F_INC        = (none)
    FFLAGS       = -O -fopenmp -mcmodel=medium
    FLINKFLAGS   = -O -fopenmp
    RAND         = (none)
```

G.5 Log programa CG

```
-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
Start in 16:44:43--23/07/2014
 NAS Parallel Benchmarks (NPB3.3-OMP) - CG Benchmark
 Size:       75000
 Iterations:                    75
 Number of available threads:     1
 Initialization time =           6.075 seconds
   iteration           ||r||                 zeta
        1       0.22762323025366E-12    59.9994751578754
        2       0.84455939289322E-15    21.7627846142536
        3       0.86451102263234E-15    22.2876617043224
        4       0.86169772248241E-15    22.5230738188346
        5       0.87122209570921E-15    22.6275390653892
        6       0.87197718525557E-15    22.6740259189533
        7       0.87383226191852E-15    22.6949056826251
        8       0.88787023980150E-15    22.7044023166872
        9       0.88321652736396E-15    22.7087834345620
       10       0.88090946461777E-15    22.7108351397177
       11       0.88080381638316E-15    22.7118107121341
       12       0.88463150736327E-15    22.7122816240971
       13       0.88438630770811E-15    22.7125122663242
       14       0.88573824525624E-15    22.7126268007594
       15       0.86946518959002E-15    22.7126844161819
       16       0.88046447579561E-15    22.7127137461755
       17       0.87920822938849E-15    22.7127288402000
       18       0.88449594405755E-15    22.7127366848296
       19       0.88745902095650E-15    22.7127407981217
       20       0.87873860878805E-15    22.7127429721364
       21       0.87906399623303E-15    22.7127441294025
       22       0.87756197073811E-15    22.7127447493901
       23       0.87343804279763E-15    22.7127450834533
       24       0.87856249629235E-15    22.7127452643881
       25       0.87501512399146E-15    22.7127453628451
       26       0.88127148778810E-15    22.7127454166517
       27       0.87220515272948E-15    22.7127454461696
       28       0.87814167439566E-15    22.7127454624211
       29       0.88852857605335E-15    22.7127454713974
       30       0.87914495939807E-15    22.7127454763703
```

Continuação LOG CG

31	0.87859189026752E-15	22.7127454791343
32	0.88532589464799E-15	22.7127454806740
33	0.87006618233683E-15	22.7127454815315
34	0.87145465008671E-15	22.7127454820131
35	0.87343875145808E-15	22.7127454822840
36	0.87317608461062E-15	22.7127454824348
37	0.87401406300885E-15	22.7127454825202
38	0.87073210766472E-15	22.7127454825684
39	0.87591336616315E-15	22.7127454825966
40	0.87253350501552E-15	22.7127454826116
41	0.87424275989612E-15	22.7127454826202
42	0.87279830213115E-15	22.7127454826253
43	0.86954422239948E-15	22.7127454826277
44	0.87241285314720E-15	22.7127454826297
45	0.87083236142108E-15	22.7127454826310
46	0.86510041104973E-15	22.7127454826310
47	0.87112121547755E-15	22.7127454826314
48	0.86865990245470E-15	22.7127454826317
49	0.87105089984189E-15	22.7127454826309
50	0.86803338174709E-15	22.7127454826322
51	0.87050943900934E-15	22.7127454826324
52	0.86997010794457E-15	22.7127454826313
53	0.87042722906034E-15	22.7127454826315
54	0.87282055242783E-15	22.7127454826309
55	0.86726140790934E-15	22.7127454826315
56	0.86471252901711E-15	22.7127454826316
57	0.87059321251784E-15	22.7127454826318
58	0.86383217256710E-15	22.7127454826314
59	0.86317324181346E-15	22.7127454826314
60	0.85793827942534E-15	22.7127454826318
61	0.85857990960913E-15	22.7127454826315
62	0.87440405664890E-15	22.7127454826318
63	0.86730106501390E-15	22.7127454826313
64	0.86338483644472E-15	22.7127454826317
65	0.86397143821750E-15	22.7127454826309
66	0.86244746229103E-15	22.7127454826319
67	0.86353571565226E-15	22.7127454826319
68	0.86132573355830E-15	22.7127454826321

Continuação LOG CG

```
       69       0.85589770904341E-15    22.7127454826317
       70       0.86021344008458E-15    22.7127454826326
       71       0.86652304940985E-15    22.7127454826314
       72       0.85856054198012E-15    22.7127454826322
       73       0.86764812167700E-15    22.7127454826316
       74       0.86009609757360E-15    22.7127454826318
       75       0.85919504297487E-15    22.7127454826310
 Benchmark completed
 VERIFICATION SUCCESSFUL
 Zeta is     0.2271274548263E+02
 Error is    0.6256775397791E-15
 CG Benchmark Completed.
 Class           =                        B
 Size            =                    75000
 Iterations      =                       75
 Time in seconds =                    89.02
 Total threads   =                        1
 Avail threads   =                        1
 Mop/s total     =                   614.59
 Mop/s/thread    =                   614.59
 Operation type  =           floating point
 Verification    =               SUCCESSFUL
 Version         =                      3.3
 Compile date    =              17 Jun 2014
 Compile options:
    F77          = gfortran
    FLINK        = $(F77)
    F_LIB        = (none)
    F_INC        = (none)
    FFLAGS       = -O -fopenmp -mcmodel=medium
    FLINKFLAGS   = -O -fopenmp
    RAND         = randi8
 Please send all errors/feedbacks to:
 NPB Development Team
 npb@nas.nasa.gov
Finish in 16:46:18--23/07/2014
```

G.6 Log programa EP

```
-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
Start in 16:46:18--23/07/2014
 NAS Parallel Benchmarks (NPB3.3-OMP) - EP Benchmark
 Number of random numbers generated:      2147483648
 Number of available threads:                      1
EP Benchmark Results:
CPU Time =   74.9168
N = 2^   30
No. Gaussian Pairs =     843345606.
Sums =     4.033815542441498D+04   -2.660669192809233D+04
Counts:
  0     393058470.
  1     375280898.
  2      70460742.
  3       4438852.
  4        105691.
  5           948.
  6             5.
  7             0.
  8             0.
  9             0.
 EP Benchmark Completed.
 Class           =                        B
 Size            =               2147483648
 Iterations      =                        0
 Time in seconds =                    74.92
 Total threads   =                        1
 Avail threads   =                        1
 Mop/s total     =                    28.66
 Mop/s/thread    =                    28.66
 Operation type  = Random numbers generated
 Verification    =               SUCCESSFUL
 Version         =                      3.3
 Compile date    =              17 Jun 2014
 Compile options:
    F77          = gfortran
    FLINK        = $(F77)
    F_LIB        = (none)
```

Continuação do LOG EP

```
    FLINK        = $(F77)
    F_LIB        = (none)
    F_INC        = (none)
    FFLAGS       = -O -fopenmp -mcmodel=medium
    FLINKFLAGS   = -O -fopenmp
    RAND         = randi8
 Please send all errors/feedbacks to:
 NPB Development Team
 npb@nas.nasa.gov
Finish in 16:47:34--23/07/2014
-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
```

G.7 Log programa FT

```
 Start in 06:01:31--24/07/2014
  NAS Parallel Benchmarks (NPB3.3-OMP) - FT Benchmark
  Size                :  512x 256x 256
  Iterations                  :     20
  Number of available threads :      1
  T =    1     Checksum =    5.177643571579D+02    5.077803458597D+02
  T =    2     Checksum =    5.154521291263D+02    5.088249431599D+02
  T =    3     Checksum =    5.146409228650D+02    5.096208912659D+02
  T =    4     Checksum =    5.142378756213D+02    5.101023387619D+02
  T =    5     Checksum =    5.139626667737D+02    5.103976610618D+02
  T =    6     Checksum =    5.137423460082D+02    5.105948019802D+02
  T =    7     Checksum =    5.135547056878D+02    5.107404165783D+02
  T =    8     Checksum =    5.133910925467D+02    5.108576573661D+02
  T =    9     Checksum =    5.132470705390D+02    5.109577278523D+02
  T =   10     Checksum =    5.131197729984D+02    5.110460304483D+02
  T =   11     Checksum =    5.130070319283D+02    5.111252433800D+02
  T =   12     Checksum =    5.129070537032D+02    5.111968077719D+02
  T =   13     Checksum =    5.128182883503D+02    5.112616233064D+02
  T =   14     Checksum =    5.127393733383D+02    5.113203605551D+02
  T =   15     Checksum =    5.126691062021D+02    5.113735928093D+02
  T =   16     Checksum =    5.126064276005D+02    5.114218460548D+02
  T =   17     Checksum =    5.125504076570D+02    5.114656139760D+02
```

Continuação LOG FT

```
T =   18     Checksum =    5.125002331721D+02    5.115053595966D+02
 T =   19     Checksum =    5.124551951846D+02    5.115415130407D+02
 T =   20     Checksum =    5.124146770029D+02    5.115744692211D+02
 Result verification successful
 class = B
 FT Benchmark Completed.
 Class           =                        B
 Size            =            512x 256x 256
 Iterations      =                       20
 Time in seconds =                    71.69
 Total threads   =                        1
 Avail threads   =                        1
 Mop/s total     =                  1284.11
 Mop/s/thread    =                  1284.11
 Operation type  =           floating point
 Verification    =               SUCCESSFUL
 Version         =                      3.3
 Compile date    =              17 Jun 2014
 Compile options:
    F77          = gfortran
    FLINK        = $(F77)
    F_LIB        = (none)
    F_INC        = (none)
    FFLAGS       = -O -fopenmp -mcmodel=medium
    FLINKFLAGS   = -O -fopenmp
    RAND         = randi8


 Please send all errors/feedbacks to:

 NPB Development Team
 npb@nas.nasa.gov


Finish in 06:02:48--24/07/2014
-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
```

G.8 Log programa IS

```
-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
Start in 16:48:51--23/07/2014
 NAS Parallel Benchmarks (NPB3.3-OMP) - IS Benchmark
 Size:  33554432  (class B)
 Iterations:  10
 Number of available threads:  1
   iteration
        1
        2
        3
        4
        5
        6
        7
        8
        9
        10
 IS Benchmark Completed
 Class           =                        B
 Size            =                 33554432
 Iterations      =                       10
 Time in seconds =                     4.23
 Total threads   =                        1
 Avail threads   =                        1
 Mop/s total     =                    79.41
 Mop/s/thread    =                    79.41
 Operation type  =              keys ranked
 Verification    =               SUCCESSFUL
 Version         =                      3.3
 Compile date    =              17 Jun 2014
 Compile options:
    CC           = cc
    CLINK        = $(CC)
    C_LIB        = -lm
    C_INC        = (none)
    CFLAGS       = -O -fopenmp
    CLINKFLAGS   = -O -fopenmp
 Finish in 16:49:00--23/07/2014
```

G.9 Log programa LU

```
-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
Start in 16:49:00--23/07/2014
 NAS Parallel Benchmarks (NPB3.3-OMP) - LU Benchmark
 Size:  102x 102x 102
 Iterations:                    250
 Number of available threads:     1
 Time step    1
 Time step   20
 Time step   40
 Time step   60
 Time step   80
 Time step  100
 Time step  120
 Time step  140
 Time step  160
 Time step  180
 Time step  200
 Time step  220
 Time step  240
 Time step  250
 Verification being performed for class B
 Accuracy setting for epsilon =  0.1000000000000E-07
 Comparison of RMS-norms of residual
           1   0.3553267296998E+04 0.3553267296998E+04 0.1663740739831E-14
           2   0.2621475079531E+03 0.2621475079531E+03 0.2168375328251E-15
           3   0.8833372185095E+03 0.8833372185095E+03 0.2574030287401E-15
           4   0.7781277473943E+03 0.7781277473943E+03 0.2629854912438E-14
           5   0.7308796959255E+04 0.7308796959255E+04 0.7466301555590E-15
 Comparison of RMS-norms of solution error
           1   0.1140117638021E+03 0.1140117638021E+03 0.0000000000000E+00
           2   0.8109896365542E+01 0.8109896365542E+01 0.2190357014854E-15
           3   0.2848059731770E+02 0.2848059731770E+02 0.0000000000000E+00
           4   0.2590539456783E+02 0.2590539456783E+02 0.1371418477915E-15
           5   0.2605490750486E+03 0.2605490750486E+03 0.0000000000000E+00
 Comparison of surface integral
               0.4788716270331E+02 0.4788716270331E+02 0.0000000000000E+00
 Verification Successful
```

Continuação do Log.

```
 LU Benchmark Completed.
 Class           =                        B
 Size            =            102x 102x 102
 Iterations      =                      250
 Time in seconds =                   247.40
 Total threads   =                        1
 Avail threads   =                        1
 Mop/s total     =                  2016.26
 Mop/s/thread    =                  2016.26
 Operation type  =           floating point
 Verification    =               SUCCESSFUL
 Version         =                      3.3
 Compile date    =              17 Jun 2014
 Compile options:
    F77          = gfortran
    FLINK        = $(F77)
    F_LIB        = (none)
    F_INC        = (none)
    FFLAGS       = -O -fopenmp -mcmodel=medium
    FLINKFLAGS   = -O -fopenmp
    RAND         = (none)
 Please send all errors/feedbacks to:
 NPB Development Team
 npb@nas.nasa.gov
Finish in 16:53:10--23/07/2014
-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
```

APÊNDICE H. ARTIGO ERAD 2014

Comparação das Ferramentas OpenNebula e OpenStack em Nuvem Composta de Estações de Trabalho**

Carlos Alberto Franco Maron[1], Dalvan Griebler[2], Claudio Schepke[3]

[1]Curso Superior de Tecnologia em Redes de Computadores – Faculdade Três de Maio (SETREM) – Três de Maio – RS – Brasil

[2]Programa de Pós-Graduação em Ciência da Computação – Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS) – Porto Alegre – RS – Brasil

[3]Campus Alegrete – Universidade Federal do Pampa (UNIPAMPA) – Alegrete – RS – Brasil

francocaam@gmail.com,dalvan.griebler@acad.pucrs.br,
claudioschepke@unipampa.edu.br

Resumo. *Ferramentas de computação em nuvem para o modelo de serviço IaaS como o OpenNebula e o OpenStack são implantadas em grandes centros de processamento. O objetivo deste trabalho é investigar e comparar o comportamento delas em um ambiente mais restrito, como o de estações de trabalho. Os resultados mostraram que a ferramenta OpenNebula leva vantagem nas principais características avaliadas.*

1. Introdução

A Computação em nuvem (CN) possibilita acessar recursos computacionais (por exemplo, servidores, armazenamento, redes, serviços e aplicações) de maneira prática e sob demanda, rapidamente e que podem ser liberados para o usuário sem qualquer envolvimento gerencial [Mell e T. Grance 2011]. Isso pode ser muito importante para agilizar o desenvolvimento do trabalho, reduzir custos, facilitar o emprego de recursos de alto processamento, evitar gastos com manutenção e licenças de *software*.

As nuvens podem ser caracterizadas em diferentes tipos (pública, privada e híbrida) e diferentes modelos de serviços (IaaS - *Infrastructure as a Service*, PaaS - *Plataform as a Service* e SaaS - *Software as a Service*) [Marks e Lozano 2010, Veras 2011]. Neste trabalho, o escopo são as ferramentas *open source* para implantação de nuvem que suportam o modelo IaaS.

A motivação deste trabalho é que as empresas ou Instituições de ensino possuem equipamentos com abundância de processamento para tarefas menos exigentes. Para alguns casos, as tarefas em determinadas situações não chegam a ocupar todo o potencial de processamento que os equipamentos oferecem. Com isso, toda esta capacidade computacional poderia ser usada para a implantação de uma nuvem privada, oferecendo processamento para outros recursos dentro do ambiente em que elas se encontram.

** Trabalho realizado com equipamentos do Laboratório de Redes de Computadores da SETREM

A dificuldade não está somente em implantar uma nuvem, mas também em escolher a ferramenta mais apropriada para o projeto de redes. Neste contexto, o objetivo deste trabalho é caracterizar, estudar e comparar as ferramentas OpenStack e OpenNebula, escolhidas através de análise prévia [Thomé, Hengtes, Griebler 2013], evidenciando o cenário com uso de estações de trabalho convencionais.

Este artigo está dividido em 4 seções. A Seção 2 descreve alguns trabalhos relacionados. Na Seção 3 são apresentadas as análises realizadas para diferentes cenários, avaliando qual a ferramenta mais adequada em cada caso. Por fim, na Seção 4, é relatada a conclusão do trabalho.

2. Trabalhos Relacionados

Alguns trabalhos relacionados abordando avaliações de ferramentas para computação em nuvem são encontrados na literatura.

O trabalho [Laszewski et al 2012] teve como objetivo comparar a escalabilidade das ferramentas OpenNebula, OpenStack, Eucalyptus e Nimbus. Neste sentido, o ambiente de teste foi o cluster India, composto por processadores Intel Xeon X5570 com 24GB de memória, disco de 500GB com 7200RPM (3GB/s) e uma rede de interconexão de 1Gb/s. Os resultados mostraram que o OpenNebula teve a melhor escalabilidade, permitindo o provisionamento de 148 VM de uma só vez e sem nenhum erro. A segunda melhor ferramenta foi o OpenStack, conseguindo o provisionamento de no máximo 64 VM (enviando no máximo 10 de cada vez) com cerca de 50% de falhas.

Em [Khurshid et al 2009] o objetivo é avaliar o desempenho de um ambiente de CN que utiliza a ferramenta Open Cirrus. Para esta avaliação, foram utilizados o Planetab e o Emulab, simulando usuários distribuídos e instalações da nuvem. O cluster usado para os testes possui 128 nodos HP DL160 com processador dual quad core, 16GB de memória, 2TB de disco e uma rede 1Gb/s. Com os experimentos realizados, foi descoberto que configurações internas e características da rede afetaram o desempenho na transferência de dados.

Como pode ser visto, ambos as pesquisas usam um ambiente robusto e especializado com alto poder computacional. A diferença diante deles, é que este trabalho tem o propósito de avaliar o comportamento das ferramentas em um ambiente computacional composto de estações de trabalho. Além disso, inicialmente o foco é a avaliação das características de instalação e configuração das nuvens (OpenNebula e OpenStack), para que no futuro possa ser testado o desempenho delas.

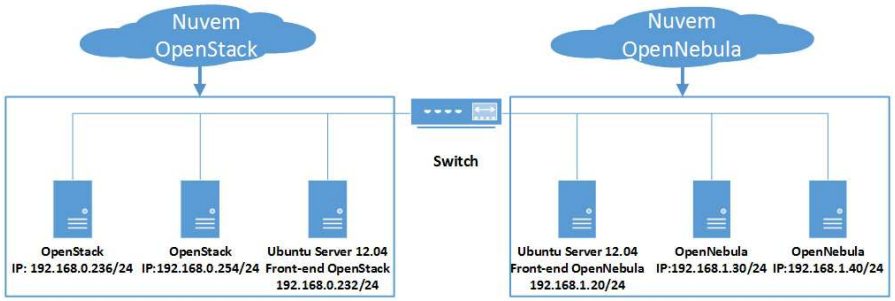
3. Análise Comparativa

As características avaliadas foram as mesmas usadas no estudo [Thomé, Hengtes, Griebler 2013], onde é realizada uma avaliação das principais ferramentas do modelo IaaS, baseando-se nas informações que a literatura oferece. No entanto, este trabalho busca testar duas delas em um cenário controlado, no sentido de avaliar e comparar o comportamento em um ambiente de estações de trabalho, além disso, verificar se os resultados obtidos foram coerentes com a literatura.

As nuvens criadas estão representadas na Figura 1. O ambiente usado para a avaliação das ferramentas OpenNebula e OpenStack é composto por estações de trabalho com capacidade de processamento similares. Sendo assim, as características técnicas são:

Processadores Intel Core i5 650 com 3.20 Ghz, memória RAM de 4 GB DDR3 de 1333 Mhz, 500 GB operando em Sata II, e placa de rede on-board 10/100 Mbits. Conforme a Figura 1, cada equipamento está conectado fisicamente em um *Switch*, com capacidade de até 100 Mbits/s. Embora tenha se utilizado o mesmo *Switch* nos experimentos, ambas as nuvens (Nuvem OpenStack e Nuvem OpenNebula) operam em redes diferentes, ou seja, elas não se comunicam.

Figura 1: Ambiente físico para testes



O procedimento de avaliação das características foi realizado da seguinte forma: **Interface** – avalia a forma de acesso às ferramentas. **Gerenciamento de energia** – avalia como a ferramenta trata o gerenciamento de energia para o uso de seus recursos e a forma de como isto é feito, com uso de recursos próprios ou com ferramentas em paralelo. **Balanceamento de carga** – como cada ferramenta gerencia a distribuição de carga de trabalho entre os nodos e, se isto acontece. **Rede** – como cada ferramenta aborda os tipos de conexões com as máquinas virtuais. **Armazenamento** – trata dos sistemas de armazenamento utilizados por cada ferramenta. **Monitoramento** – analisa se tem a possibilidade de exercer algum tipo de monitoramento, como disponibilidade, uso computacional dos componentes da nuvem. **Integração** – busca avaliar as formas de integração com outros tipos de ferramentas de computação em nuvem. **Virtualização** – avalia os virtualizadores suportados em ambas as ferramentas. **Segurança** – avalia os tipos de segurança que podem ser empregadas nas ferramentas. **Escalabilidade** – avalia se é possível adicionar novos nodos sem afetar a nuvem. **Tolerância a falhas** – avaliar se a tolerância a falhas existe e funciona.

Tabela 1: Análise Comparativa

Característica Avaliada	Melhor Avaliação	Coerente com a bibliografia
Interface	OpenNebula	Sim
Gerenciamento de energia	OpenNebula	Indeterminado
Balanceamento de carga	OpenStack	Sim
Rede	Ambos	Parcialmente
Armazenamento	Ambos	Parcialmente
Monitoramento	OpenNebula	Sim

Integração	Ambos	Indeterminado
Virtualização	Ambos	Parcialmente
Segurança	OpenNebula	Sim
Adição de nodos	Ambos	Sim
Tolerância a falhas	OpenNebula	Parcialmente

Com relação a avaliação dos itens o *OpenNebula* apresenta uma interface mais objetiva em relação ao *OpenStack* que possui termos de difícil compreensão.

OpenNebula apresentou uma solução compatível com a estrutura mas devido a escassez de informações o recurso não pode ser implementado. *OpenStack* apresenta uma solução compatível somente com processadores da linha Intel Xeon, sendo divergente do ambiente de testes.

O balanceamento de carga realizado pelo *OpenStack* é com base em um ferramenta que monitora o estado atual de carga dos processadores nos nodos. *OpenNebula* apresentou uma solução muito genérica, distribuindo por igual a quantidade de máquinas virtuais no ambiente.

No monitoramento o *OpenNebula* apresentou o método de monitoramento através de graficos e exibindo uma quantidade maior de informações monitoradas.

Na segurança a ferramenta *OpenNebula* permite a criação de ACL's, sendo possível aplicar um nível maior de permissões aos usuário fazem uso dos recursos da ferramenta.

No item tolerância a falhas o *OpenNebula* na remoção intencional de um nodo, o controlador da nuvem conseguiu se sobressair, dividindo os recursos em outros nodos. A ferramenta OpenStack apresenta as mesmas características porém era necessário a reestruturação de todo o ambiente para simulação.

4. Conclusão

A partir dos testes realizados e da análise comparativa pode-se constatar que OpenNebula é a ferramenta mais apropriada para a utilização em ambientes formados por estações de trabalho. Além de apresentar teoricamente as melhores características, estas puderam ser de fato comprovadas através de uma implantação prática, obtendo resultados melhores em relação a ferramenta Openstack. Em termos de trabalhos futuros, pretende-se avaliar ainda de forma prática o quesito gerenciamento de energia, integração e o desempenho nestes ambientes.

5. Referências

E. A. Marks e B. Lozano. Executive's Guide to Cloud Computing. 1º Ed. New Jersey: Published by John Wiley & Sons, Inc., 2010, p.285.

P. Mell e T. Grance. The NIST Definition of Cloud Computing. Gaithersburg: National Institute of Standards and Technology, 2011, p.7.

B. Thomé, E. Hentges e D. Griebler. Computação em Nuvem: Análise Comparativa de Ferramentas Open Source para IaaS. Anais da 11ª Escola Regional de Redes de Computadores, 2013.

M. Veras. Virtualização: Componente Central do Datacenter. 1º Ed. Rio de Janeiro: Brasport Livros e Multimidia Ltda, 2011, p. 364.

G. V. Laszewski, J. Diaz, F. Wang e G. C. Fox. Comparison of Multiple Cloud Frameworks. 2012 IEEE Fifth International Conference on Cloud Computing. Washington. 2012. p.734-741.

A. Khurshid, A. Al-nayeem e I. Gupta. Performance Evaluation of the Illinois Cloud Computing Testbed. [S.I], Urbana-Champaign: Illinois Digital Environment for Access to Learning and Scholarsip, 2009, p.12.

APÊNDICE I. TESTE ESTATÍSTICO DAS APLICAÇÕES PARALELAS – NPB-OMP

I.1 Teste estatístico para programa NPB-OMP [BT]

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_OpenMP_BT1	252,2770	40	,63237	,09999
	OPENSTACK_OpenMP_BT1	266,4293	40	1,69541	,26807
Pair 2	NATIVO_OpenMP_BT2	137,0985	40	,39343	,06221
	OPENSTACK_OpenMP_BT2	147,8230	40	1,27493	,20158
Pair 3	NATIVO_OpenMP_BT3	142,6798	40	,28156	,04452
	OPENSTACK_OpenMP_BT3	161,9683	40	1,47789	,23367
Pair 4	NATIVO_OpenMP_BT4	131,0163	40	,66210	,10469
	OPENSTACK_OpenMP_BT4	157,4703	40	9,05994	1,43250

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_OpenMP_BT1 - OPENSTACK_OpenMP_BT1	-14,15225	1,81237	,28656	-14,73187	-13,57263	-49,387	39	,000
Pair 2	NATIVO_OpenMP_BT2 - OPENSTACK_OpenMP_BT2	-10,72450	1,34085	,21201	-11,15332	-10,29568	-50,586	39	,000
Pair 3	NATIVO_OpenMP_BT3 - OPENSTACK_OpenMP_BT3	-19,28850	1,46769	,23206	-19,75789	-18,81911	-83,118	39	,000
Pair 4	NATIVO_OpenMP_BT4 - OPENSTACK_OpenMP_BT4	-26,45400	9,10679	1,43991	-29,36649	-23,54151	-18,372	39	,000

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_OpenMP_BT1	252,2770	40	,63237	,09999
	OPENNEBULA_OpenMP_BT1	262,7628	40	1,19376	,18875
Pair 2	NATIVO_OpenMP_BT2	137,0985	40	,39343	,06221
	OPENNEBULA_OpenMP_BT2	143,3243	40	,34334	,05429
Pair 3	NATIVO_OpenMP_BT3	142,6798	40	,28156	,04452
	OPENNEBULA_OpenMP_BT3	150,0035	40	,60216	,09521
Pair 4	NATIVO_OpenMP_BT4	131,0163	40	,66210	,10469
	OPENNEBULA_OpenMP_BT4	140,0240	40	1,44937	,22917

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_OpenMP_BT1 - OPENNEBULA_OpenMP_BT1	-10,48575	1,41306	,22342	-10,93767	-10,03383	-46,932	39	,000
Pair 2	NATIVO_OpenMP_BT2 - OPENNEBULA_OpenMP_BT2	-6,22575	,41798	,06609	-6,35943	-6,09207	-94,203	39	,000
Pair 3	NATIVO_OpenMP_BT3 - OPENNEBULA_OpenMP_BT3	-7,32375	,69698	,11020	-7,54665	-7,10085	-66,458	39	,000
Pair 4	NATIVO_OpenMP_BT4 - OPENNEBULA_OpenMP_BT4	-9,00775	1,70137	,26901	-9,55187	-8,46363	-33,485	39	,000

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	OPENSTACK_OpenMP_BT1	266,4293	40	1,69541	,26807
	OPENNEBULA_OpenMP_BT1	262,7628	40	1,19376	,18875
Pair 2	OPENSTACK_OpenMP_BT2	147,8230	40	1,27493	,20158
	OPENNEBULA_OpenMP_BT2	143,3243	40	,34334	,05429
Pair 3	OPENSTACK_OpenMP_BT3	161,9683	40	1,47789	,23367
	OPENNEBULA_OpenMP_BT3	150,0035	40	,60216	,09521
Pair 4	OPENSTACK_OpenMP_BT4	157,4703	40	9,05994	1,43250
	OPENNEBULA_OpenMP_BT4	140,0240	40	1,44937	,22917

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	OPENSTACK_OpenMP_BT1 - OPENNEBULA_OpenMP_BT1	3,66650	1,67033	,26410	3,13230	4,20070	13,883	39	,000
Pair 2	OPENSTACK_OpenMP_BT2 - OPENNEBULA_OpenMP_BT2	4,49875	1,39050	,21986	4,05405	4,94345	20,462	39	,000
Pair 3	OPENSTACK_OpenMP_BT3 - OPENNEBULA_OpenMP_BT3	11,96475	1,94042	,30681	11,34417	12,58533	38,998	39	,000
Pair 4	OPENSTACK_OpenMP_BT4 - OPENNEBULA_OpenMP_BT4	17,44625	9,18357	1,45205	14,50920	20,38330	12,015	39	,000

I.2 Teste estatístico para programa NPB-OMP [SP]

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_OpenMP_SP1	227,2330	40	,28258	,04468
	OPENSTACK_OpenMP_SP1	244,3827	40	2,42531	,38347
Pair 2	NATIVO_OpenMP_SP2	143,2223	40	,32143	,05082
	OPENSTACK_OpenMP_SP2	159,0157	40	1,40688	,22245
Pair 3	NATIVO_OpenMP_SP3	160,8738	40	,27485	,04346
	OPENSTACK_OpenMP_SP3	181,0697	40	,91407	,14453
Pair 4	NATIVO_OpenMP_SP4	176,8323	40	,31298	,04949
	OPENSTACK_OpenMP_SP4	192,3585	40	,51156	,08088

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_OpenMP_SP1 - OPENSTACK_OpenMP_SP1	-17,14975	2,49134	,39392	-17,94652	-16,35298	-43,537	39	,000
Pair 2	NATIVO_OpenMP_SP2 - OPENSTACK_OpenMP_SP2	-15,79350	1,48623	,23499	-16,26882	-15,31818	-67,208	39	,000
Pair 3	NATIVO_OpenMP_SP3 - OPENSTACK_OpenMP_SP3	-20,19600	,88749	,14032	-20,47983	-19,91217	-143,924	39	,000
Pair 4	NATIVO_OpenMP_SP4 - OPENSTACK_OpenMP_SP4	-15,52625	,58625	,09269	-15,71374	-15,33876	-167,500	39	,000

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_OpenMP_SP1	227,2330	40	,28258	,04468
	OPENNEBULA_OpenMP_SP1	238,4440	40	,65896	,10419
Pair 2	NATIVO_OpenMP_SP2	143,2223	40	,32143	,05082
	OPENNEBULA_OpenMP_SP2	152,4090	40	,52220	,08257
Pair 3	NATIVO_OpenMP_SP3	160,8738	40	,27485	,04346
	OPENNEBULA_OpenMP_SP3	172,5785	40	1,22161	,19315
Pair 4	NATIVO_OpenMP_SP4	176,8323	40	,31298	,04949
	OPENNEBULA_OpenMP_SP4	187,6783	40	,74492	,11778

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_OpenMP_SP1 - OPENNEBULA_OpenMP_SP1	-11,21100	,76611	,12113	-11,45601	-10,96599	-92,552	39	,000
Pair 2	NATIVO_OpenMP_SP2 - OPENNEBULA_OpenMP_SP2	-9,18675	,58815	,09299	-9,37485	-8,99865	-98,789	39	,000
Pair 3	NATIVO_OpenMP_SP3 - OPENNEBULA_OpenMP_SP3	-11,70475	1,21520	,19214	-12,09339	-11,31611	-60,918	39	,000
Pair 4	NATIVO_OpenMP_SP4 - OPENNEBULA_OpenMP_SP4	-10,84600	,74765	,11821	-11,08511	-10,60689	-91,749	39	,000

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	OPENSTACK_OpenMP_SP1	244,3827	40	2,42531	,38347
	OPENNEBULA_OpenMP_SP1	238,4440	40	,65896	,10419
Pair 2	OPENSTACK_OpenMP_SP2	159,0157	40	1,40688	,22245
	OPENNEBULA_OpenMP_SP2	152,4090	40	,52220	,08257
Pair 3	OPENSTACK_OpenMP_SP3	181,0697	40	,91407	,14453
	OPENNEBULA_OpenMP_SP3	172,5785	40	1,22161	,19315
Pair 4	OPENSTACK_OpenMP_SP4	192,3585	40	,51156	,08088
	OPENNEBULA_OpenMP_SP4	187,6783	40	,74492	,11778

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	OPENSTACK_OpenMP_SP1 - OPENNEBULA_OpenMP_SP1	5,93875	2,10199	,33235	5,26650	6,61100	17,869	39	,000
Pair 2	OPENSTACK_OpenMP_SP2 - OPENNEBULA_OpenMP_SP2	6,60675	1,36912	,21648	6,16888	7,04462	30,519	39	,000
Pair 3	OPENSTACK_OpenMP_SP3 - OPENNEBULA_OpenMP_SP3	8,49125	1,98061	,31316	7,85782	9,12468	27,115	39	,000
Pair 4	OPENSTACK_OpenMP_SP4 - OPENNEBULA_OpenMP_SP4	4,68025	,66582	,10528	4,46731	4,89319	44,457	39	,000

I.3 Teste estatístico para programa NPB-OMP [UA]

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_OpenMP_UA1	213,4100	40	,33096	,05233
	OPENSTACK_OpenMP_UA1	226,1065	40	2,45310	,38787
Pair 2	NATIVO_OpenMP_UA2	128,4160	40	,09674	,01530
	OPENSTACK_OpenMP_UA2	138,0143	40	,91896	,14530
Pair 3	NATIVO_OpenMP_UA3	136,0622	40	,18847	,02980
	OPENSTACK_OpenMP_UA3	144,2885	40	,45273	,07158
Pair 4	NATIVO_OpenMP_UA4	115,7277	40	,15874	,02510
	OPENSTACK_OpenMP_UA4	126,8815	40	,71785	,11350

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_OpenMP_UA1 - OPENSTACK_OpenMP_UA1	-12,69650	2,50017	,39531	-13,49609	-11,89691	-32,118	39	,000
Pair 2	NATIVO_OpenMP_UA2 - OPENSTACK_OpenMP_UA2	-9,59825	,89463	,14145	-9,88437	-9,31213	-67,854	39	,000
Pair 3	NATIVO_OpenMP_UA3 - OPENSTACK_OpenMP_UA3	-8,22625	,47259	,07472	-8,37739	-8,07511	-110,090	39	,000
Pair 4	NATIVO_OpenMP_UA4 - OPENSTACK_OpenMP_UA4	-11,15375	,71490	,11303	-11,38238	-10,92512	-98,675	39	,000

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_OpenMP_UA1	213,4100	40	,33096	,05233
	OPENNEBULA_OpenMP_UA1	220,8813	40	,49833	,07879
Pair 2	NATIVO_OpenMP_UA2	128,4160	40	,09674	,01530
	OPENNEBULA_OpenMP_UA2	133,0998	40	,23000	,03637
Pair 3	NATIVO_OpenMP_UA3	136,0622	40	,18847	,02980
	OPENNEBULA_OpenMP_UA3	140,7925	40	,25856	,04088
Pair 4	NATIVO_OpenMP_UA4	115,7277	40	,15874	,02510
	OPENNEBULA_OpenMP_UA4	120,6960	40	,28548	,04514

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_OpenMP_UA1 - OPENNEBULA_OpenMP_UA1	-7,47125	,54828	,08669	-7,64660	-7,29590	-86,182	39	,000
Pair 2	NATIVO_OpenMP_UA2 - OPENNEBULA_OpenMP_UA2	-4,68375	,26580	,04203	-4,76876	-4,59874	-111,447	39	,000
Pair 3	NATIVO_OpenMP_UA3 - OPENNEBULA_OpenMP_UA3	-4,73025	,31625	,05000	-4,83139	-4,62911	-94,599	39	,000
Pair 4	NATIVO_OpenMP_UA4 - OPENNEBULA_OpenMP_UA4	-4,96825	,30803	,04870	-5,06676	-4,86974	-102,010	39	,000

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	OPENSTACK_OpenMP_UA1	226,1065	40	2,45310	,38787
	OPENNEBULA_OpenMP_UA1	220,8813	40	,49833	,07879
Pair 2	OPENSTACK_OpenMP_UA2	138,0143	40	,91896	,14530
	OPENNEBULA_OpenMP_UA2	133,0998	40	,23000	,03637
Pair 3	OPENSTACK_OpenMP_UA3	144,2885	40	,45273	,07158
	OPENNEBULA_OpenMP_UA3	140,7925	40	,25856	,04088
Pair 4	OPENSTACK_OpenMP_UA4	126,8815	40	,71785	,11350
	OPENNEBULA_OpenMP_UA4	120,6960	40	,28548	,04514

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	OPENNEBULA_OpenMP_UA1 - OPENSTACK_OpenMP_UA1	-5,22525	2,30393	,36428	-5,96208	-4,48842	-14,344	39	,000
Pair 2	OPENNEBULA_OpenMP_UA2 - OPENSTACK_OpenMP_UA2	-4,91450	,98092	,15510	-5,22821	-4,60079	-31,687	39	,000
Pair 3	OPENNEBULA_OpenMP_UA3 - OPENSTACK_OpenMP_UA3	-3,49600	,63420	,10028	-3,69883	-3,29317	-34,864	39	,000
Pair 4	OPENNEBULA_OpenMP_UA4 - OPENSTACK_OpenMP_UA4	-6,18550	,78986	,12489	-6,43811	-5,93289	-49,529	39	,000

I.4 Testes estatístico para programa NPB-OMP [CG]

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_OpenMP_CG1	88,6183	40	,33086	,05231
	OPENSTACK_OpenMP_CG1	89,1975	40	1,23959	,19600
Pair 2	NATIVO_OpenMP_CG2	46,3308	40	1,10726	,17507
	OPENSTACK_OpenMP_CG2	51,2950	40	,26610	,04207
Pair 3	NATIVO_OpenMP_CG3	57,6955	40	,18460	,02919
	OPENSTACK_OpenMP_CG3	61,9555	40	,31911	,05046
Pair 4	NATIVO_OpenMP_CG4	45,8793	40	,07381	,01167
	OPENSTACK_OpenMP_CG4	52,0593	40	,29535	,04670

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_OpenMP_CG1 - OPENSTACK_OpenMP_CG1	-,57925	1,24585	,19699	-,97769	-,18081	-2,941	39	,005
Pair 2	NATIVO_OpenMP_CG2 - OPENSTACK_OpenMP_CG2	-4,96425	1,08108	,17093	-5,31000	-4,61850	-29,042	39	,000
Pair 3	NATIVO_OpenMP_CG3 - OPENSTACK_OpenMP_CG3	-4,26000	,33403	,05281	-4,36683	-4,15317	-80,660	39	,000
Pair 4	NATIVO_OpenMP_CG4 - OPENSTACK_OpenMP_CG4	-6,18000	,32873	,05198	-6,28513	-6,07487	-118,897	39	,000

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_OpenMP_CG1	88,6183	40	,33086	,05231
	OPENNEBULA_OpenMP_CG1	87,7765	40	1,61629	,25556
Pair 2	NATIVO_OpenMP_CG2	46,3308	40	1,10726	,17507
	OPENNEBULA_OpenMP_CG2	49,1137	40	,17951	,02838
Pair 3	NATIVO_OpenMP_CG3	57,6955	40	,18460	,02919
	OPENNEBULA_OpenMP_CG3	60,2768	40	,27443	,04339
Pair 4	NATIVO_OpenMP_CG4	45,8793	40	,07381	,01167
	OPENNEBULA_OpenMP_CG4	49,6837	40	,15788	,02496

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_OpenMP_CG1 - OPENNEBULA_OpenMP_CG1	,84175	1,70738	,26996	,29570	1,38780	3,118	39	,003
Pair 2	NATIVO_OpenMP_CG2 - OPENNEBULA_OpenMP_CG2	-2,78300	1,12394	,17771	-3,14245	-2,42355	-15,660	39	,000
Pair 3	NATIVO_OpenMP_CG3 - OPENNEBULA_OpenMP_CG3	-2,58125	,37737	,05967	-2,70194	-2,46056	-43,260	39	,000
Pair 4	NATIVO_OpenMP_CG4 - OPENNEBULA_OpenMP_CG4	-3,80450	,16126	,02550	-3,85607	-3,75293	-149,211	39	,000

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	OPENSTACK_OpenMP_CG1	89,1975	40	1,23959	,19600
	OPENNEBULA_OpenMP_CG1	87,7765	40	1,61629	,25556
Pair 2	OPENSTACK_OpenMP_CG2	51,2950	40	,26610	,04207
	OPENNEBULA_OpenMP_CG2	49,1137	40	,17951	,02838
Pair 3	OPENSTACK_OpenMP_CG3	61,9555	40	,31911	,05046
	OPENNEBULA_OpenMP_CG3	60,2768	40	,27443	,04339
Pair 4	OPENSTACK_OpenMP_CG4	52,0593	40	,29535	,04670
	OPENNEBULA_OpenMP_CG4	49,6837	40	,15788	,02496

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	OPENNEBULA_OpenMP_CG1 - OPENSTACK_OpenMP_CG1	-1,42100	2,15267	,34037	-2,10946	-,73254	-4,175	39	,000
Pair 2	OPENNEBULA_OpenMP_CG2 - OPENSTACK_OpenMP_CG2	-2,18125	,29813	,04714	-2,27660	-2,08590	-46,273	39	,000
Pair 3	OPENNEBULA_OpenMP_CG3 - OPENSTACK_OpenMP_CG3	-1,67875	,53547	,08467	-1,85000	-1,50750	-19,828	39	,000
Pair 4	OPENNEBULA_OpenMP_CG4 - OPENSTACK_OpenMP_CG4	-2,37550	,35579	,05625	-2,48929	-2,26171	-42,227	39	,000

I.5 Teste estatístico para programa NPB-OMP [EP]

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_OpenMP_EP1	74,7332	40	,34891	,05517
	OPENSTACK_OpenMP_EP1	76,6432	40	,66297	,10482
Pair 2	NATIVO_OpenMP_EP2	38,2135	40	,12093	,01912
	OPENSTACK_OpenMP_EP2	38,8785	40	,17123	,02707
Pair 3	NATIVO_OpenMP_EP3	29,0598	40	,16482	,02606
	OPENSTACK_OpenMP_EP3	29,1972	40	,29896	,04727
Pair 4	NATIVO_OpenMP_EP4	22,3082	40	,10461	,01654
	OPENSTACK_OpenMP_EP4	23,4453	40	,21004	,03321

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_OpenMP_EP1 - OPENSTACK_OpenMP_EP1	-1,91000	,69501	,10989	-2,13228	-1,68772	-17,381	39	,000
Pair 2	NATIVO_OpenMP_EP2 - OPENSTACK_OpenMP_EP2	-,66500	,22595	,03573	-,73726	-,59274	-18,614	39	,000
Pair 3	NATIVO_OpenMP_EP3 - OPENSTACK_OpenMP_EP3	-,13750	,30822	,04873	-,23607	-,03893	-2,821	39	,007
Pair 4	NATIVO_OpenMP_EP4 - OPENSTACK_OpenMP_EP4	-1,13700	,20428	,03230	-1,20233	-1,07167	-35,202	39	,000

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_OpenMP_EP1	74,7332	40	,34891	,05517
	OPENNEBULA_OpenMP_EP1	76,0195	40	,42234	,06678
Pair 2	NATIVO_OpenMP_EP2	38,2135	40	,12093	,01912
	OPENNEBULA_OpenMP_EP2	38,5495	40	,15576	,02463
Pair 3	NATIVO_OpenMP_EP3	29,0598	40	,16482	,02606
	OPENNEBULA_OpenMP_EP3	29,2502	40	,28246	,04466
Pair 4	NATIVO_OpenMP_EP4	22,3082	40	,10461	,01654
	OPENNEBULA_OpenMP_EP4	22,7790	40	,18429	,02914

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_OpenMP_EP1 - OPENNEBULA_OpenMP_EP1	-1,28625	,49534	,07832	-1,44467	-1,12783	-16,423	39	,000
Pair 2	NATIVO_OpenMP_EP2 - OPENNEBULA_OpenMP_EP2	-,33600	,20765	,03283	-,40241	-,26959	-10,234	39	,000
Pair 3	NATIVO_OpenMP_EP3 - OPENNEBULA_OpenMP_EP3	-,19050	,30366	,04801	-,28762	-,09338	-3,968	39	,000
Pair 4	NATIVO_OpenMP_EP4 - OPENNEBULA_OpenMP_EP4	-,47075	,20663	,03267	-,53683	-,40467	-14,409	39	,000

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	OPENSTACK_OpenMP_EP1	76,6432	40	,66297	,10482
	OPENNEBULA_OpenMP_EP1	76,0195	40	,42234	,06678
Pair 2	OPENSTACK_OpenMP_EP2	38,8785	40	,17123	,02707
	OPENNEBULA_OpenMP_EP2	38,5495	40	,15576	,02463
Pair 3	OPENSTACK_OpenMP_EP3	29,1972	40	,29896	,04727
	OPENNEBULA_OpenMP_EP3	29,2502	40	,28246	,04466
Pair 4	OPENSTACK_OpenMP_EP4	23,4453	40	,21004	,03321
	OPENNEBULA_OpenMP_EP4	22,7790	40	,18429	,02914

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	OPENSTACK_OpenMP_EP1 - OPENNEBULA_OpenMP_EP1	,62375	,76415	,12082	,37936	,86814	5,163	39	,000
Pair 2	OPENSTACK_OpenMP_EP2 - OPENNEBULA_OpenMP_EP2	,32900	,21941	,03469	,25883	,39917	9,483	39	,000
Pair 3	OPENSTACK_OpenMP_EP3 - OPENNEBULA_OpenMP_EP3	-,05300	,35865	,05671	-,16770	,06170	-,935	39	,356
Pair 4	OPENSTACK_OpenMP_EP4 - OPENNEBULA_OpenMP_EP4	,66625	,28760	,04547	,57427	,75823	14,651	39	,000

I.6 Teste estatístico para programa NPB-OMP [LU]

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_OpenMP_LU1	245,1220	40	,87449	,13827
	OPENSTACK_OpenMP_LU1	267,4428	40	3,18849	,50415
Pair 2	NATIVO_OpenMP_LU2	163,1918	40	,47346	,07486
	OPENSTACK_OpenMP_LU2	184,0473	40	1,75899	,27812
Pair 3	NATIVO_OpenMP_LU3	174,9973	40	,43551	,06886
	OPENSTACK_OpenMP_LU3	199,4173	40	1,50309	,23766
Pair 4	NATIVO_OpenMP_LU4	160,7652	40	,33114	,05236
	OPENSTACK_OpenMP_LU4	188,8447	40	1,08730	,17192

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_OpenMP_LU1 - OPENSTACK_OpenMP_LU1	-22,32075	3,32100	,52510	-23,38286	-21,25864	-42,508	39	,000
Pair 2	NATIVO_OpenMP_LU2 - OPENSTACK_OpenMP_LU2	-20,85550	1,83620	,29033	-21,44274	-20,26826	-71,834	39	,000
Pair 3	NATIVO_OpenMP_LU3 - OPENSTACK_OpenMP_LU3	-24,42000	1,63994	,25930	-24,94448	-23,89552	-94,178	39	,000
Pair 4	NATIVO_OpenMP_LU4 - OPENSTACK_OpenMP_LU4	-28,07950	1,05389	,16663	-28,41655	-27,74245	-168,510	39	,000

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_OpenMP_LU1	245,1220	40	,87449	,13827
	OPENNEBULA_OpenMP_LU1	260,8680	40	,92570	,14637
Pair 2	NATIVO_OpenMP_LU2	163,1918	40	,47346	,07486
	OPENNEBULA_OpenMP_LU2	177,6285	40	,53492	,08458
Pair 3	NATIVO_OpenMP_LU3	174,9973	40	,43551	,06886
	OPENNEBULA_OpenMP_LU3	190,4453	40	,58245	,09209
Pair 4	NATIVO_OpenMP_LU4	160,7652	40	,33114	,05236
	OPENNEBULA_OpenMP_LU4	179,5315	40	,49614	,07845

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_OpenMP_LU1 - OPENNEBULA_OpenMP_LU1	-15,74600	1,26566	,20012	-16,15078	-15,34122	-78,683	39	,000
Pair 2	NATIVO_OpenMP_LU2 - OPENNEBULA_OpenMP_LU2	-14,43675	,82797	,13091	-14,70155	-14,17195	-110,277	39	,000
Pair 3	NATIVO_OpenMP_LU3 - OPENNEBULA_OpenMP_LU3	-15,44800	,67573	,10684	-15,66411	-15,23189	-144,588	39	,000
Pair 4	NATIVO_OpenMP_LU4 - OPENNEBULA_OpenMP_LU4	-18,76625	,61473	,09720	-18,96285	-18,56965	-193,073	39	,000

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	OPENSTACK_OpenMP_LU1	267,4428	40	3,18849	,50415
	OPENNEBULA_OpenMP_LU1	260,8680	40	,92570	,14637
Pair 2	OPENSTACK_OpenMP_LU2	184,0473	40	1,75899	,27812
	OPENNEBULA_OpenMP_LU2	177,6285	40	,53492	,08458
Pair 3	OPENSTACK_OpenMP_LU3	199,4173	40	1,50309	,23766
	OPENNEBULA_OpenMP_LU3	190,4453	40	,58245	,09209
Pair 4	OPENSTACK_OpenMP_LU4	188,8447	40	1,08730	,17192
	OPENNEBULA_OpenMP_LU4	179,5315	40	,49614	,07845

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	OPENSTACK_OpenMP_LU1 - OPENNEBULA_OpenMP_LU1	6,57475	2,66792	,42183	5,72151	7,42799	15,586	39	,000
Pair 2	OPENSTACK_OpenMP_LU2 - OPENNEBULA_OpenMP_LU2	6,41875	1,58819	,25111	5,91082	6,92668	25,561	39	,000
Pair 3	OPENSTACK_OpenMP_LU3 - OPENNEBULA_OpenMP_LU3	8,97200	2,02231	,31975	8,32523	9,61877	28,059	39	,000
Pair 4	OPENSTACK_OpenMP_LU4 - OPENNEBULA_OpenMP_LU4	9,31325	1,30479	,20630	8,89596	9,73054	45,143	39	,000

I.7 Teste estatístico para programa NPB-OMP [MG]

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_OpenMP_MG1	15,8318	40	,03686	,00583
	OPENSTACK_OpenMP_MG1	16,6775	40	,17134	,02709
Pair 2	NATIVO_OpenMP_MG2	8,4188	40	,00686	,00109
	OPENSTACK_OpenMP_MG2	14,8495	40	3,35472	,53043
Pair 3	NATIVO_OpenMP_MG3	8,8810	40	,06663	,01054
	OPENSTACK_OpenMP_MG3	10,0330	40	,05743	,00908
Pair 4	NATIVO_OpenMP_MG4	7,6400	40	,00877	,00139
	OPENSTACK_OpenMP_MG4	8,2847	40	,06660	,01053

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_OpenMP_MG1 - OPENSTACK_OpenMP_MG1	-,84575	,17936	,02836	-,90311	-,78839	-29,823	39	,000
Pair 2	NATIVO_OpenMP_MG2 - OPENSTACK_OpenMP_MG2	-6,43075	3,35493	,53046	-7,50371	-5,35779	-12,123	39	,000
Pair 3	NATIVO_OpenMP_MG3 - OPENSTACK_OpenMP_MG3	-1,15200	,08250	,01304	-1,17838	-1,12562	-88,314	39	,000
Pair 4	NATIVO_OpenMP_MG4 - OPENSTACK_OpenMP_MG4	-,64475	,06733	,01065	-,66628	-,62322	-60,564	39	,000

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_OpenMP_MG1	15,8318	40	,03686	,00583
	OPENNEBULA_OpenMP_MG1	16,4005	40	,11732	,01855
Pair 2	NATIVO_OpenMP_MG2	8,4188	40	,00686	,00109
	OPENNEBULA_OpenMP_MG2	8,7987	40	,05095	,00806
Pair 3	NATIVO_OpenMP_MG3	8,8810	40	,06663	,01054
	OPENNEBULA_OpenMP_MG3	9,2280	40	,11071	,01751
Pair 4	NATIVO_OpenMP_MG4	7,6400	40	,00877	,00139
	OPENNEBULA_OpenMP_MG4	7,9222	40	,05433	,00859

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_OpenMP_MG1 - OPENNEBULA_OpenMP_MG1	-,56875	,12296	,01944	-,60807	-,52943	-29,254	39	,000
Pair 2	NATIVO_OpenMP_MG2 - OPENNEBULA_OpenMP_MG2	-,38000	,05054	,00799	-,39616	-,36384	-47,557	39	,000
Pair 3	NATIVO_OpenMP_MG3 - OPENNEBULA_OpenMP_MG3	-,34700	,14110	,02231	-,39213	-,30187	-15,554	39	,000
Pair 4	NATIVO_OpenMP_MG4 - OPENNEBULA_OpenMP_MG4	-,28225	,05572	,00881	-,30007	-,26443	-32,035	39	,000

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	OPENSTACK_OpenMP_MG1	16,6775	40	,17134	,02709
	OPENNEBULA_OpenMP_MG1	16,4005	40	,11732	,01855
Pair 2	OPENSTACK_OpenMP_MG2	14,8495	40	3,35472	,53043
	OPENNEBULA_OpenMP_MG2	8,7987	40	,05095	,00806
Pair 3	OPENSTACK_OpenMP_MG3	10,0330	40	,05743	,00908
	OPENNEBULA_OpenMP_MG3	9,2280	40	,11071	,01751
Pair 4	OPENSTACK_OpenMP_MG4	8,2847	40	,06660	,01053
	OPENNEBULA_OpenMP_MG4	7,9222	40	,05433	,00859

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	OPENSTACK_OpenMP_MG1 - OPENNEBULA_OpenMP_MG1	,27700	,17203	,02720	,22198	,33202	10,183	39	,000
Pair 2	OPENSTACK_OpenMP_MG2 - OPENNEBULA_OpenMP_MG2	6,05075	3,35088	,52982	4,97909	7,12241	11,420	39	,000
Pair 3	OPENSTACK_OpenMP_MG3 - OPENNEBULA_OpenMP_MG3	,80500	,13847	,02189	,76071	,84929	36,768	39	,000
Pair 4	OPENSTACK_OpenMP_MG4 - OPENNEBULA_OpenMP_MG4	,36250	,06879	,01088	,34050	,38450	33,328	39	,000

I.8 Testes estatístico para programa NPB-OMP [IS]

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_OpenMP_IS1	4,2333	40	,01591	,00252
	OPENSTACK_OpenMP_IS1	4,9773	40	,06587	,01041
Pair 2	NATIVO_OpenMP_IS2	2,1685	40	,00362	,00057
	OPENSTACK_OpenMP_IS2	2,5895	40	,04362	,00690
Pair 3	NATIVO_OpenMP_IS3	2,0050	40	,04478	,00708
	OPENSTACK_OpenMP_IS3	2,9665	40	,12721	,02011
Pair 4	NATIVO_OpenMP_IS4	1,8798	40	,01291	,00204
	OPENSTACK_OpenMP_IS4	3,4560	40	,03185	,00504

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_OpenMP_IS1 - OPENSTACK_OpenMP_IS1	-,74400	,06598	,01043	-,76510	-,72290	-71,321	39	,000
Pair 2	NATIVO_OpenMP_IS2 - OPENSTACK_OpenMP_IS2	-,42100	,04355	,00689	-,43493	-,40707	-61,143	39	,000
Pair 3	NATIVO_OpenMP_IS3 - OPENSTACK_OpenMP_IS3	-,96150	,13366	,02113	-1,00425	-,91875	-45,497	39	,000
Pair 4	NATIVO_OpenMP_IS4 - OPENSTACK_OpenMP_IS4	-1,57625	,03176	,00502	-1,58641	-1,56609	-313,895	39	,000

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_OpenMP_IS1	4,2333	40	,01591	,00252
	OPENNEBULA_OpenMP_IS1	4,9095	40	,04977	,00787
Pair 2	NATIVO_OpenMP_IS2	2,1685	40	,00362	,00057
	OPENNEBULA_OpenMP_IS2	2,5333	40	,03108	,00491
Pair 3	NATIVO_OpenMP_IS3	2,0050	40	,04478	,00708
	OPENNEBULA_OpenMP_IS3	2,5812	40	,06969	,01102
Pair 4	NATIVO_OpenMP_IS4	1,8798	40	,01291	,00204
	OPENNEBULA_OpenMP_IS4	3,4467	40	,02223	,00352

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_OpenMP_IS1 - OPENNEBULA_OpenMP_IS1	-,67625	,05266	,00833	-,69309	-,65941	-81,223	39	,000
Pair 2	NATIVO_OpenMP_IS2 - OPENNEBULA_OpenMP_IS2	-,36475	,03154	,00499	-,37484	-,35466	-73,140	39	,000
Pair 3	NATIVO_OpenMP_IS3 - OPENNEBULA_OpenMP_IS3	-,57625	,07537	,01192	-,60035	-,55215	-48,356	39	,000
Pair 4	NATIVO_OpenMP_IS4 - OPENNEBULA_OpenMP_IS4	-1,56700	,02483	,00393	-1,57494	-1,55906	-399,176	39	,000

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	OPENSTACK_OpenMP_IS1	4,9773	40	,06587	,01041
	OPENNEBULA_OpenMP_IS1	4,9095	40	,04977	,00787
Pair 2	OPENSTACK_OpenMP_IS2	2,5895	40	,04362	,00690
	OPENNEBULA_OpenMP_IS2	2,5333	40	,03108	,00491
Pair 3	OPENSTACK_OpenMP_IS3	2,9665	40	,12721	,02011
	OPENNEBULA_OpenMP_IS3	2,5812	40	,06969	,01102
Pair 4	OPENSTACK_OpenMP_IS4	3,4560	40	,03185	,00504
	OPENNEBULA_OpenMP_IS4	3,4467	40	,02223	,00352

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	OPENSTACK_OpenMP_IS1 - OPENNEBULA_OpenMP_IS1	,06775	,07409	,01172	,04405	,09145	5,783	39	,000
Pair 2	OPENSTACK_OpenMP_IS2 - OPENNEBULA_OpenMP_IS2	,05625	,05077	,00803	,04001	,07249	7,007	39	,000
Pair 3	OPENSTACK_OpenMP_IS3 - OPENNEBULA_OpenMP_IS3	,38525	,14376	,02273	,33927	,43123	16,949	39	,000
Pair 4	OPENSTACK_OpenMP_IS4 - OPENNEBULA_OpenMP_IS4	,00925	,03407	,00539	-,00165	,02015	1,717	39	,094

APÊNDICE J. TESTE ESTATÍSCO DAS APLICAÇÕES PARALELAS – NPB-MPI

J.1 Teste estatístico para programa NPB-MPI [MG]

Amostras das Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_MPI_MG1	14,4067	40	,07298	,01332
	OPENSTACK_MPI_MG1	15,2770	40	,08009	,01462
Pair 2	NATIVO_MPI_MG2	15,2783	40	,02335	,00426
	OPENSTACK_MPI_MG2	21,2633	40	1,39448	,25460
Pair 3	NATIVO_MPI_MG4	15,4123	40	,14067	,02386
	OPENSTACK_MPI_MG4	20,9000	40	,52088	,09510
Pair 4	NATIVO_MPI_MG8	12,9267	40	,36509	,06666
	OPENSTACK_MPI_MG8	16,1193	40	,60001	,10955
Pair 5	NATIVO_MPI_MG16	11,6123	40	,31107	,05679
	OPENSTACK_MPI_MG16	17,7703	40	,72617	,13258

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_MPI_MG1 - OPENSTACK_MPI_MG1	-,97033	,10440	,01906	-1,00932	-,93135	-50,906	29	,000
Pair 2	NATIVO_MPI_MG2 - OPENSTACK_MPI_MG2	-5,98500	1,39352	,25442	-6,50535	-5,46465	-23,524	29	,000
Pair 3	NATIVO_MPI_MG4 - OPENSTACK_MPI_MG4	-5,48767	,51935	,09482	-5,68160	-5,29374	-57,874	29	,000
Pair 4	NATIVO_MPI_MG8 - OPENSTACK_MPI_MG8	-3,19267	,68990	,12596	-3,45028	-2,93505	-25,347	29	,000
Pair 5	NATIVO_MPI_MG16 - OPENSTACK_MPI_MG16	-6,15800	,79144	,14450	-6,45353	-5,86247	-42,617	29	,000

Amostras das Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_MPI_MG1	14,4067	40	,07298	,01332
	OPENNEBULA_MPI_MG1	15,0113	40	,06356	,01160
Pair 2	NATIVO_MPI_MG2	15,2783	40	,02335	,00426
	OPENNEBULA_MPI_MG2	17,2770	40	,33450	,06107
Pair 3	NATIVO_MPI_MG4	15,4123	40	,14067	,02386
	OPENNEBULA_MPI_MG4	17,1927	40	,22234	,04059
Pair 4	NATIVO_MPI_MG8	12,9267	40	,36509	,06666
	OPENNEBULA_MPI_MG8	15,3750	40	,43954	,08025
Pair 5	NATIVO_MPI_MG16	11,6123	40	,31107	,05679
	OPENNEBULA_MPI_MG16	17,3910	40	,65742	,12003

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_MPI_MG1 - OPENNEBULA_MPI_MG1	-,70467	,09024	,01648	-,73836	-,67097	-42,771	29	,000
Pair 2	NATIVO_MPI_MG2 - OPENNEBULA_MPI_MG2	-1,99867	,33768	,06165	-2,12476	-1,87258	-32,419	29	,000
Pair 3	NATIVO_MPI_MG4 - OPENNEBULA_MPI_MG4	-1,78033	,40240	,05521	-1,89325	-1,66742	-32,247	29	,000
Pair 4	NATIVO_MPI_MG8 - OPENNEBULA_MPI_MG8	-2,44833	,52103	,09513	-2,64289	-2,25378	-25,737	29	,000
Pair 5	NATIVO_MPI_MG16 - OPENNEBULA_MPI_MG16	-5,77867	,80276	,14656	-6,07842	-5,47891	-39,428	29	,000

Amostras das Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	OPENSTACK_MPI_MG1	15,2770	40	,08009	,01462
	OPENNEBULA_MPI_MG1	15,0113	40	,06356	,01160
Pair 2	OPENSTACK_MPI_MG2	21,2633	40	1,39448	,25460
	OPENNEBULA_MPI_MG2	17,2770	40	,33450	,06107
Pair 3	OPENSTACK_MPI_MG4	20,9000	40	,52088	,09510
	OPENNEBULA_MPI_MG4	17,1927	40	,22234	,04059
Pair 4	OPENSTACK_MPI_MG8	16,1193	40	,60001	,10955
	OPENNEBULA_MPI_MG8	15,3750	40	,43954	,08025
Pair 5	OPENSTACK_MPI_MG16	17,7703	40	,72617	,13258
	OPENNEBULA_MPI_MG16	17,3910	40	,65742	,12003

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	OPENSTACK_MPI_MG1 - OPENNEBULA_MPI_MG1	,26567	,10408	,01900	,22680	,40453	13,981	29	,000
Pair 2	OPENSTACK_MPI_MG2 - OPENNEBULA_MPI_MG2	3,98633	1,41209	,25781	3,45905	4,51362	15,462	29	,000
Pair 3	OPENSTACK_MPI_MG4 - OPENNEBULA_MPI_MG4	3,70733	,60052	,10964	3,48310	3,93157	33,814	29	,000
Pair 4	OPENSTACK_MPI_MG8 - OPENNEBULA_MPI_MG8	,74433	,79225	,14465	,44850	1,04017	5,146	29	,000
Pair 5	OPENSTACK_MPI_MG16 - OPENNEBULA_MPI_MG16	,37933	,95868	,17503	,02135	,73731	2,167	29	,039

J.2 Testes estatístico para programa NPB-MPI [LU]

Amostras das Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_MPI_LU1	250,4513	40	,59421	,10849
	OPENSTACK_MPI_LU1	274,1990	40	,90542	,16531
Pair 2	NATIVO_MPI_LU2	166,8327	40	,26362	,04813
	OPENSTACK_MPI_LU2	185,4610	40	3,69980	,67549
Pair 3	NATIVO_MPI_LU4	102,6553	40	,11985	,02188
	OPENSTACK_MPI_LU4	122,8310	40	3,23996	,59153
Pair 4	NATIVO_MPI_LU8	81,1087	40	1,27472	,23273
	OPENSTACK_MPI_LU8	106,9400	40	1,18083	,21559
Pair 5	NATIVO_MPI_LU16	81,5260	40	,94175	,17194
	OPENSTACK_MPI_LU16	126,9663	40	1,86059	,33970

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_MPI_LU1 - OPENSTACK_MPI_LU1	-23,74767	1,09786	,20044	-24,15762	-23,33772	-118,477	29	,000
Pair 2	NATIVO_MPI_LU2 - OPENSTACK_MPI_LU2	-18,62833	3,77186	,68864	-20,03677	-17,21990	-27,051	29	,000
Pair 3	NATIVO_MPI_LU4 - OPENSTACK_MPI_LU4	-20,17567	3,26693	,59646	-21,39556	-18,95578	-33,826	29	,000
Pair 4	NATIVO_MPI_LU8 - OPENSTACK_MPI_LU8	-25,82133	1,56929	,28651	-26,40732	-25,23535	-90,123	29	,000
Pair 5	NATIVO_MPI_LU16 - OPENSTACK_MPI_LU16	-45,44033	2,03925	,37231	-46,20180	-44,67887	-122,048	29	,000

Amostras das Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_MPI_LU1	250,4513	40	,59421	,10849
	OPENNEBULA_MPI_LU1	267,5250	40	,94234	,17205
Pair 2	NATIVO_MPI_LU2	166,8327	40	,26362	,04813
	OPENNEBULA_MPI_LU2	182,0577	40	,32270	,05892
Pair 3	NATIVO_MPI_LU4	102,6553	40	,11985	,02188
	OPENNEBULA_MPI_LU4	115,0620	40	,16682	,04046
Pair 4	NATIVO_MPI_LU8	81,1087	40	1,27472	,23273
	OPENNEBULA_MPI_LU8	115,5320	40	4,14612	,75698
Pair 5	NATIVO_MPI_LU16	81,5260	40	,94175	,17194
	OPENNEBULA_MPI_LU16	133,2553	40	8,40427	1,51614

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_MPI_LU1 - OPENNEBULA_MPI_LU1	-17,07367	1,04242	,19032	-17,46291	-16,68442	-89,711	29	,000
Pair 2	NATIVO_MPI_LU2 - OPENNEBULA_MPI_LU2	-15,22500	,36431	,06651	-15,36104	-15,08896	-228,900	29	,000
Pair 3	NATIVO_MPI_LU4 - OPENNEBULA_MPI_LU4	-12,40667	,20348	,03715	-12,48265	-12,34069	-333,967	29	,000
Pair 4	NATIVO_MPI_LU8 - OPENNEBULA_MPI_LU8	-34,42333	4,22605	,77157	-36,00137	-32,84540	-44,615	29	,000
Pair 5	NATIVO_MPI_LU16 - OPENNEBULA_MPI_LU16	-51,72933	8,06250	1,47200	-54,73992	-48,71875	-35,142	29	,000

Amostras das Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	OPENSTACK_MPI_LU1	274,1990	40	,90542	,16531
	OPENNEBULA_MPI_LU1	267,5250	40	,94234	,17205
Pair 2	OPENSTACK_MPI_LU2	185,4610	40	3,69980	,67549
	OPENNEBULA_MPI_LU2	182,0577	40	,32270	,05892
Pair 3	OPENSTACK_MPI_LU4	122,8310	40	3,23996	,59153
	OPENNEBULA_MPI_LU4	115,0620	40	,16682	,04046
Pair 4	OPENSTACK_MPI_LU8	106,9400	40	1,18083	,21559
	OPENNEBULA_MPI_LU8	115,5320	40	4,14612	,75698
Pair 5	OPENSTACK_MPI_LU16	126,9663	40	1,86059	,33970
	OPENNEBULA_MPI_LU16	133,2553	40	8,40427	1,51614

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	OPENSTACK_MPI_LU1 - OPENNEBULA_MPI_LU1	6,67400	1,34478	,24552	6,17185	7,17615	27,183	29	,000
Pair 2	OPENSTACK_MPI_LU2 - OPENNEBULA_MPI_LU2	3,40333	3,64554	,66558	2,04207	4,76460	5,113	29	,000
Pair 3	OPENSTACK_MPI_LU4 - OPENNEBULA_MPI_LU4	7,76900	3,16678	,57817	6,58650	8,95150	13,437	29	,000
Pair 4	OPENSTACK_MPI_LU8 - OPENNEBULA_MPI_LU8	-8,60200	4,21144	,76890	-10,17458	-7,02942	-11,187	29	,000
Pair 5	OPENSTACK_MPI_LU16 - OPENNEBULA_MPI_LU16	-6,28900	8,01328	1,46402	-9,28121	-3,29679	-4,299	29	,000

J.3 Testes estatístico para programa NPB-MPI [IS]

Amostras das Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_MPI_IS1	3,5527	40	,02778	,00507
	OPENSTACK_MPI_IS1	4,1047	40	,05409	,00969
Pair 2	NATIVO_MPI_IS2	31,2883	40	,04900	,00895
	OPENSTACK_MPI_IS2	43,8997	40	3,33294	,60851
Pair 3	NATIVO_MPI_IS4	49,3210	40	,85621	,15632
	OPENSTACK_MPI_IS4	52,9400	40	,97809	,17857
Pair 4	NATIVO_MPI_IS8	44,3550	40	1,31385	,23988
	OPENSTACK_MPI_IS8	49,5660	40	1,28717	,23500
Pair 5	NATIVO_MPI_IS16	37,3507	40	1,04961	,19163
	OPENSTACK_MPI_IS16	48,7270	40	2,20376	,40235

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_MPI_IS1 - OPENSTACK_MPI_IS1	-,55200	,06405	,01151	-,57554	-,52846	-47,954	29	,000
Pair 2	NATIVO_MPI_IS2 - OPENSTACK_MPI_IS2	-12,61133	3,33923	,60966	-13,85822	-11,36445	-20,686	29	,000
Pair 3	NATIVO_MPI_IS4 - OPENSTACK_MPI_IS4	-3,60900	1,52333	,27812	-4,17782	-3,04018	-12,976	29	,000
Pair 4	NATIVO_MPI_IS8 - OPENSTACK_MPI_IS8	-5,21100	1,70181	,31071	-5,84647	-4,57553	-16,771	29	,000
Pair 5	NATIVO_MPI_IS16 - OPENSTACK_MPI_IS16	-11,37633	2,61825	,47802	-12,35400	-10,39866	-23,799	29	,000

Amostras das Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_MPI_IS1	3,5527	40	,02778	,00507
	OPENNEBULA_MPI_IS1	4,0213	40	,02609	,00476
Pair 2	NATIVO_MPI_IS2	31,2883	40	,04900	,00895
	OPENNEBULA_MPI_IS2	35,2033	40	,40485	,07392
Pair 3	NATIVO_MPI_IS4	49,3210	40	,85621	,15632
	OPENNEBULA_MPI_IS4	48,2353	40	,77288	,14111
Pair 4	NATIVO_MPI_IS8	44,3550	40	1,31385	,23988
	OPENNEBULA_MPI_IS8	42,6600	40	1,04670	,19110
Pair 5	NATIVO_MPI_IS16	37,3507	40	1,04961	,19163
	OPENNEBULA_MPI_IS16	52,0123	40	4,03446	,73659

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_MPI_IS1 - OPENNEBULA_MPI_IS1	-,46867	,03431	,00626	-,48148	-,45585	-74,808	29	,000
Pair 2	NATIVO_MPI_IS2 - OPENNEBULA_MPI_IS2	-3,91500	,40314	,07360	-4,06554	-3,76446	-53,191	29	,000
Pair 3	NATIVO_MPI_IS4 - OPENNEBULA_MPI_IS4	1,08567	1,12223	,20489	,66662	1,50471	5,299	29	,000
Pair 4	NATIVO_MPI_IS8 - OPENNEBULA_MPI_IS8	1,69500	1,62927	,29746	1,08662	2,40338	5,698	29	,000
Pair 5	NATIVO_MPI_IS16 - OPENNEBULA_MPI_IS16	-14,66167	4,28715	,78272	-16,26252	-13,06082	-18,732	29	,000

Amostras das Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	OPENSTACK_MPI_IS1	4,1047	40	,05409	,00969
	OPENNEBULA_MPI_IS1	4,0213	40	,02609	,00476
Pair 2	OPENSTACK_MPI_IS2	43,8997	40	3,33294	,60851
	OPENNEBULA_MPI_IS2	35,2033	40	,40485	,07392
Pair 3	OPENSTACK_MPI_IS4	52,9400	40	,97809	,17857
	OPENNEBULA_MPI_IS4	48,2353	40	,77288	,14111
Pair 4	OPENSTACK_MPI_IS8	49,5660	40	1,28717	,23500
	OPENNEBULA_MPI_IS8	42,6600	40	1,04670	,19110
Pair 5	OPENSTACK_MPI_IS16	48,7270	40	2,20376	,40235
	OPENNEBULA_MPI_IS16	52,0123	40	4,03446	,73659

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	OPENSTACK_MPI_IS1 - OPENNEBULA_MPI_IS1	,08333	,05744	,01049	,06189	,10478	7,947	29	,000
Pair 2	OPENSTACK_MPI_IS2 - OPENNEBULA_MPI_IS2	8,69633	3,47570	,63457	7,39849	9,99418	13,704	29	,000
Pair 3	OPENSTACK_MPI_IS4 - OPENNEBULA_MPI_IS4	4,69467	1,38225	,25236	4,17853	5,21081	18,603	29	,000
Pair 4	OPENSTACK_MPI_IS8 - OPENNEBULA_MPI_IS8	6,90600	1,78204	,32535	6,24058	7,57142	21,226	29	,000
Pair 5	OPENSTACK_MPI_IS16 - OPENNEBULA_MPI_IS16	-3,28533	4,52938	,82695	-4,97663	-1,59404	-3,973	29	,000

J.4 Teste estatístico para programa NPB-MPI [FT]

Amostras das Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_MPI_FT1	81,5360	40	,52731	,09627
	OPENSTACK_MPI_FT1	87,2490	40	16,19216	2,95627
Pair 2	NATIVO_MPI_FT2	291,6937	40	,09718	,01774
	OPENSTACK_MPI_FT2	399,0080	40	17,85495	3,25985
Pair 3	NATIVO_MPI_FT4	265,8243	40	1,22357	,22339
	OPENSTACK_MPI_FT4	364,9253	40	8,64006	1,57563
Pair 4	NATIVO_MPI_FT8	199,7240	40	1,03548	,18905
	OPENSTACK_MPI_FT8	262,2843	40	2,80025	,51125
Pair 5	NATIVO_MPI_FT16	218,9593	40	4,15321	,75827
	OPENSTACK_MPI_FT16	324,1687	40	6,97920	1,27422

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_MPI_FT1 - OPENSTACK_MPI_FT1	-5,71400	16,11961	2,94402	-11,73216	,40616	-1,941	29	,062
Pair 2	NATIVO_MPI_FT2 - OPENSTACK_MPI_FT2	-107,31433	17,85409	3,25970	-113,98116	-100,64751	-32,922	29	,000
Pair 3	NATIVO_MPI_FT4 - OPENSTACK_MPI_FT4	-99,10100	8,79399	1,60556	-102,38473	-95,81727	-61,724	29	,000
Pair 4	NATIVO_MPI_FT8 - OPENSTACK_MPI_FT8	-62,56133	2,85865	,52192	-63,62877	-61,49390	-119,869	29	,000
Pair 5	NATIVO_MPI_FT16 - OPENSTACK_MPI_FT16	-105,20933	8,41233	1,53587	-108,35055	-102,06812	-68,501	29	,000

Amostras das Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_MPI_FT1	81,5360	40	,52731	,09627
	OPENNEBULA_MPI_FT1	82,9570	40	5,46936	,99856
Pair 2	NATIVO_MPI_FT2	291,6937	40	,09718	,01774
	OPENNEBULA_MPI_FT2	320,5083	40	1,14850	,20969
Pair 3	NATIVO_MPI_FT4	265,8243	40	1,22357	,22339
	OPENNEBULA_MPI_FT4	290,1993	40	,98235	,17935
Pair 4	NATIVO_MPI_FT8	199,7240	40	1,03548	,18905
	OPENNEBULA_MPI_FT8	256,1903	40	1,45890	,26636
Pair 5	NATIVO_MPI_FT16	218,9593	40	4,15321	,75827
	OPENNEBULA_MPI_FT16	273,9713	40	2,22464	,40616

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_MPI_FT1 - OPENNEBULA_MPI_FT1	-1,42100	5,36104	,97879	-3,42284	,58084	-1,452	29	,157
Pair 2	NATIVO_MPI_FT2 - OPENNEBULA_MPI_FT2	-28,81467	1,13351	,20695	-29,23793	-28,39141	-139,235	29	,000
Pair 3	NATIVO_MPI_FT4 - OPENNEBULA_MPI_FT4	-24,37500	1,53514	,28028	-24,94823	-23,80177	-86,967	29	,000
Pair 4	NATIVO_MPI_FT8 - OPENNEBULA_MPI_FT8	-56,46733	1,83527	,33507	-57,15263	-55,78203	-168,523	29	,000
Pair 5	NATIVO_MPI_FT16 - OPENNEBULA_MPI_FT16	-55,01200	4,49165	,82006	-56,68921	-53,33479	-67,083	29	,000

Amostras das Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	OPENSTACK_MPI_FT1	87,2490	40	16,19216	2,95627
	OPENNEBULA_MPI_FT1	82,9570	40	5,46936	,99856
Pair 2	OPENSTACK_MPI_FT2	399,0080	40	17,85495	3,25985
	OPENNEBULA_MPI_FT2	320,5083	40	1,14850	,20969
Pair 3	OPENSTACK_MPI_FT4	364,9253	40	8,64006	1,57563
	OPENNEBULA_MPI_FT4	290,1993	40	,98235	,17935
Pair 4	OPENSTACK_MPI_FT8	262,2843	40	2,80025	,51125
	OPENNEBULA_MPI_FT8	256,1903	40	1,45890	,26636
Pair 5	OPENSTACK_MPI_FT16	324,1687	40	6,97920	1,27422
	OPENNEBULA_MPI_FT16	273,9713	40	2,22464	,40616

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	OPENSTACK_MPI_FT1 - OPENNEBULA_MPI_FT1	4,29200	14,42759	2,63411	-1,09535	9,67935	1,629	29	,114
Pair 2	OPENSTACK_MPI_FT2 - OPENNEBULA_MPI_FT2	78,49967	18,01469	3,28902	71,77287	85,22646	23,867	29	,000
Pair 3	OPENSTACK_MPI_FT4 - OPENNEBULA_MPI_FT4	74,72600	8,49420	1,55082	71,55422	77,89778	48,185	29	,000
Pair 4	OPENSTACK_MPI_FT8 - OPENNEBULA_MPI_FT8	6,09400	3,40692	,62202	4,82183	7,36617	9,797	29	,000
Pair 5	OPENSTACK_MPI_FT16 - OPENNEBULA_MPI_FT16	50,19733	6,83832	1,24850	47,64386	52,75080	40,206	29	,000

J.4 Teste estatístico para programa NPB-MPI [SP]

Amostras das Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_MPI_SP1	342,7463	40	1,07903	,19700
	OPENSTACK_MPI_SP1	386,4997	40	2,39827	,43786
Pair 2	NATIVO_MPI_SP4	269,6100	40	1,86707	,34088
	OPENSTACK_MPI_SP4	321,0403	40	3,09589	,56523
Pair 3	NATIVO_MPI_SP9	418,8663	40	1,27916	,23354
	OPENSTACK_MPI_SP9	556,5060	40	5,54479	1,01234
Pair 4	NATIVO_MPI_SP16	351,5883	40	1,14054	,20823
	OPENSTACK_MPI_SP16	468,1957	40	3,02252	,55183

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_MPI_SP1 - OPENSTACK_MPI_SP1	-43,75333	1,90809	,34837	-44,46582	-43,04084	-125,595	29	,000
Pair 2	NATIVO_MPI_SP4 - OPENSTACK_MPI_SP4	-51,42033	3,71799	,67881	-52,80865	-50,03201	-75,751	29	,000
Pair 3	NATIVO_MPI_SP9 - OPENSTACK_MPI_SP9	-137,63967	5,60314	1,02299	-139,73191	-135,54742	-134,547	29	,000
Pair 4	NATIVO_MPI_SP16 - OPENSTACK_MPI_SP16	-116,60733	3,31194	,60467	-117,84403	-115,37063	-192,843	29	,000

Amostras das Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_MPI_SP1	342,7463	40	1,07903	,19700
	OPENNEBULA_MPI_SP1	377,1123	40	1,28732	,23503
Pair 2	NATIVO_MPI_SP4	269,6100	40	1,86707	,34088
	OPENNEBULA_MPI_SP4	272,0803	40	1,33598	,24392
Pair 3	NATIVO_MPI_SP9	418,8663	40	1,27916	,23354
	OPENNEBULA_MPI_SP9	457,7033	40	2,02321	,36939
Pair 4	NATIVO_MPI_SP16	351,5883	40	1,14054	,20823
	OPENNEBULA_MPI_SP16	409,0513	40	5,75136	1,05005

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_MPI_SP1 - OPENNEBULA_MPI_SP1	-34,36600	1,33967	,24459	-34,86624	-33,86576	-140,505	29	,000
Pair 2	NATIVO_MPI_SP4 - OPENNEBULA_MPI_SP4	-2,47033	2,20284	,40218	-3,29289	-1,64778	-6,142	29	,000
Pair 3	NATIVO_MPI_SP9 - OPENNEBULA_MPI_SP9	-38,83700	2,01596	,36806	-39,58977	-38,08423	-105,518	29	,000
Pair 4	NATIVO_MPI_SP16 - OPENNEBULA_MPI_SP16	-57,46400	5,82701	1,06386	-59,63884	-55,28716	-54,014	29	,000

Amostras das Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	OPENSTACK_MPI_SP1	386,4997	40	2,39827	,43786
	OPENNEBULA_MPI_SP1	377,1123	40	1,28732	,23503
Pair 2	OPENSTACK_MPI_SP4	321,0403	40	3,09589	,56523
	OPENNEBULA_MPI_SP4	272,0803	40	1,33598	,24392
Pair 3	OPENSTACK_MPI_SP9	556,5060	40	5,54479	1,01234
	OPENNEBULA_MPI_SP9	457,7033	40	2,02321	,36939
Pair 4	OPENSTACK_MPI_SP16	468,1957	40	3,02252	,55183
	OPENNEBULA_MPI_SP16	409,0513	40	5,75136	1,05005

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	OPENSTACK_MPI_SP1 - OPENNEBULA_MPI_SP1	9,38733	2,25077	,41093	8,54688	10,22779	22,844	29	,000
Pair 2	OPENSTACK_MPI_SP4 - OPENNEBULA_MPI_SP4	48,95000	3,58449	,65444	47,61153	50,28847	74,797	29	,000
Pair 3	OPENSTACK_MPI_SP9 - OPENNEBULA_MPI_SP9	98,80267	5,66951	1,03511	96,68564	100,91970	95,452	29	,000
Pair 4	OPENSTACK_MPI_SP16 - OPENNEBULA_MPI_SP16	59,14433	6,76628	1,23535	56,61776	61,67090	47,877	29	,000

J.5 Teste estatístico para programa NPB-MPI [CG]

Amostras das Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_MPI_CG1	89,7567	40	,45206	,08254
	OPENSTACK_MPI_CG1	103,1523	40	,78913	,14408
Pair 2	NATIVO_MPI_CG2	96,2983	40	,08099	,01479
	OPENSTACK_MPI_CG2	140,2513	40	4,11749	,75175
Pair 3	NATIVO_MPI_CG4	178,3323	40	,18700	,03414
	OPENSTACK_MPI_CG4	210,2513	40	2,59848	,47441
Pair 4	NATIVO_MPI_CG8	202,9190	40	3,17605	,57986
	OPENSTACK_MPI_CG8	242,2663	40	4,95134	,90399
Pair 5	NATIVO_MPI_CG16	501,0357	40	1,44238	,26334
	OPENSTACK_MPI_CG16	673,8120	40	1,72959	,31578

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_MPI_CG1 - OPENSTACK_MPI_CG1	-13,39567	,91806	,16761	-13,73848	-13,05286	-79,920	29	,000
Pair 2	NATIVO_MPI_CG2 - OPENSTACK_MPI_CG2	-33,95400	4,12360	,75286	-35,49278	-32,41322	-45,099	29	,000
Pair 3	NATIVO_MPI_CG4 - OPENSTACK_MPI_CG4	-31,91900	2,65813	,48531	-32,91156	-40,92644	-65,771	29	,000
Pair 4	NATIVO_MPI_CG8 - OPENSTACK_MPI_CG8	-39,34733	5,50941	1,00588	-41,40458	-37,29009	-39,118	29	,000
Pair 5	NATIVO_MPI_CG16 - OPENSTACK_MPI_CG16	-172,77633	2,45141	,44756	-173,69170	-171,86096	-386,037	29	,000

Amostras das Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_MPI_CG1	89,7567	40	,45206	,08254
	OPENNEBULA_MPI_CG1	100,5407	40	,52938	,09665
Pair 2	NATIVO_MPI_CG2	96,2983	40	,08099	,01479
	OPENNEBULA_MPI_CG2	108,2920	40	,60529	,11051
Pair 3	NATIVO_MPI_CG4	178,3323	40	,18700	,03414
	OPENNEBULA_MPI_CG4	175,6773	40	1,40336	,25622
Pair 4	NATIVO_MPI_CG8	202,9190	40	3,17605	,57986
	OPENNEBULA_MPI_CG8	256,0017	40	1,88569	,34428
Pair 5	NATIVO_MPI_CG16	501,0357	40	1,44238	,26334
	OPENNEBULA_MPI_CG16	547,5383	40	2,74988	,50206

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_MPI_CG1 - OPENNEBULA_MPI_CG1	-10,78400	,81973	,14966	-11,09009	-10,47791	-72,056	29	,000
Pair 2	NATIVO_MPI_CG2 - OPENNEBULA_MPI_CG2	-11,99367	,60191	,10989	-12,21842	-11,76891	-109,140	29	,000
Pair 3	NATIVO_MPI_CG4 - OPENNEBULA_MPI_CG4	2,65500	1,40938	,25732	2,12873	3,18127	10,318	29	,000
Pair 4	NATIVO_MPI_CG8 - OPENNEBULA_MPI_CG8	-53,08267	3,71577	,67840	-54,47016	-51,69518	-78,246	29	,000
Pair 5	NATIVO_MPI_CG16 - OPENNEBULA_MPI_CG16	-46,50267	3,44614	,62918	-47,78948	-45,21586	-73,910	29	,000

Amostras das Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	OPENSTACK_MPI_CG1	103,1523	40	,78913	,14408
	OPENNEBULA_MPI_CG1	100,5407	40	,52938	,09665
Pair 2	OPENSTACK_MPI_CG2	140,2513	40	4,11749	,75175
	OPENNEBULA_MPI_CG2	108,2920	40	,60529	,11051
Pair 3	OPENSTACK_MPI_CG4	210,2513	40	2,59848	,47441
	OPENNEBULA_MPI_CG4	175,6773	40	1,40336	,25622
Pair 4	OPENSTACK_MPI_CG8	242,2663	40	4,95134	,90399
	OPENNEBULA_MPI_CG8	256,0017	40	1,88569	,34428
Pair 5	OPENSTACK_MPI_CG16	673,8120	40	1,72959	,31578
	OPENNEBULA_MPI_CG16	547,5383	40	2,74988	,50206

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	OPENSTACK_MPI_CG1 - OPENNEBULA_MPI_CG1	2,61167	,85722	,15651	2,29158	2,93176	16,687	29	,000
Pair 2	OPENSTACK_MPI_CG2 - OPENNEBULA_MPI_CG2	21,95933	4,12831	,75372	20,41780	23,50087	29,135	29	,000
Pair 3	OPENSTACK_MPI_CG4 - OPENNEBULA_MPI_CG4	34,57400	3,07584	,56157	33,42546	35,72254	61,567	29	,000
Pair 4	OPENSTACK_MPI_CG8 - OPENNEBULA_MPI_CG8	-13,73533	5,33879	,97472	-15,72887	-11,74180	-14,092	29	,000
Pair 5	OPENSTACK_MPI_CG16 - OPENNEBULA_MPI_CG16	126,27367	3,43537	,62721	124,99088	127,55646	201,326	29	,000

J.6 Teste estatístico para programa NPB-MPI [EP]

Amostras das Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_MPI_EP1	74,7023	40	,73922	,13496
	OPENSTACK_MPI_EP1	75,8457	40	,73181	,13361
Pair 2	NATIVO_MPI_EP2	37,3947	40	,26465	,04832
	OPENSTACK_MPI_EP2	37,9253	40	,34597	,06317
Pair 3	NATIVO_MPI_EP4	18,8953	40	,21051	,03843
	OPENSTACK_MPI_EP4	19,1653	40	,21790	,03978
Pair 4	NATIVO_MPI_EP8	9,5220	40	,06599	,01205
	OPENSTACK_MPI_EP8	9,8403	40	,11214	,02047
Pair 5	NATIVO_MPI_EP9	10,2370	40	,06939	,01267
	OPENSTACK_MPI_EP9	9,9580	40	,12899	,02355
Pair 6	NATIVO_MPI_EP16	5,8743	40	,04297	,00784
	OPENSTACK_MPI_EP16	6,1967	40	,10018	,01829

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_MPI_EP1 - OPENSTACK_MPI_EP1	-1,14333	1,16251	,21224	-1,57742	-,70924	-5,387	29	,000
Pair 2	NATIVO_MPI_EP2 - OPENSTACK_MPI_EP2	-,54067	,50659	,09249	-,71983	-,34150	-5,738	29	,000
Pair 3	NATIVO_MPI_EP4 - OPENSTACK_MPI_EP4	-,27000	,27450	,05012	-,37250	-,16750	-5,387	29	,000
Pair 4	NATIVO_MPI_EP8 - OPENSTACK_MPI_EP8	-,40833	,12537	,02289	-,35515	-,26152	-13,471	29	,000
Pair 5	NATIVO_MPI_EP9 - OPENSTACK_MPI_EP9	,27900	,13939	,02545	,22695	,33105	10,963	29	,000
Pair 6	NATIVO_MPI_EP16 - OPENSTACK_MPI_EP16	-,32233	,11767	,02148	-,36627	-,27839	-15,004	29	,000

Amostras das Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_MPI_EP1	74,7023	40	,73922	,13496
	OPENNEBULA_MPI_EP1	75,2667	40	,40519	,07398
Pair 2	NATIVO_MPI_EP2	37,3947	40	,26465	,04832
	OPENNEBULA_MPI_EP2	37,8077	40	,26010	,04749
Pair 3	NATIVO_MPI_EP4	18,8953	40	,21051	,03843
	OPENNEBULA_MPI_EP4	18,9743	40	,14720	,02687
Pair 4	NATIVO_MPI_EP8	9,5220	40	,06599	,01205
	OPENNEBULA_MPI_EP8	9,6067	40	,06424	,01173
Pair 5	NATIVO_MPI_EP9	10,2370	40	,06939	,01267
	OPENNEBULA_MPI_EP9	10,2577	40	,15251	,02784
Pair 6	NATIVO_MPI_EP16	5,8743	40	,04297	,00784
	OPENNEBULA_MPI_EP16	6,0197	40	,05898	,01077

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_MPI_EP1 - OPENNEBULA_MPI_EP1	-,56433	,89982	,16428	-,90033	-,22834	-3,435	29	,002
Pair 2	NATIVO_MPI_EP2 - OPENNEBULA_MPI_EP2	-,41400	,34017	,06028	-,53629	-,28971	-6,851	29	,000
Pair 3	NATIVO_MPI_EP4 - OPENNEBULA_MPI_EP4	-,07900	,24961	,04557	-,17221	,01421	-1,733	29	,094
Pair 4	NATIVO_MPI_EP8 - OPENNEBULA_MPI_EP8	-,08467	,09755	,01781	-,12109	-,04824	-4,754	29	,000
Pair 5	NATIVO_MPI_EP9 - OPENNEBULA_MPI_EP9	-,02067	,17743	,03239	-,08692	,04559	-,638	29	,529
Pair 6	NATIVO_MPI_EP16 - OPENNEBULA_MPI_EP16	-,14533	,07855	,01434	-,17467	-,11600	-10,134	29	,000

Amostras das Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	OPENSTACK_MPI_EP1	75,8457	40	,73181	,13361
	OPENNEBULA_MPI_EP1	75,2667	40	,40519	,07398
Pair 2	OPENSTACK_MPI_EP2	37,9253	40	,34597	,06317
	OPENNEBULA_MPI_EP2	37,8077	40	,26010	,04749
Pair 3	OPENSTACK_MPI_EP4	19,1653	40	,21790	,03978
	OPENNEBULA_MPI_EP4	18,9743	40	,14720	,02687
Pair 4	OPENSTACK_MPI_EP8	9,8403	40	,11214	,02047
	OPENNEBULA_MPI_EP8	9,6067	40	,06424	,01173
Pair 5	OPENSTACK_MPI_EP9	9,9580	40	,12899	,02355
	OPENNEBULA_MPI_EP9	10,2577	40	,15251	,02784
Pair 6	OPENSTACK_MPI_EP16	6,1967	40	,10018	,01829
	OPENNEBULA_MPI_EP16	6,0197	40	,05898	,01077

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	OPENSTACK_MPI_EP1 - OPENNEBULA_MPI_EP1	,57900	,86901	,15866	,25451	,90349	3,649	29	,001
Pair 2	OPENSTACK_MPI_EP2 - OPENNEBULA_MPI_EP2	,11767	,43715	,07981	-,04557	,28090	1,474	29	,151
Pair 3	OPENSTACK_MPI_EP4 - OPENNEBULA_MPI_EP4	,19100	,24067	,04211	,10487	,27713	4,535	29	,000
Pair 4	OPENSTACK_MPI_EP8 - OPENNEBULA_MPI_EP8	,22367	,13967	,02550	,17151	,27582	8,771	29	,000
Pair 5	OPENSTACK_MPI_EP9 - OPENNEBULA_MPI_EP9	-,29967	,21782	,03977	-,38100	-,21833	-7,535	29	,000
Pair 6	OPENSTACK_MPI_EP16 - OPENNEBULA_MPI_EP16	,17700	,10114	,01846	,13923	,21477	9,586	29	,000

J.7 Teste estatístico para programa NPB-MPI[BT]

Amostras das Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_MPI_BT1	268,6877	40	,33610	,06136
	OPENSTACK_MPI_BT1	285,7057	40	1,40757	,23873
Pair 2	NATIVO_MPI_BT4	176,3103	40	1,13714	,20761
	OPENSTACK_MPI_BT4	203,5703	40	1,93443	,35318
Pair 3	NATIVO_MPI_BT9	236,9863	40	1,07094	,19553
	OPENSTACK_MPI_BT9	318,9167	40	20,08634	3,66725
Pair 4	NATIVO_MPI_BT16	196,7937	40	,78271	,14290
	OPENSTACK_MPI_BT16	259,9037	40	1,40271	,25610

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_MPI_BT1 - OPENSTACK_MPI_BT1	-17,01800	1,39269	,25427	-17,53804	-16,49796	-66,929	29	,000
Pair 2	NATIVO_MPI_BT4 - OPENSTACK_MPI_BT4	-27,26000	2,07850	,37948	-28,03612	-26,48388	-71,835	29	,000
Pair 3	NATIVO_MPI_BT9 - OPENSTACK_MPI_BT9	-81,94033	20,19766	3,68757	-89,47227	-74,38840	-22,218	29	,000
Pair 4	NATIVO_MPI_BT16 - OPENSTACK_MPI_BT16	-63,11000	1,58839	,29000	-63,70311	-62,51689	-217,622	29	,000

Amostras das Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_MPI_BT1	268,6877	40	,33610	,06136
	OPENNEBULA_MPI_BT1	281,0343	40	,81011	,14791
Pair 2	NATIVO_MPI_BT4	176,3103	40	1,13714	,20761
	OPENNEBULA_MPI_BT4	175,0760	40	,86863	,15859
Pair 3	NATIVO_MPI_BT9	236,9863	40	1,07094	,19553
	OPENNEBULA_MPI_BT9	258,8020	40	1,22051	,22283
Pair 4	NATIVO_MPI_BT16	196,7937	40	,78271	,14290
	OPENNEBULA_MPI_BT16	229,1733	40	2,75036	,50215

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_MPI_BT1 - OPENNEBULA_MPI_BT1	-12,34667	,94846	,17316	-12,70083	-11,99251	-71,400	29	,000
Pair 2	NATIVO_MPI_BT4 - OPENNEBULA_MPI_BT4	1,23433	1,38112	,25216	,71862	1,75005	4,895	29	,000
Pair 3	NATIVO_MPI_BT9 - OPENNEBULA_MPI_BT9	-21,81567	1,68539	,40771	-22,44500	-21,18633	-70,897	29	,000
Pair 4	NATIVO_MPI_BT16 - OPENNEBULA_MPI_BT16	-32,37967	2,70875	,49455	-33,39113	-31,36820	-65,473	29	,000

Amostras das Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	OPENSTACK_MPI_BT1	285,7057	40	1,40757	,23873
	OPENNEBULA_MPI_BT1	281,0343	40	,81011	,14791
Pair 2	OPENSTACK_MPI_BT4	203,5703	40	1,93443	,35318
	OPENNEBULA_MPI_BT4	175,0760	40	,86863	,15859
Pair 3	OPENSTACK_MPI_BT9	318,9167	40	20,08634	3,66725
	OPENNEBULA_MPI_BT9	258,8020	40	1,22051	,22283
Pair 4	OPENSTACK_MPI_BT16	259,9037	40	1,40271	,25610
	OPENNEBULA_MPI_BT16	229,1733	40	2,75036	,50215

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	OPENSTACK_MPI_BT1 - OPENNEBULA_MPI_BT1	4,67133	1,56819	,28631	4,08576	5,25690	16,316	29	,000
Pair 2	OPENSTACK_MPI_BT4 - OPENNEBULA_MPI_BT4	28,49433	1,92050	,35063	27,77721	29,21146	81,265	29	,000
Pair 3	OPENSTACK_MPI_BT9 - OPENNEBULA_MPI_BT9	60,11467	19,90314	3,63380	52,68271	67,54662	16,543	29	,000
Pair 4	OPENSTACK_MPI_BT16 - OPENNEBULA_MPI_BT16	40,74033	3,41833	,62410	29,45391	32,00676	49,240	29	,000

APÊNDICE K. TESTES ESTATÍSTICO PARA INFRAESTRUTURA

K.1 Teste estatístico para unidade de armazenamento

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_IOZONE_WRITE	1,5558E6	40	2,01440E5	31850,45192
	OPENSTACK_IOZONE_WRITE	1,4047E6	40	2,49432E5	39438,64881
Pair 2	NATIVO_IOZONE_READ	1,8010E6	40	1,41675E5	22400,84414
	OPENSTACK_IOZONE_READ	1,6879E6	40	2,82167E5	44614,51805
Pair 3	NATIVO_IOZONE_REWRITE	4,8245E6	40	7,92679E5	1,25334E5
	OPENSTACK_IOZONE_REWRITE	2,8751E6	40	2,14465E6	3,39099E5
Pair 4	NATIVO_IOZONE_REREAD	5,1599E6	40	1,89873E5	40021,58904
	OPENSTACK_IOZONE_REREAD	4,6760E6	40	4,94557E5	78196,38065

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_IOZONE_WRITE - OPENSTACK_IOZONE_WRITE	1,51111E5	3,49546E5	55268,00140	39321,36540	2,62902E5	2,734	39	,009
Pair 2	NATIVO_IOZONE_READ - OPENSTACK_IOZONE_READ	1,13147E5	3,20645E5	50698,41840	10599,49445	2,15694E5	2,232	39	,031
Pair 3	NATIVO_IOZONE_REWRITE - OPENSTACK_IOZONE_REWRITE	1,94943E6	2,13166E6	3,37044E5	1,26769E6	2,63116E6	5,784	39	,000
Pair 4	NATIVO_IOZONE_REREAD - OPENSTACK_IOZONE_REREAD	4,83874E5	5,07349E5	80218,90368	3,21616E5	6,46132E5	6,032	39	,000

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_IOZONE_WRITE	1,5558E6	40	2,01440E5	31850,45192
	OPENNEBULA_IOZONE_WRITE	1,2480E6	40	2,63887E5	41724,17779
Pair 2	NATIVO_IOZONE_READ	1,8010E6	40	1,41675E5	22400,84414
	OPENNEBULA_IOZONE_READ	1,5045E6	40	3,44043E5	54398,03474
Pair 3	NATIVO_IOZONE_REWRITE	4,8245E6	40	7,92679E5	1,25334E5
	OPENNEBULA_IOZONE_REWRITE	816551,7000	40	1,77337E6	2,80395E5
Pair 4	NATIVO_IOZONE_REREAD	5,1599E6	40	1,89873E5	40021,58904
	OPENNEBULA_IOZONE_REREAD	4,0353E6	40	6,11027E5	96611,92542

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_IOZONE_WRITE - OPENNEBULA_IOZONE_WRITE	3,07740E5	3,66213E5	57903,32923	1,90609E5	4,24850E5	5,315	39	,000
Pair 2	NATIVO_IOZONE_READ - OPENNEBULA_IOZONE_READ	2,96457E5	3,43805E5	54360,41767	1,86503E5	4,06412E5	5,454	39	,000
Pair 3	NATIVO_IOZONE_REWRITE - OPENNEBULA_IOZONE_REWRITE	4,00799E6	1,88727E6	2,98403E5	3,40441E6	4,61156E6	13,431	39	,000
Pair 4	NATIVO_IOZONE_REREAD - OPENNEBULA_IOZONE_REREAD	1,12463E6	6,17646E5	97658,36885	9,27096E5	1,32216E6	11,516	39	,000

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	OPENSTACK_IOZONE_WRITE	1,4047E6	40	2,49432E5	39438,64881
	OPENNEBULA_IOZONE_WRITE	1,2480E6	40	2,63887E5	41724,17779
Pair 2	OPENSTACK_IOZONE_READ	1,6879E6	40	2,82167E5	44614,51805
	OPENNEBULA_IOZONE_READ	1,5045E6	40	3,44043E5	54398,03474
Pair 3	OPENSTACK_IOZONE_REWRITE	2,8751E6	40	2,14465E6	3,39099E5
	OPENNEBULA_IOZONE_REWRITE	816551,7000	40	1,77337E6	2,80395E5
Pair 4	OPENSTACK_IOZONE_REREAD	4,6760E6	40	4,94557E5	78196,38065
	OPENNEBULA_IOZONE_REREAD	4,0353E6	40	6,11027E5	96611,92542

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	OPENSTACK_IOZONE_WRITE - OPENNEBULA_IOZONE_WRITE	1,56618E5	3,03899E5	48050,70531	59426,69967	2,53810E5	3,259	39	,002
Pair 2	OPENSTACK_IOZONE_READ - OPENNEBULA_IOZONE_READ	1,83311E5	4,38840E5	69385,15631	42966,02435	3,23655E5	2,642	39	,012
Pair 3	OPENSTACK_IOZONE_REWRITE - OPENNEBULA_IOZONE_REWRITE	2,05856E6	2,87906E6	4,55219E5	1,13779E6	2,97933E6	4,522	39	,000
Pair 4	OPENSTACK_IOZONE_REREAD - OPENNEBULA_IOZONE_REREAD	6,40755E5	8,39956E5	1,32809E5	3,72125E5	9,09386E5	4,825	39	,000

K.2 Teste estatístico para memória RAM

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_STREAM_ADD	8,9545E7	40	2,00170E6	3,16497E5
	OPENSTACK_STREAM_ADD	7,9501E7	40	1,00978E6	1,59660E5
Pair 2	NATIVO_STREAM_COPY	8,2310E7	40	1,80772E6	2,85826E5
	OPENSTACK_STREAM_COPY	7,0531E7	40	7,14728E5	1,14008E5
Pair 3	NATIVO_STREAM_SCALE	8,0925E7	40	2,26236E6	3,57711E5
	OPENSTACK_STREAM_SCALE	6,7892E7	40	7,09726E5	1,12218E5
Pair 4	NATIVO_STREAM_TRIADE	8,9747E7	40	2,14756E6	3,39558E5
	OPENSTACK_STREAM_TRIAD	7,8893E7	40	1,02623E6	1,62261E5

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_STREAM_ADD - OPENSTACK_STREAM_ADD	1,00449E7	1,17049E6	1,85070E5	9,67061E6	1,04193E7	54,276	39	,000
Pair 2	NATIVO_STREAM_COPY - OPENSTACK_STREAM_COPY	1,17791E7	1,25097E6	1,97795E5	1,13790E7	1,21791E7	59,552	39	,000
Pair 3	NATIVO_STREAM_SCALE - OPENSTACK_STREAM_SCALE	1,40331E7	1,64203E6	2,59628E5	1,25080E7	1,35583E7	50,199	39	,000
Pair 4	NATIVO_STREAM_TRIADE - OPENSTACK_STREAM_TRIAD	1,08540E7	1,40363E6	2,06122E5	1,04370E7	1,12709E7	52,658	39	,000

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_STREAM_ADD	8,9545E7	40	2,00170E6	3,16497E5
	OPENNEBULA_STREAM_ADD	8,2271E7	40	2,18671E6	3,45749E5
Pair 2	NATIVO_STREAM_COPY	8,2310E7	40	1,80772E6	2,85826E5
	OPENNEBULA_STREAM_COPY	7,3728E7	40	1,97699E6	3,12590E5
Pair 3	NATIVO_STREAM_SCALE	8,0925E7	40	2,26236E6	3,57711E5
	OPENNEBULA_STREAM_SCALE	8,1891E7	40	2,09097E6	3,40611E5
Pair 4	NATIVO_STREAM_TRIADE	8,9747E7	40	2,14756E6	3,39558E5
	OPENNEBULA_STREAM_TRIAD	7,1831E7	40	2,24253E6	3,54575E5

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	NATIVO_STREAM_ADD - OPENNEBULA_STREAM_ADD	7,27433E6	1,28297E6	2,02855E5	6,86402E6	7,68464E6	35,860	39	,000
Pair 2	NATIVO_STREAM_COPY - OPENNEBULA_STREAM_COPY	8,58201E6	4,57920E5	72403,44847	8,43556E6	8,72846E6	118,540	39	,000
Pair 3	NATIVO_STREAM_SCALE - OPENNEBULA_STREAM_SCALE	-9,65217E5	1,07792E6	1,70434E5	-1,40995E6	-6,20482E5	-5,663	39	,000
Pair 4	NATIVO_STREAM_TRIADE - OPENNEBULA_STREAM_TRIAD	1,79163E7	2,59180E6	4,09800E5	1,70874E7	1,87452E7	43,720	39	,000

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	OPENSTACK_STREAM_ADD	7,9501E7	40	1,00978E6	1,59660E5
	OPENNEBULA_STREAM_ADD	8,2271E7	40	2,18671E6	3,45749E5
Pair 2	OPENSTACK_STREAM_COPY	7,0531E7	40	7,14728E5	1,14008E5
	OPENNEBULA_STREAM_COPY	7,3728E7	40	1,97699E6	3,12590E5
Pair 3	OPENSTACK_STREAM_SCALE	6,7892E7	40	7,09726E5	1,12218E5
	OPENNEBULA_STREAM_SCALE	8,1891E7	40	2,09097E6	3,40611E5
Pair 4	OPENSTACK_STREAM_TRIAD	7,8893E7	40	1,02623E6	1,62261E5
	OPENNEBULA_STREAM_TRIAD	7,1831E7	40	2,24253E6	3,54575E5

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	OPENSTACK_STREAM_ADD - OPENNEBULA_STREAM_ADD	-2,77061E6	1,62854E6	2,57494E5	-3,29145E6	-2,24978E6	-10,760	39	,000
Pair 2	OPENSTACK_STREAM_COPY - OPENNEBULA_STREAM_COPY	-3,19706E6	1,37153E6	2,16858E5	-3,63570E6	-2,75842E6	-14,743	39	,000
Pair 3	OPENSTACK_STREAM_SCALE - OPENNEBULA_STREAM_SCALE	-1,39983E7	1,54579E6	2,44410E5	-1,44927E7	-1,35040E7	-57,274	39	,000
Pair 4	OPENSTACK_STREAM_TRIAD - OPENNEBULA_STREAM_TRIAD	7,06232E6	2,32196E6	3,67134E5	6,31972E6	7,80492E6	19,236	39	,000

K.2 Teste estatístico para rede

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_IPERF_BANDWIDTH	94,4275	40	,04522	,00715
	OPENSTACK_IPERF_BANDWITH	66,3575	40	25,69165	4,06221

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference Lower	Upper			
Pair 1	NATIVO_IPERF_BANDWIDTH - OPENSTACK_IPERF_BANDWITH	28,07000	25,70332	4,06405	19,84968	36,29032	6,907	39	,000

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_IPERF_BANDWIDTH	94,4275	40	,04522	,00715
	OPENNEBULA_IPERF_BANDWIDTH	93,8750	40	,49704	,07859

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference Lower	Upper			
Pair 1	NATIVO_IPERF_BANDWIDTH - OPENNEBULA_IPERF_BANDWIDTH	,55250	,51589	,08157	,38751	,71749	6,773	39	,000

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	OPENSTACK_IPERF_BANDWITH	66,3575	40	25,69165	4,06221
	OPENNEBULA_IPERF_BANDWIDTH	93,8750	40	,49704	,07859

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference Lower	Upper			
Pair 1	OPENSTACK_IPERF_BANDWITH - OPENNEBULA_IPERF_BANDWIDTH	-27,51750	25,53252	4,03705	-35,68320	-19,35180	-6,816	39	,000

K.3 Resultado estatístico para processador

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_LINPACK	361966,5161	40	798,83260	126,40652
	OPENSTACK_LINPACK	3,4541E8	40	1,54059E6	2,43589E5

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference Lower	Upper			
Pair 1	NATIVO_LINPACK - OPENSTACK_LINPACK	-3,45046E8	1,54057E6	2,43585E5	-3,45539E8	-3,44554E8	-1416,532	39	,000

Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	NATIVO_LINPACK	361966,5161	40	798,83260	126,40652
	OPENNEBULA_LINPACK	3,5111E8	40	2,08845E6	3,40213E5

Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference Lower	Upper			
Pair 1	NATIVO_LINPACK - OPENNEBULA_LINPACK	-3,50751E8	2,08820E6	3,40173E5	-3,51418E8	-3,50083E8	-1062,324	39	,000

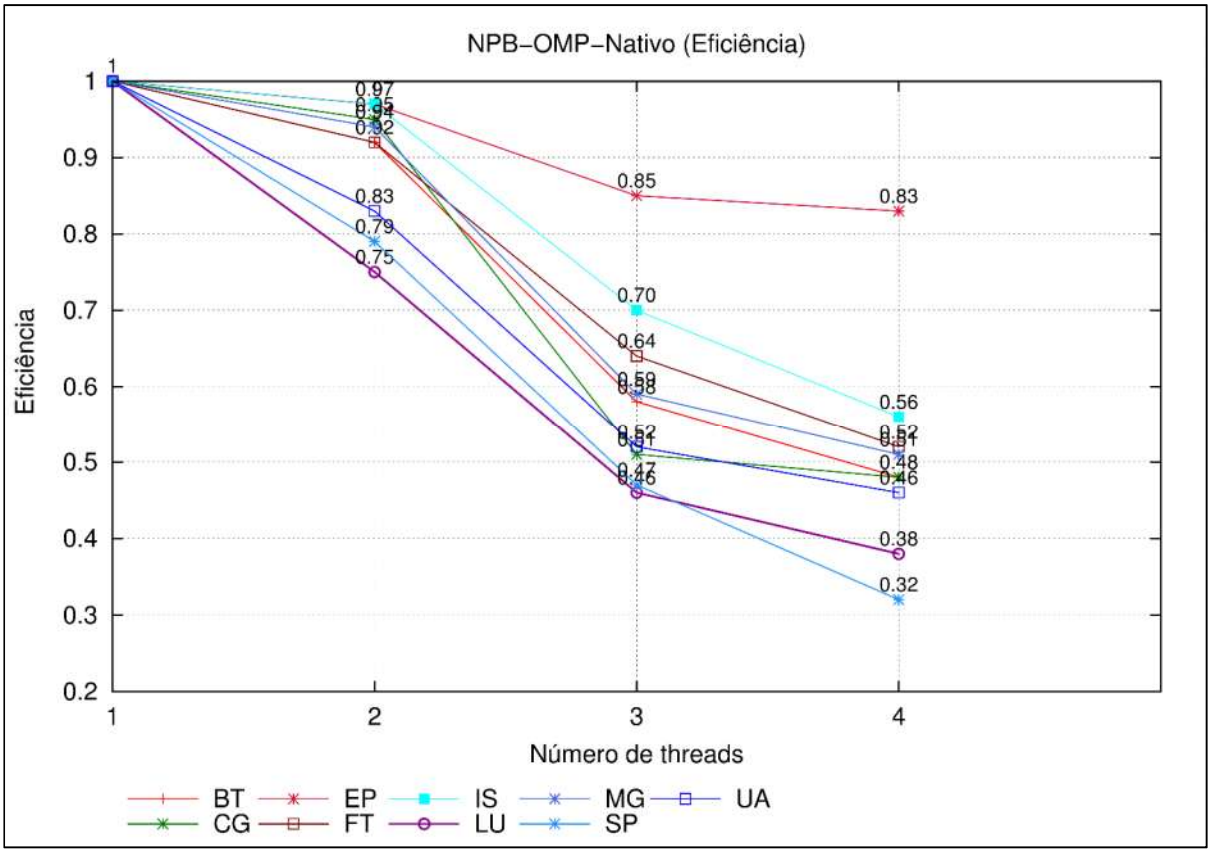
Amostras Estatísticas Pareadas

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	OPENSTACK_LINPACK	3,4541E8	40	1,54059E6	2,43589E5
	OPENNEBULA_LINPACK	3,5111E8	40	2,08845E6	3,40213E5

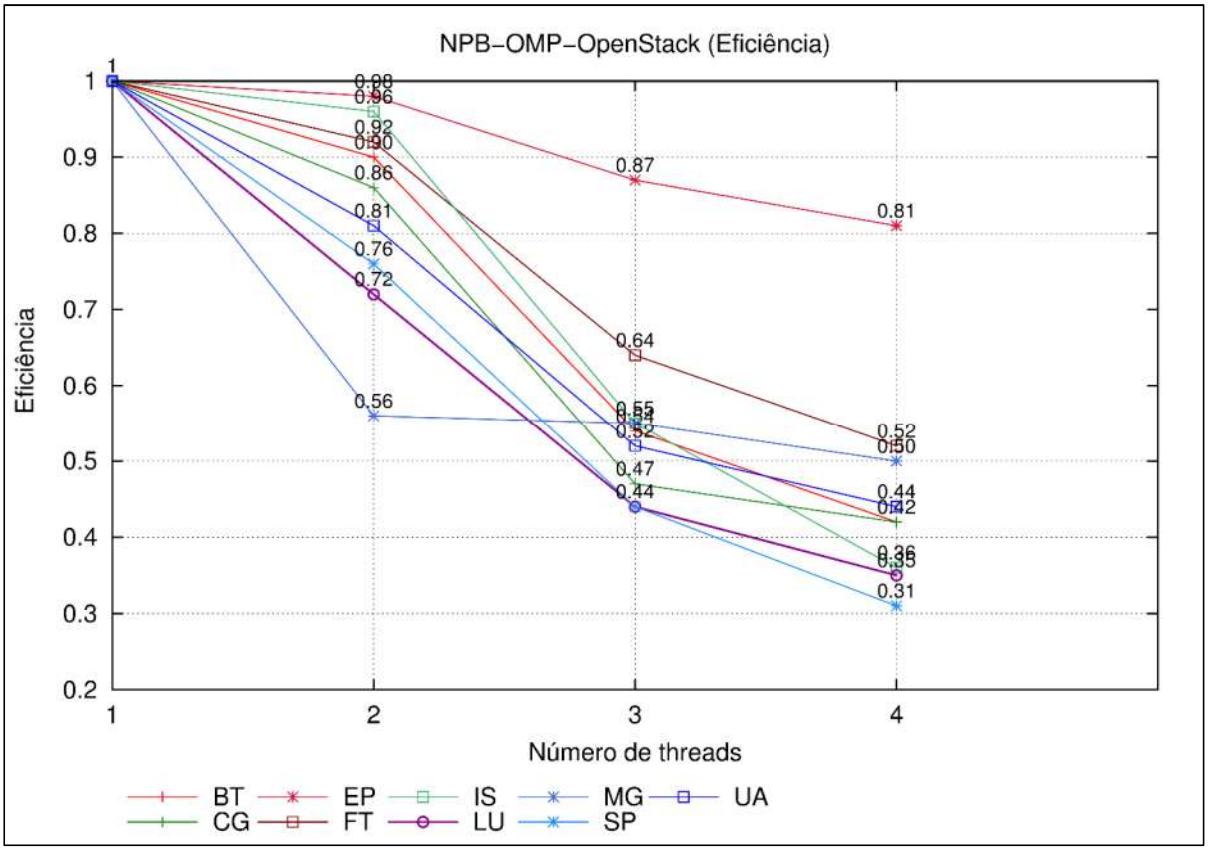
Teste Estatístico das Amostras

		Paired Differences					t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference Lower	Upper			
Pair 1	OPENSTACK_LINPACK - OPENNEBULA_LINPACK	-5,70420E6	2,33337E6	3,68938E5	-6,45044E6	-4,95795E6	-15,461	39	,000

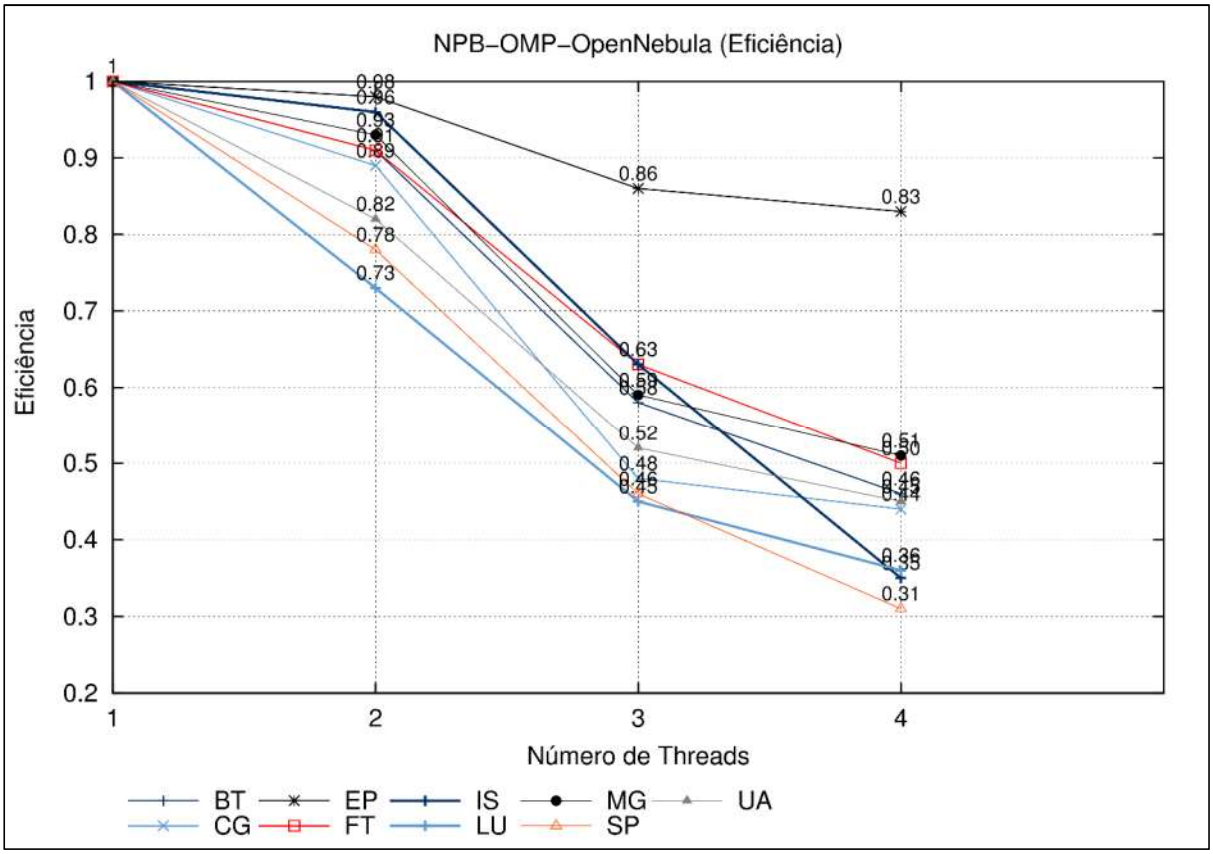
APÊNDICE L. RESULTADO DE EFICIÊNCIA DOS PROGRAMAS DA SUÍTE NPB-OMP – NATIVO



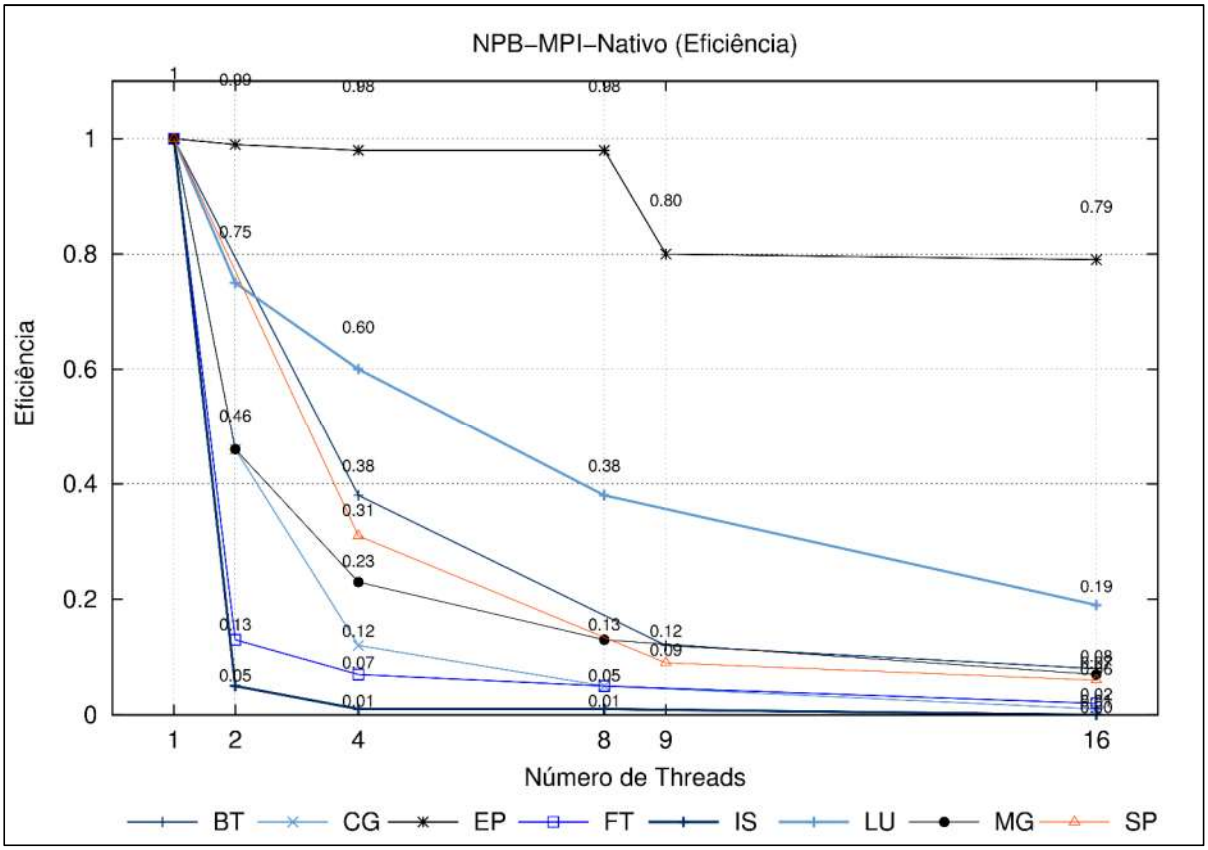
APÊNDICE M. RESULTADO DE EFICIÊNCIA DOS PROGRAMAS DA SUÍTE NPB-OMP – OPENSTACK



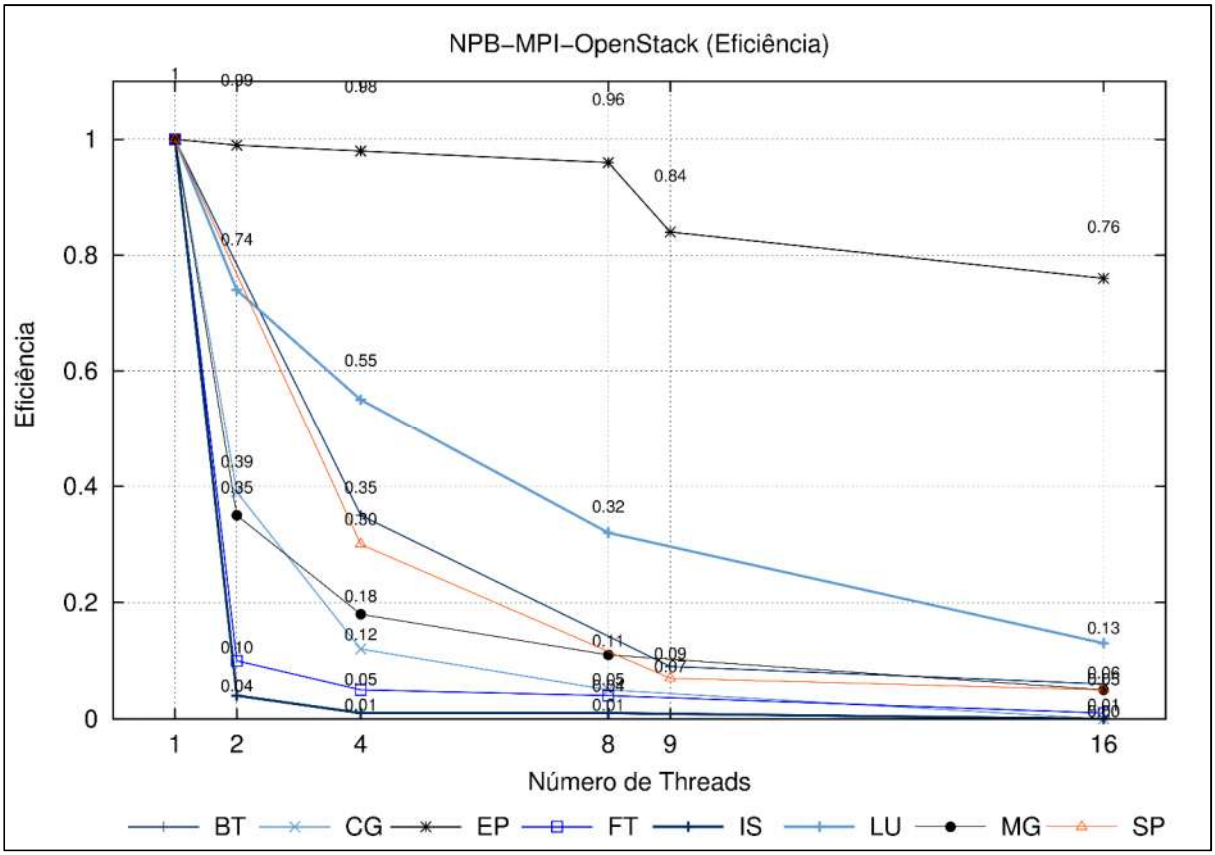
APÊNDICE N. RESULTADO DE EFICIÊNCIA DOS PROGRAMAS DA SUÍTE NPB-OMP – OPENNEBULA



APÊNDICE O. RESULTADO DE EFICIÊNCIA DOS PROGRAMAS DA SUÍTE NPB-MPI – NATIVO



APÊNDICE P. RESULTADO DE EFICIÊNCIA DOS PROGRAMAS DA SUÍTE NPB-MPI – OPENSTACK



APÊNDICE Q. RESULTADO DE EFICIÊNCIA DOS PROGRAMAS DA SUÍTE NPB-MPI – OPENNEBULA