Avaliação de Desempenho da Agregação de Interfaces de Rede em Ambientes de Nuvem Privada HiPerfCloud: High Performance in Cloud

Technical Report · December 2019

DOI: 10.13140/RG.2.2.14800.87044

2 authors, including:

Dalvan Griebler

Pontifícia Universidade Católica do Rio Grande do Sul

95 PUBLICATIONS **218** CITATIONS

Avaliação de Desempenho da Agregação de Interfaces de Rede em Ambientes de Nuvem Privada

Anderson Mattheus Maliszewski, Dalvan Griebler

Três de Maio, Brasil
2019

HiPerfCloud:

High Performance in Cloud

Avaliação de Ambientes de Nuvem IaaS

RT5: Relatório Técnico de Pesquisa (Atividades de 2018)

Reference: Maliszewski, A. M.; Griebler, D. *Avaliação de Desempenho da Agregação de Interfaces de Rede em Ambientes de Nuvem Privada.* Laboratory of Advanced Researches on Cloud Computing (LARCC), Technical Report, 2019.

ID do Documento:	LARCC-HiPerfCloud-RT5
Versão:	1
Autores:	Anderson Mattheus Maliszewski, Dalvan Griebler
Objetivo:	Avaliar a camada de virtualização e agregação de interfaces de rede na nuvem privada
Tarefa:	—
Hardware:	1 *cluster* isolado usando 3 máquinas idênticas, cada uma com 24 GB de RAM (1333 MHz), 2 processadores Intel Xeon X5560 (quad-core 2.80GHz), discos SATA II (7200 RPM) e conectados em uma rede Gigabit (10/1000). 1 *cluster* composto por 2 máquinas idênticas, cada uma com 2 processadores AMD Opteron 2425 HE (six-core 2.1GHz), 32 GB de RAM, 4 Placas de interface de rede Gigabit (NICs) interligadas por um Switch Gigabit.
Ambiente:	Sistema Operacional (Ubuntu Server 18.04), Virtualizadores (KVM, LXC e LXD), CloudStack (vers. 4.8), OpenNebula (vers 5.6.1). Aplicações Científicas (NPB)
Softwares:	GNUPlot (Gráficos), Latex (Documentos).

Tarefa	Responsável	Instituição	Papel	Data
Criado por:	Dalvan Griebler	SETREM/PUCRS	Coordenador	12/12/2018
Editado por:	Anderson M. Maliszewski	SETREM	Pesquisador	22/05/2019
Revisão por:	Dalvan Griebler	SETREM/PUCRS	Coordenador	22/08/2019
Correções por:	Anderson M. Maliszewski	SETREM	Pesquisador	14/12/2019
Revisado por:	Vera Lúcia Benedetti	SETREM	Colaboradora	20/12/2019
Aprovado por:	Dalvan Griebler	SETREM/PUCRS	Coordenador	23/12/2019
	Ildo Corso	SETREM/ABASE	Colaborador	23/12/2019

Lista de colaboradores internos e externos

Abaixo é listado (em ordem alfabética) as pessoas que fizeram contribuições para este relatório técnico:

- Anderson Mattheus Maliszewsi (SETREM)
- Dalvan Griebler (SETREM/PUCRS)

Resumo Geral

O objetivo do **Projeto HiPerfCloud (*High Performance in Cloud*)** é avaliar o desempenho em ambientes de nuvens IaaS (*Infrastructure as a Service*) e analisar as características de implantação e gerenciamento nas ferramentas disponíveis. Este documento da continuidade aos Relatório Técnicos anteriores - RT1-2015 [18], RT2-2016 [33], RT3-2017 [19] e RT4-2018 [17], mostrando novos resultados usando a implantação de nuvem privada CloudStack, na qual foi realizada uma avaliação de desempenho comparativa de duas tecnologias de virtualização, sendo elas o KVM (virtualização a nível de Kernel) e LXC (virtualização a nível de Sistema Operacional), usando aplicações que demandam alto desempenho em instâncias de nuvem com recursos dedicados (um usuário *single-tenant)* e compartilhados (dois usuários *multi-tenant)*. Além disso, outros resultados obtidos que serão abordados nesse documento, utilizaram a implantação de nuvem privada OpenNebula para realização da agregação de interfaces de rede usando contêineres LXD e verificando se esta abordagem provê melhor desempenho para as mesmas aplicações previamente utilizadas.

Contexto do Relatório

Este documento é o quinto Relatório Técnico relativo ao **Projeto HiPerfCloud** e apresenta resultados com foco em duas avaliações distintas, que são tidas como principais pontos de gargalo em nuvens computacionais. Primeiro em relação a camada de virtualização e segundo em relação a rede. Foram avaliados o desempenho dos dois virtualizadores mais presentes na *cloud* (KVM e LXC) usando aplicações científicas executadas com e sem concorrência de recursos. Também são avaliadas os benefícios que a agregação de rede proporciona para instâncias criadas na núvem privada, na qual são executadas aplicação cientificas com diferentes demandas de rede.

Estrutura do Relatório

Este documento inicialmente apresenta a estrutura geral. Em seguida, a computação em nuvem é contextualizada em conjunto com as virtualizações utilizadas, as metodologias de agregação de interfaces de rede e as ferramentas IaaS de código aberto. Por fim, são apresentados as características dos ambientes implantados, os experimentos realizados e a análise dos resultados obtidos. Ao final deste documento, estão inseridos os 4 artigos que foram publicados usando os resultados aqui apresentados.

Sumário

1. Introdução

A computação em nuvem é um modelo de serviços que fornece soluções alternativas aos custos de aquisição e manutenção de ambientes de computação, baseando-se principalmente em dois itens principais [4, 2, 30]. Sendo eles:

- O provisionamento de um ambiente elástico, no qual o seu usuário pode acessar, adquirir e liberar recursos de computação de acordo com a demanda atual.
- A precificação das instâncias usando o modelo de cobrança de pagamento por uso, no qual o usuário é cobrado apenas pelo período de tempo em que os recursos foram utilizados.

Ao contrário dos sistemas tradicionais, não há investimentos iniciais em infraestrutura e licenças de software. Devido à elasticidade e ao modelo de cobrança de pagamento por uso, o custo total de manutenção pode ser próximo de zero quando os recursos não estão em uso. Além disso, os custos de instalações, depreciação de hardware, consumo de energia e refrigeração são eliminados ou, pelo menos, significativamente reduzidos. A quantidade de recursos disponíveis é praticamente ilimitada. Portanto, a motivação de mover aplicações para a nuvem é reduzir esses custos, aumentando a escalabilidade e disponibilidade.

A computação de alto desempenho (HPC) exige muito poder computacional sendo necessário altos investimentos e o dimensionamento correto destes para sua obtenção. O grande problema é a dimensão destes investimentos. Se a infraestrutura for superdimensionada, os recursos financeiros foram mal alocados, uma vez que os recursos de computação estão sendo subutilizados. Por outro lado, se a infraestrutura tiver sido dimensionada apenas para a demanda atual, quaisquer aumentos futuros na demanda computacional não serão atendidos, exigindo novos investimentos. Além disso, existem outros custos envolvidos na infraestrutura em execução que se materializam na manutenção, consumo de energia, depreciação do hardware, refrigeração e instalações. Esses custos estão relacionados aos ambientes HPC e a qualquer ambiente de computação de uso geral.

As características da computação em nuvem tornam o modelo uma alternativa interessante para o uso como ambientes HPC. Além disso, a computação em nuvem pode ser mais ecológica em comparação com a computação tradicional. Como os recursos físicos não são acessíveis pelos usuários, a plataforma em nuvem pode migrar a carga de trabalho de um servidor para outro e realizar a consolidação de máquinas virtuais. Com isso, é possível desligar as máquinas não utilizadas. A possibilidade de usar um grande conjunto de máquinas sem a necessidade de adquiri-las tem uma enorme vantagem, especialmente para usuários em projetos temporários ou na execução de aplicações sem o uso constante das máquinas.

Entretanto, esse ambiente não esta livre de problemas, que na maioria das vezes esta relacionado com a perda de desempenho imposta pela camada de virtualização e pela rede. A virtualização é tida como causadora de perdas de desempenho pois esta as induz se comparada ao ambiente nativo. Os esforços para reduzir estas perdas culminaram com a criação de contêineres, que utilizam uma virtualização leve, conhecida como virtualização a nível de sistema operacional [35]. O segundo problema diz respeito a área de HPC, que busca extrair o máximo dos recursos com determinada capacidade nominal dos servidores. Isso, por sua vez, só é atingido com a garantia de exclusividade de recursos, ou seja, que todos os recursos da máquina estejam disponíveis para determinada tarefa. Sendo assim, a utilização da *cloud* para HPC, não utiliza a premissa *multi-tenancy* ou multi-usuário, uma metodologia chave para este ambiente.

Embora essa alocação possa ser garantida com recursos como memória, processador e armazenamento, ela não é assegurada em relação a interconexão de rede, pois os equipamentos

de comutação são compartilhados entre diversos servidores, levando ao processamento de diversos fluxos originados de instâncias distintas [25]. Portanto, neste trabalho, foram objetivadas avaliações nos dois problemas descritos, comparando primeiramente o desempenho de duas virtualizações diferentes em conjunto com o sistema nativo, assim como uma metodologia de agregação de rede, que tem por definição aumentar a capacidade de rede, usando aplicações HPC em ambientes de nuvem privada. A seguir, são definidas genericamente a visão geral do trabalho, assim como as principais terminologias e a estrutura do documento.

1.1 Visão Geral

Neste relatório são apresentados os resultados das avaliações realizadas com as nuvens privadas criadas com os gerenciadores de nuvem CloudStack e OpenNebula, utilizando diversas metodologias de avaliação de desempenho em relação aos problemas já mencionados, sendo eles o impacto da virtualização e a interconexão de redes.

1.2 Terminologia

- **Infraestrutura:** Representa os recursos de processamento: Memória RAM, armazenamento, rede e processador.
- **Aplicações Paralelas:** Área da computação de alto desempenho.
- **Benchmark:** Programa para teste específico de determinado recurso ou serviço.
- **Cluster**: Conjunto de computadores interligados por uma rede somando recursos.
- **OpenNebula**: Ferramenta de código aberto para gerenciamento de infraestrutura de nuvem IaaS.
- **CloudStack**: Ferramenta de código aberto para gerenciamento de infraestrutura de nuvem IaaS.
- **KVM**: *Kernel-based Virtual Machine* (KVM) solução de virtualização completa para ambientes Linux.
- **LXC:** *Linux Containers* (LXC) solução de virtualização a nível de sistema operacional para ambientes Linux.
- **LXD:** *Linux Containers* (LXD) solução de virtualização a nível de sistema operacional para ambientes Linux.
- **NPB-MPI**: Carga de trabalho composto por diferentes *kernels* de simulação de aplicações paralelas com bibliotecas MPI.
- **NPB-OMP**: Carga de trabalho composto por diferentes *kernels* de simulação de aplicações paralelas com bibliotecas OMP.
- **NetPIPE**: Ferramenta utilizada para avaliação do desempenho de rede.

1.3 Estrutura deste Documento

Este documento está organizado em cinco capítulos:

- Capítulo 1: Apresenta um visão geral do documento.
- Capítulo 2: Nesta seção, encontra-se o referencial sobre a computação em nuvem e seus serviços, bem como sobre as diferentes virtualizações utilizadas, metodologias de agregação e ferramentas IaaS de código aberto.
- Capítulo 3: Apresenta a metodologia utilizada para a execução dos experimentos nos ambientes implantados e os resultados dos testes.
- Capítulo 4: Conclusão do estudo a partir dos resultados.

2. Contexto

Este relatório técnico consiste em demonstrar os resultados obtidos através da pesquisa realizada com nuvens privadas IaaS. Dessa forma, são primeiramente apresentados os resultados da avaliação de desempenho utilizando o gerenciador de nuvem privada CloudStack, na qual foram criadas instâncias com dois modos de virtualização, assim como, avaliando o ambiente com um único usuário e dois usuários concorrentes e posterior, os resultados da avaliação da metologia de agregação de interfaces de rede usando o gerenciador de nuvem privada OpenNebula. Neste trabalho, buscou-se aprofundar o conhecimento e obter resultados relacionados aos impactos da virtualização, utilizada como base na computação em nuvem, assim como sobre a interconexão de rede, que como foi anteriormente descrita, é tida como uma das principais causas de perdas de desempenho de aplicações sendo executadas na nuvem. No Capítulo 3 são apresentados os resultados e por fim no Capítulo 4 as conclusões de ambas avaliações.

2.1 Computação em Nuvem

A computação em nuvem é um modelo que fornece acesso sob demanda para um conjunto de recursos de computação que podem ser disponibilizados com esforços mínimos, exigindo apenas uma conexão de rede [20]. Este modelo foi desenvolvido com a combinação e evolução da Computação Distribuída e Virtualização, com fortes contribuições vindas da Computação em Grade e Paralela. Um serviço de computação em nuvem precisa apresentar as seguintes características para ser considerado aderente à definição do NIST [20] ().

- **Autoatendimento sob demanda**: Essa característica significa que é possível alocar e desalocar recursos computacionais sem a necessidade de interação com a equipe do fornecedor.
- **Amplo acesso à rede**: Todos os serviços oferecidos pelo provedor devem estar acessíveis em uma rede, LAN ou Internet e o acesso precisa estar disponível através de mecanismos padrão.
- **Agrupamento de recursos**: O provedor possui e gerencia um conjunto de recursos de computação física e atende a vários usuários de acordo com sua demanda.
- **Rápida elasticidade**: O provedor permite que o usuário aloque e desaloque recursos em qualquer quantidade, a qualquer momento. Para o usuário, os recursos disponíveis parecem ilimitados.
- **Serviço medido**: O provedor é responsável por controlar o uso e a disponibilidade dos recursos. Essas informações precisam ser disponibilizadas ao usuário para garantir o SLA (contrato de nível de serviço). Eles também são usados para fins de cobrança.

O resumo dessas características estabelece basicamente que a computação em nuvem precisa estar acessível pela Internet e deve fornecer elasticidade dos recursos em um modelo de cobrança por uso. Além das características, a nuvem divide-se em três camadas, conhecidas como IaaS (Infraestrutura como Serviço), PaaS (Plataforma como Serviço) e SaaS (Software como serviço) [20]. Na Figura 2.1 as camadas da computação em nuvem são representadas em um formato de pilha, sendo a primeira delas a de software como serviço, com maior nível de abstração e o menor nível de controle devido a distância que possuí da infraestrutura subjacente, bem como permissões estabelecidas. Na outra extremidade apresenta-se a infraestrutura como

serviço, com menor nível de abstração e maior nível de controle, justamente pela proximidade com os recursos físicos. No meio termo, esta localizado a plataforma como serviço, que dispõe de níveis de abstração e controle medianos, se comparados as duas extremidades.

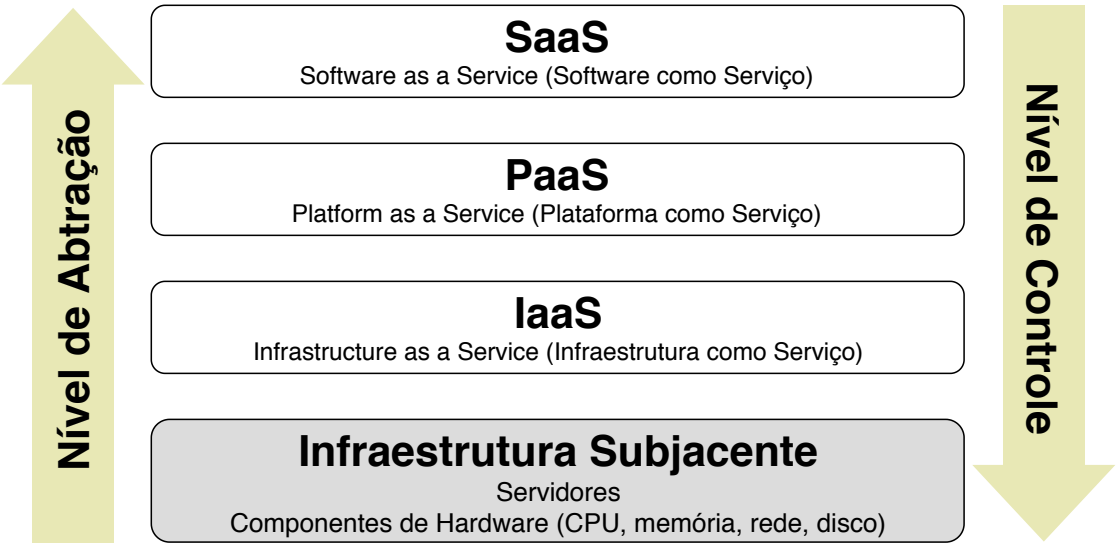


Figura 2.1: Visão geral das 3 camadas padrão da computação em nuvem.

Abaixo, as 3 camadas são descritas com maiores detalhes:

- **IaaS (Infraestrutura como Serviço)** fornece acesso a recursos de TI virtualizados para computação, armazenamento e rede. Geralmente, os recursos são máquinas virtuais (VM), armazenamento acessível em rede e configurações de rede. O usuário deste serviço é um administrador do sistema, que interage com o provedor para alocar, configurar e desalocar os recursos. Esses recursos podem ser disponibilizados ao usuário final; portanto, temos duas funções, uma para o administrador do sistema e outra para o usuário final, mas normalmente é apenas o administrador que interage com o provedor. O provedor é responsável por manter os serviços de baixo nível, como máquinas físicas, rede e hypervisor. Também é responsabilidade do provedor, manter um conjunto atualizado de imagens do sistema operacional, definir as características das instâncias de VMs disponíveis e alocação de recursos no hardware físico. Um dos benefícios do IaaS é que o usuário tem total controle sobre os recursos, pois possui acesso administrativo às VMs. A capacidade de alugar recursos computacionais é uma vantagem, pois o usuário pode alocar e desalocar recursos de acordo com sua demanda. Um grande problema no IaaS é a infraestrutura de rede que é compartilhada entre várias VMs. Ela é alocada a partir de um *pool* comum e os componentes não têm o mesmo nível de isolamento que os servidores físicos; perdas de desempenho tornam-se obvias devido a esse compartilhamento.

- **PaaS (Plataforma como Serviço)** fornece um kit de ferramentas completo que permite ao usuário desenvolver e gerenciar aplicações. O usuário desse modelo de serviço normalmente é um técnico, como administradores de aplicações, desenvolvedores e testadores. O provedor define e suporta um conjunto de ferramentas de desenvolvimento. Esta definição abrange as linguagens de programação suportadas e suas respectivas interfaces de programação de aplicações (API). Todos os recursos físicos são controlados pelo provedor, é de sua responsabilidade manter toda a infraestrutura desenvolvida no serviço. O provedor também é responsável por manter todos os componentes da plataforma atualizados, por exemplo, aplicando *patches* de segurança ou novas versões de determinadas bibliotecas. Os benefícios da adoção de PaaS para o usuário são o baixo custo, e na maioria dos casos,

nenhum custo para desenvolver uma aplicação e disponibilizá-la aos clientes. Nesse caso, o usuário será cobrado exatamente quando seu cliente usar a aplicação. O Google App Engine e o Microsoft Windows Azure são exemplos de fornecedores comerciais nos quais o desenvolvedor não precisa pagar nada pelo desenvolvimento e implantação de uma aplicação, com certas limitações. A escalabilidade da plataforma é garantida pelo provedor, o usuário não tem custos iniciais para adicionar mais poder de computação a aplicação. O usuário precisa seguir as diretrizes do provedor para desenvolver sua aplicação e isso causa uma situação em que a aplicação desenvolvida depende completamente do provedor. Essa situação causa uma dependência tecnológica do provedor e o usuário não pode usar sua aplicação em outra plataforma sem modificações. Essa dependência do provedor é a principal restrição desse modelo de serviço.

- **SaaS (Software as a Service)** por sua vez, fornece ao usuário a capacidade de usar uma aplicação implantada como serviço pela Internet. O usuário do SaaS pode ser uma única pessoa que deseja ter acesso a determinado software ou uma organização que fornece acesso aos seus membros. As principais responsabilidades do fornecedor são garantir que o software fornecido seja suportado e testado. O provedor também é responsável por garantir a confidencialidade dos dados, o que significa que os dados de determinado usuário não podem ser acessados por outros. O controle de segurança do acesso do usuário, os sites e procedimentos de recuperação de desastres e a verificação de conformidade também são de responsabilidade do fornecedor, além de questões de plataforma, como sistema operacional, dispositivos de hardware e opções de configuração. Em geral, nenhum procedimento de instalação e configuração é necessário, isso diminui o tempo para a distribuição do software. O uso eficiente de licenças de software é um dos benefícios. Existem muitos modelos de licença de software que podem ser usados, por exemplo, é possível que uma empresa com 150 funcionários decida usar um aplicação de gerenciamento de projetos baseado em nuvem, e todos os funcionários precisem ter acesso a ele, mas não necessariamente no mesmo tempo. Nessa situação, é possível que a empresa adquira um número limitado de licenças, como 50, e pode fornecer acesso simultâneo a 50 funcionários, mas todos os 150 funcionários têm acesso a aplicação. O número de usuários simultâneos é controlado pelo provedor. Os problemas deste serviço estão relacionados principalmente à segurança, inclusive no acesso ao usuário e na confidencialidade dos dados. O acesso é feito normalmente usando um navegador da Web pela Internet, sendo este um ambiente não seguro e com a possibilidade de ocorrer vazamentos de dados. O acesso à Internet não está sob controle do provedor nem do usuário; em um cenário real, é possível que o usuário não possa acessar o serviço, mesmo quando estiver em funcionamento.

Por fim, a nuvem segundo sua definição [20] possuí 4 modelos de implementação, sendo eles:

- **Nuvem Privada** é o modelo em que os recursos físicos são controlados pelo usuário, um único usuário ou uma organização. Os cenários mais comuns de uso desse modelo são uma organização que decide virtualizar seu *datacenter* (configuração no local) ou se uma organização deseja alugar todos os recursos de um provedor (cenário terceirizado). Nos dois cenários, o usuário tem exclusividade sobre os recursos físicos. Em termos de custo, esse modelo é o mais caro dos quatro. Isso ocorre porque o usuário precisa comprar e manter os recursos sozinho ou alugar recursos exclusivos de um provedor. Os recursos são dedicados à organização, então não há outros usuários acessando-o. Esse modelo atinge o mesmo nível de isolamento e segurança que o usuário dos recursos computacionais tradicionais. Essas características tornam este modelo o mais seguro de todos.

- **Nuvem Comunitária** pode ser considerada como uma derivação de uma nuvem privada. As principais preocupações relacionadas a ambos são as mesmas, como políticas de segu-

rança e gerenciamento de recursos. A diferença entre os dois modelos é que os recursos da nuvem são compartilhados entre as organizações, ou seja, os usuários de uma nuvem comunitária são um grupo de organizações, conhecido como membros. Geralmente, os membros compartilham interesses, podem ser empresas diferentes pertencentes ao mesmo grupo econômico ou departamentos diferentes de uma instituição pública.

- **Nuvem Pública** é o modelo de implementação geralmente tratado como a definição real de computação em nuvem. Os recursos sempre pertencem a um provedor e são compartilhados com seus vários usuários. O provedor define suas políticas e preços, e os usuários precisam concordar com eles para ter acesso aos recursos. O usuário desse modelo pode ser uma organização inteira ou apenas uma pessoa, porque é possível alocar apenas uma máquina pequena e barata. Este modelo é aderente a todas as características essenciais, principalmente o pagamento por uso é totalmente realizado, porque não há custos iniciais para o usuário. A elasticidade é concedida pela infraestrutura do provedor e, para o usuário, parece ser ilimitada. O acesso à nuvem é feito através de uma conexão com a Internet. O controle do usuário é definido pelo provedor e o usuário precisa usar suas credenciais para ter acesso aos recursos. Normalmente, cada usuário tem credenciais exclusivas, mesmo quando são usuários de uma organização com uma conta corporativa.

- **Nuvem Híbrida** é um modelo que combina dois ou mais modelos privados, públicos e também comunitários. No entanto, essa "agregação" apresenta uma grande complexidade na configuração, pois pode mudar à medida que os provedores entram e saem da nuvem. Um cenário mais realista é uma organização que possui uma nuvem privada conectada a um provedor de nuvem pública. Nesse caso, a parte pública é uma extensão da parte privada. Quando os recursos dedicados da organização são totalmente utilizados, a organização pode alocar novos recursos no provedor público. Para esse cenário, a plataforma em nuvem usada internamente pela organização precisa ser compatível com o provedor público.

2.2 Virtualização

A virtualização é uma tecnologia que mudou a maneira como os serviços são fornecidos e é considerada a tecnologia principal para criação da computação em nuvem. Ela possibilita um melhor uso e eficiência dos recursos, instalando vários sistemas operacionais em diferentes máquinas virtuais no mesmo *hardware* [22]. Neste trabalho utilizamos dois tipos distintos de virtualização, sendo elas a de nível de *kernel* (KVM) e de sistema operacional (LXC). Ambas são descritas abaixo.

2.2.1 KVM

É uma solução de código aberto que cria um ambiente de virtualização completa, que por sua vez permite que o sistema operacional crie máquinas virtuais com sistemas operacionais convidados não modificados, transformando o sistema operacional Linux em um *hypervisor*. Utiliza extensões de virtualização (Intel VT-x ou AMD-V), que permitem que um programa do espaço do usuário utilize os recursos de virtualização de hardware de vários processadores [12]. Todo máquina virtual criada é tratada com um processo regular do Linux e conta com hardware virtual dedicado e isolamento [23]. Devido a forma como implementa as máquinas virtuais (usando a virtualização completa) é tido na literatura cientifica, como uma virtualização que tende a afetar o desempenho do ambiente ou mesmo de aplicações que nela executam, uma

vez que cada VM possui um sistema operacional próprio, com *kernel* separado do sistema subjacente [24].

2.2.2 LXC

É uma virtualização em nível de Sistema Operacional (SO), que abstrai os recursos computacionais por meio de Grupos de Controle (*cgroups*) e gerencia os recursos do sistema usando *namespaces*, desta forma criando e isolando os chamados contêineres [13][7]. Esses contêineres compartilham o mesmo *kernel* com o sistema operacional nativo e, portanto, seus processos e sistema de arquivos são acessíveis a partir do *host* que os esta "hospedando", tornando possível executar instruções nativas sem nenhum mecanismo especial de interpretação. Assim, através do uso de contêineres, o sistema operacional dá as aplicações a ilusão de estarem executando em máquinas separadas enquanto compartilham os recursos subjacentes. Portanto, contêineres são definidos como uma virtualização leve, que não requer emulação do hardware físico. O principal objetivo do uso de Linux Containers é fornecer recursos ou o mesmo ambiente que uma máquina virtual, utilizando uma menor quantidade de recursos computacionais.

A Figura 2.2 mostra uma comparação entre LXC (virtualização no nível do SO) e KVM (virtualização completa). Como pode ser visto, ambas as tecnologias usam a API Libvirt, o LXC requer menos abstração de software, usando o mesmo *kernel* do sistema operacional nativo e usando as Bridges Linux ou interfaces nativas para conectividade de rede, enquanto o KVM usa o *driver* VirtIO para operações de E/S e conectividade de rede oferecida às VMs.

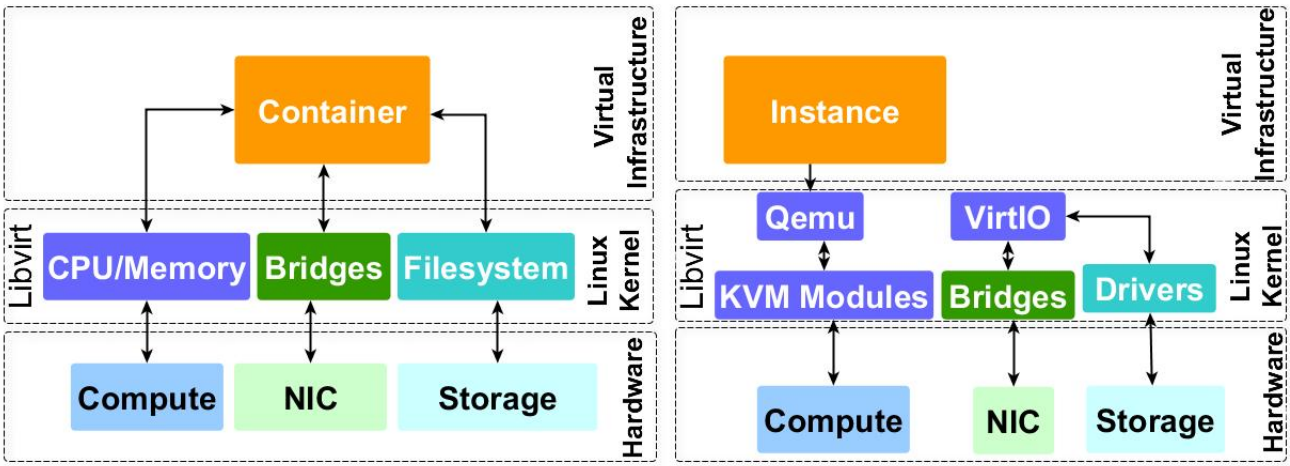


Figura 2.2: Visão geral do LXC (direita) e KVM (esquerda). Extraído de [34]

2.3 Agregação de Placas de Rede

Existem várias terminologias para a agregação de placas de rede, entre elas a agregação de links ou de placas de rede são as utilizadas neste documento. Ambas representam uma técnica que permite a combinação de várias interfaces físicas de rede em uma interface lógica visível ao sistema operacional. Seu principal objetivo é melhorar a capacidade da rede, aumentando a velocidade de transmissão e diminuindo a latência. A velocidade do *link* lógico consiste na soma das velocidades de todos os *links* físicos utilizados. Por exemplo, quando criado a agregação de quatro *links* de 100 Mbps, a velocidade teórica total seria de 400 Mbps. Na Figura 2.3 é mostrado uma visão conceitual da agregação usando duas portas no *switch* e duas interfaces de rede no lado do servidor, sendo estabelecido uma agregação de dois *links*.

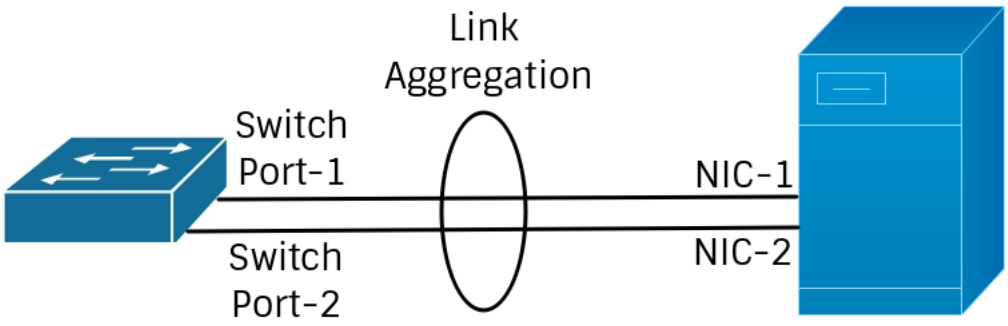


Figura 2.3: Visão conceitual da agregação de *link*.

Uma característica importante da agregação é que quando uma interface de rede falha, o tráfego desta interface pode ser distribuído entre as demais interfaces de rede que foram agregadas, e quando a taxa de transmissão for lenta, pode-se então, a partir de uma configuração específica, priorizar o envio de determinados dados. Além de ser uma técnica eficiente para aumentar a largura de banda quando necessário, há a possibilidade de gerar economia, caso o servidor já possua mais de uma placa de rede. Existem algumas métodos de agregação de interfaces, neste trabalho utilizamos principalmente o método conhecido como Bonding e também o método MPTCP. Ambos estão descritos abaixo.

2.3.1 Bonding

É um método para agregar várias interfaces de rede em uma única interface de ligação lógica. O comportamento das interfaces ligadas depende do modo de configuração sendo utilizado. Atualmente, existe o suporte para 7 modos de configuração diferentes, cada qual objetivando diferentes necessidades do ambiente. Esses modos são configuradas diretamente do arquivo de configuração de interfaces de rede do Linux, e são expressados tanto pelo nome, quanto pelo número [9]. Abaixo estão listados e descritos os 7 modos.

- **Balance-rr ou 0**: Implementa uma política *Round-robin*, transmitindo todos os pacotes em sequência do primeiro nó escravo até o último, fornecendo balanceamento de carga e tolerância a falhas.
- **Active-backup ou 1**: Implementa uma política *Active-backup*, onde somente um escravo (placa de rede) permanece ativo, a única possibilidade de outro escravo tornar-se ativo é somente em caso de falha. Neste modo, o endereço *MAC* fica visível em apenas uma porta do adaptador de rede, afim de evitar confundir-se entre os escravos. Este modo oferece tolerâncias a falhas.
- **Balance-xor ou 2**: Implementa uma política *XOR*, na qual seleciona uma interface para a transmissão de pacotes com base no resultado de uma operação XOR na contagem de interfaces de rede escravas do módulo de endereços MAC de origem e destino. Este cálculo garante que a mesma interface seja selecionada para cada endereço MAC de destino usado. Este modo fornece tolerância a falhas e balanceamento de carga.
- **Broadcast ou 3**: Implementa uma política *Broadcast*, permitindo que o trafego de dados ocorra em todas as interfaces escravas, o qual fornece tolerância a falhas.
- **802.3ad ou 4**: Implementa o protocolo *IEEE 802.3ad Dynamic link aggregation*, criando grupos de agregação com as mesmas velocidades e configurações *duplex*. Utiliza todos os escravos no agregador ativo, de acordo com a especificação 802.3ad. Este modo possui

dois pré-requisitos, que são: suporte *Ethtool* nos *drivers*, além de um *switch* que suporte a agregação de link *IEEE 802.3ad*.

- **Balance-tlb ou 5**: Implementa uma transmissão adaptativa de balanceamento de carga, onde efetua a ligação entre os canais que não necessitam de nenhum suporte especial no *switch*. O tráfego de saída é distribuído de acordo com a carga atual da rede em cada escravo, levando em consideração a velocidade de tráfego. O tráfego de entrada é recebido pelo escravo atual. Se o escravo receptor falhar, outro escravo assumirá o endereço MAC do escravo receptor que falhou. Este modo necessita que o sistema possua o *Ethtool*.

- **Balance-alb ou 6**: Implementa um balanceamento de carga adaptativo o qual inclui os modos `balance-tlb` mais `receive load balancing` para tráfego IPV4, não necessitando de nenhum suporte especial no *switch*. O balanceamento de carga de recebimento é obtido através da negociação de ARP.

2.3.2 MPTCP (Multipath TCP)

É um conjunto de extensões de protocolo propostas ao TCP convencional, padronizado pelo IETF, que permite que o TCP use efetivamente vários caminhos, dividindo um fluxo TCP em vários sub-fluxos. Os sub-fluxos aproveitam as várias interfaces de rede disponíveis em cada servidor para percorrer vários caminhos disponíveis. Tem como finalidade explorar caminhos de comunicação disponíveis, criando sub-fluxos pelos quais são transmitidos os dados, e desta forma, obtendo melhor desempenho se comparado ao TCP, distribuindo a carga do trafego para mais de um sub-fluxo [27].

A motivação de seu desenvolvimento inclui o aumento do *throughput*, utilização geral de recursos, e a resiliência para falhas de rede. Além disso, oferece o mesmo transporte confiável, em ordem, do fluxo de bytes que o TCP, alem de ter sido projetado para ser compatível com versões anteriores das aplicações e da camada de rede. Requer suporte dentro da pilha de rede nos dois pontos da extremidade (transmissor e receptor por exemplo) [27][6].

O MPTCP possui muitas propriedades desejáveis que podem oferecer muitos benefícios em ambientes de *clusters* ou mesmo na computação em nuvem. Isso inclui:

- **Agregação da largura de banda**: Essencialmente, o MPTCP expõe múltiplos caminhos de rede como uma única conexão. Cada sub-fluxo envia os dados em um caminho na rede, e os recursos da rede neste caminho são agregados.

- **Resiliência aprimorada**: O uso de vários caminhos apresenta uma vantagem óbvia de confiabilidade. Se um caminho estiver inacessível, a conexão MPTCP poderá continuar a transferência de dados nos sub-fluxos restantes, resultando em maior resiliência contra falhas de rede.

- **Prontidão operacional**: O MPTCP opera no nível do *host* final e não requer suporte da rede subjacente, além de ser transparente à rede e a aplicação, pois apresenta uma abstração de *socket* TCP padrão para os aplicação.

2.4 Ferramentas IaaS de Código Aberto

São plataformas de código aberto que permitem a entrega dos serviços de infraestrutura pelo provedor. Atuam como um orquestrador, responsável por fornecer os recursos de hardware e software de maneira agregada, automatizando os fluxos de trabalho necessários para fornecer um serviço [28]. Alguns dos orquestradores de código aberto mais importantes da comunidade

de nuvens são: Apache CloudStack, OpenStack e OpenNebula. Esses projetos de código aberto têm uma comunidade crescente e a vantagem de compartilhar conhecimento. Neste trabalho utilizamos os orquestradores CloudStack e OpenNebula, que estão descritos a seguir.

2.4.1 CloudStack

É um software de código aberto para a construção de nuvens públicas e privadas, desenvolvido e suportado por uma enorme comunidade em todo o mundo, bem como apoiado por algumas das principais empresas interessadas na área de computação em nuvem. O principal objetivo do projeto é oferecer a expansão de recursos computacionais por meio de um ambiente capaz de oferecer infraestrutura de TI, gerenciando os recursos computacionais disponíveis. Assim, o CloudStack pode fornecer uma infraestrutura de computação em nuvem de alta disponibilidade [26].

2.4.2 OpenNebula

É uma solução em nuvem que ajuda os *data-centers* virtualizados a provisionar nuvens privadas, públicas e híbridas, sendo uma ferramenta flexível para fornecer tecnologias de armazenamento, rede e virtualização, permitindo a implantação dinâmica de serviços em infraestruturas distribuídas. Foi criado como um projeto de pesquisa pelo Grupo de Pesquisa da *Distributed Systems Architecture* (DSA) em Madri em 2005 e, posteriormente, a primeira versão pública foi lançada em 2008. Ela evoluiu e se tornou completamente de código aberto, sendo frequentemente atualizado pela comunidade. Um dos diferenciais do OpenNebula é que ele garante aos usuários a total interoperabilidade com os componentes de infraestrutura existentes atualmente disponíveis [32][5].

3. Resultados

Nesta Seção são apresentados os resultados obtidos através da pesquisa realizada. Na Seção 3.1, são mostrados os resultados de desempenho das aplicações que utilizaram o gerenciador de nuvem privada OpenNebula e virtualização usando contêineres LXD, para se verificar os benefícios da agregação de placas de redes que culminaram com a publicação [16] na conferencia ISCC 2019 *(IEEE Symposium on Computers and Communications)*. Além disso, na Seção 3.2 são mostrados os resultados de desempenho de diferentes tipos de virtualização, com diferentes níveis de concorrência de recursos causada pela requisição de múltiplos usuários usando aplicações HPC. Estes resultados culminaram na publicação [15] na conferencia HPCS 2018 *(International Conference on High Performance Computing & Simulation)*.

Na Figura 3.1 esta demonstrado a metodologia escalável utilizada nas avaliações. Primeiramente, no nível mais baixo, apresenta-se as interfaces de rede, destacando a possibilidade da a utilização de várias delas ao mesmo tempo, sendo agregadas usando Bonding com o modo *Balanced-RR* no segundo nível. A seguir apresentam-se as *Bridges* Linux, utilizadas para o gerenciador da nuvem definir sua rede padrão e por fim, apresenta-se a possibilidade de utilização dos virtualizadores LXD (evolução do LXC) e KVM para criar N instâncias gerenciados pelo ferramenta IaaS responsável por criar a nuvem privada.

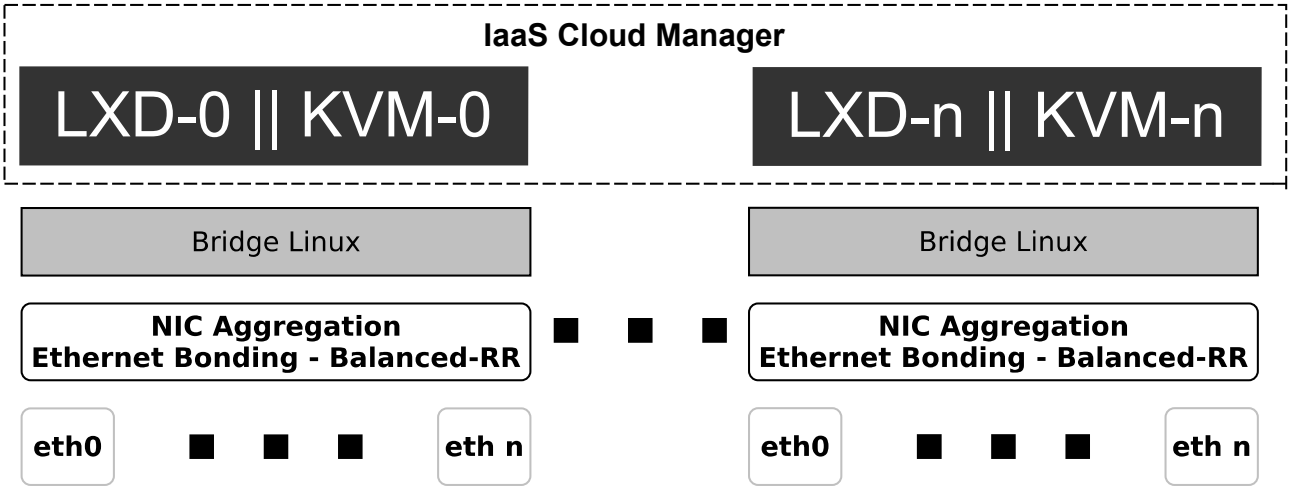


Figura 3.1: Modelo conceitual escalável da proposta das avaliações.

3.1 Minimizando as perdas de desempenho relacionadas a rede em nuvens baseadas em contêiner usando a agregação de links

Considerando a necessidade de otimizar o desempenho da rede para aplicações que tem requisitos de alto desempenho, a agregação de placas de redes (NICs) torna-se uma solução potencial. Desta forma, nesta avaliação foi abordado um dos métodos de agregação de link, conhecido como Bonding, usando contêineres LXD instanciados com o OpenNebula. Para a avaliação de desempenho foram primeiramente avaliadas as capacidades da rede em relação ao *throughput* e latência usando até quatro placas agregação e a seguir, foram usadas aplicações HPC com diferentes demandas de rede.

Primeiro, iniciou-se a configuração de agregação no cenário nativo, configurando a ligação de rede. Essa configuração foi executada nos serviços de rede do Linux, selecionando até quatro interfaces para serem "escravas" de uma interface "virtual" agregada. Em seguida, foi selecionado o modo de operação da agregação, que neste trabalho é o *balanced round-robin* (balance-rr ou modo 0).

Round-robin (RR) é um algoritmo bem conhecido e amplamente adotado em escalonadores de rede, que distribui os pacotes em ordem sequencial, da primeira interface disponível até a última. Esse modelo foi selecionado porque é o único (entre os demais modos de ligação) que pode transformar um único fluxo de conexão TCP/IP para usar mais de uma interface, melhorando potencialmente a vazão.

Como esta implementação foi usada em ambientes virtualizados que geralmente exploram *bridges* Linux para fornecer acesso de rede para instâncias, foi criada uma *bridge* sobre a interface agregada. Finalmente, a tabela de roteamento também foi modificada, tornando a interface *bridge* a padrão para o tráfego de entrada/saída. A mesma configuração foi seguida no segundo *host*, e *hosts* adicionais podem ser facilmente adicionados replicando essa configuração.

Nesta abordagem, foi utilizado o gerenciador de nuvem OpenNebula porque este já havia sido previamente implantado no ambiente e também pois trata-se de um gerenciador de nuvem privada *Open-Source* representativo. Os contêineres LXD foram utilizados por sua sobrecarga de recursos ser menor em comparação com as VMs usando virtualização completa (KVM) e para-virtualização. Os contêineres oferecem virtualização leve no nível do sistema operacional para instâncias, a fim de evitar penalidades de processamento causadas pela abstração de camadas adicionais e emulação de hardware. Assim, o ambiente de nuvem pode alcançar um desempenho quase nativo [11].

O mais recente projeto de código aberto de contêineres Linux é conhecido como LXD [14]. Ele é construído sobre o LXC, fornecendo melhorias e novos recursos com outras funcionalidades para criar e gerenciar contêineres com controle e segurança operacional refinados. O modelo conceitual para criar contêineres é o mesmo empregado pelo LXC, usando *namespaces* e *cgroups* e também *bridges* Linux para a rede. No entanto, o LXD fornece uma API que também pode ser usada pela rede, para o gerenciamento de *hosts* locais ou remotos.

Como o LXD atualmente não possui implementações nativas no OpenNebula, foi utilizado o projeto LXDone[1] para criar e gerenciar *templates* de instâncias LXD usando a versão 3.0.3. Uma instância com este modelo foi implantada em cada nó. Além disso, a nuvem implantada baseada em LXD tinha acesso dedicado aos recursos de hardware.

A velocidade da rede foi avaliada executando o *benchmark* NetPipe entre dois *hosts*, que fornecem os resultados de vazão e latência. O NetPipe *(Network Protocol- Independent Performance Evaluator)* [29] é um *benchmark* de comunicação de rede intensivo, que expõe o desempenho da rede sob uma variedade de condições. Ele escolhe os tamanhos das mensagens em intervalos regulares e com variações para avaliar totalmente a rede. Além disso, as latências são calculadas considerando o tempo de ida e volta (RTT).

Nesta avaliação, foi utilizado o NetPipe 5.1 compilado com o módulo MPI para medir a taxa de transferência e a latência da rede agregada entre os nós. Além disso, as opções `async` que usam "MPI_Irecv" assíncrono para pré-postagem e `Bdir` que enviam dados em ambas as direções (*download* e *upload*) ao mesmo tempo foram usadas. A opção `Bidir` foi usada porque a comunicação unidirecional sobreposta múltipla pode ficar fora de sincronia. No entanto, essa opção também gera a vazão combinada. Como o número de núcleos de cada servidor é 12, o NetPipe foi configurado para usar 24 processos simultaneamente, fixado em 12 processos em cada servidor.

Na Figura 3.2, a latência do TCP é plotada e apresentada em microssegundos em relação ao tamanho da mensagem em kilobytes (kB). Para mostrar resultados mais representativos,

[1] https://github.com/OpenNebula/addon-lxdone

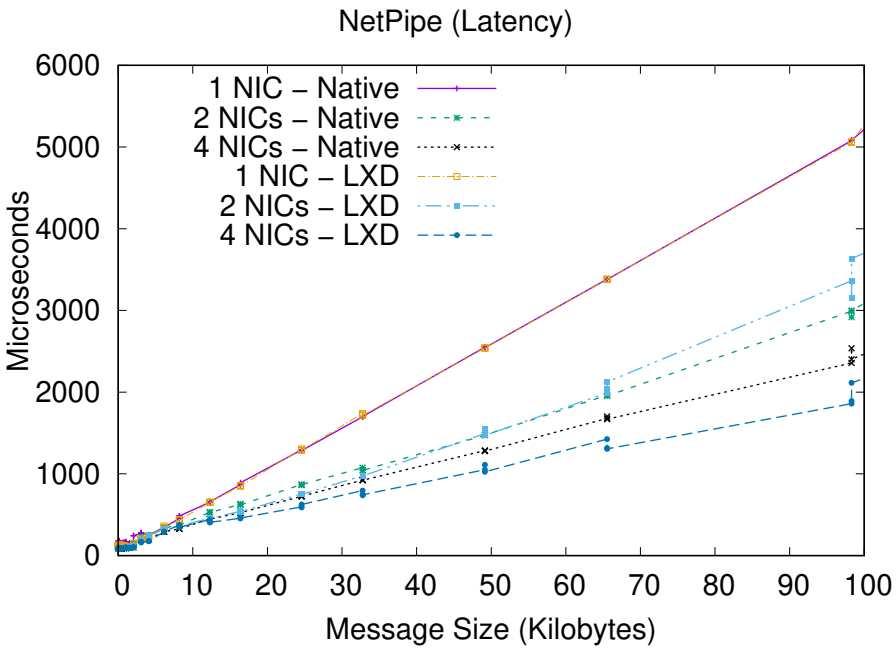


Figura 3.2: Latência TCP com agregação de NICs (Menores valores indicam melhores resultados).

escolhemos o intervalo do tamanho da mensagem aumentando de 1 a 100 kB e plotamos a latência com 2 e 4 NICs agregadas na nuvem nativa e baseada em LXD, respectivamente. Em geral, o desempenho de latência foi significativamente aprimorado no cenário LXD e nativo quando mais NICs são agregados.

Como pode ser visto, no ambiente com 1 NIC, um desempenho semelhante em cenários de nuvem nativos e baseados em LXD foi alcançado. Além disso, com dois NICs, o ambiente nativo e a nuvem baseada em LXD superam a *baseline* (ambiente nativo com uma placa). A nuvem baseada em LXD supera o cenário nativo com tamanhos de mensagens de 10 a 30 kb. Entre 40 e 60 kB, ambos apresentam resultados semelhantes. Com tamanho de mensagem maior que 70 kB, o cenário nativo mostra desempenho aprimorado. Com 4 NICs agregadas, os resultados mostram melhorias de desempenho em comparação com 2 NICs. Esse aspecto é relevante porque demonstra que a latência melhorou à medida que mais NICs foram agregadas. Ao usar 4 NICs, a nuvem baseada em LXD supera o cenário nativo em cerca de 10 kB de tamanho de mensagem. A diferença entre eles aumenta de acordo com os tamanhos das mensagens.

Os resultados de vazão são apresentados em Gigabits por segundo (Gbps) referentes ao tamanho da mensagem em kilobytes (kB) na Figura 3.3 e mostram que o desempenho entre 1 NIC do LXD e o cenário nativo não apresenta diferenças significativas. No entanto, ambos 1 NIC nativo e LXD atingem quase 2 Gbps. Esse desempenho é alcançado porque o NetPipe usou a opção de linha de comando `Bidir`, que gera a taxa de transferência combinada. Esse parâmetro foi usado para representar aplicações reais que fazem uso intensivo de rede, enviando e recebendo dados ao mesmo tempo.

Na avaliação com 2 NICs agregadas usando tamanhos de mensagens entre 10 e 50 kB, o cenário LXD supera significativamente o nativo, atingindo desempenho máximo com tamanho de mensagem de 30 kB e 3,2 Gbps. Depois disso, a nuvem baseada em LXD começa a diminuir o desempenho até que é superada pelo cenário nativo com 65 kB. Em relação à agregação de 4 NICs, quando o tamanho da mensagem atinge cerca de 10 kB, a nuvem baseada em LXD supera o nativo com diferença significativa, atingindo a taxa de transferência máxima de 5 Gbps.

Após os resultados previamente descritos e a consequente validação da abordagem de agre-

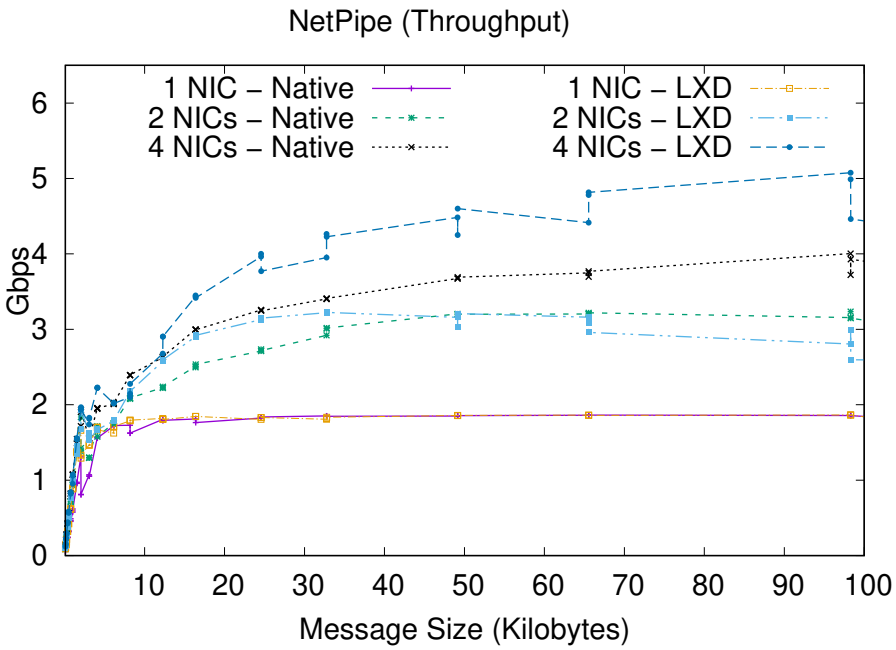


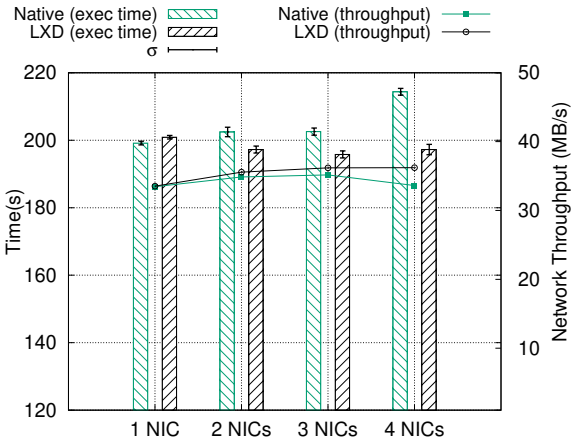
Figura 3.3: Vazão TCP com agregação de NICs (Maiores valores indicam melhores resultados).

gação de NICs, optou-se por conduzir a avaliação usando as aplicações sintéticas do conjunto *Numerical Aerodynamic Simulation Parallel Benchmarks* (NPB) [3] versão 3.3.1 compiladas com o MPI. O NPB foi projetado para auxiliar o *benchmarking* de desempenho de supercomputadores paralelos. Derivados de aplicações de dinâmica computacional de fluidos (CFD), esses *benchmarks* podem representar uma ampla variedade de aplicações HPC com diferentes tipos de requisições de recursos.

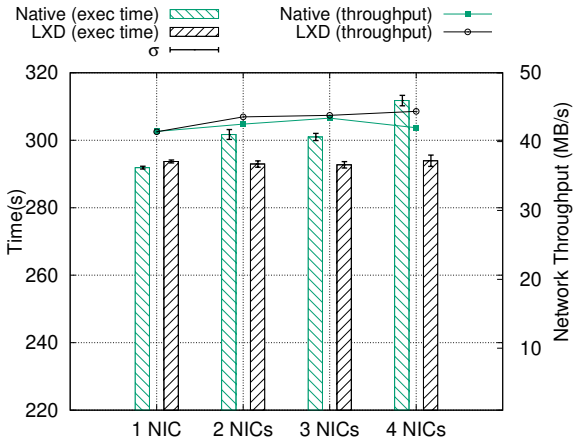
O NPB foi compilado com tamanho de problema classe C e todos os cinco *kernels* (IS, EP, CG, MG e FT) assim como as três pseudo-aplicações (BT, SP e LU) foram utilizados. Além disso, as aplicações usaram 16 processos porque cada um deles requer uma divisão específica de trabalho, por exemplo, o número de processadores deve ser uma raiz quadrada ou até mesmo uma potência de dois. Como os nós têm 12 núcleos, 8 processos foram definidos para serem executados em cada nó para equilibrar a distribuição de carga.

Embora todas as aplicações tenham sido executadas, percebeu-se que em algumas destas, a abordagem de agregação de NICs não resultava em um ganho de desempenho. Desta forma, apenas os resultados representativos das aplicações FT, IS, BT e SP, tanto para o ganho, como para uma pequena perda de desempenho são mostrados. Esses experimentos demonstram se a agregação NIC de rede pode melhorar significativamente o desempenho das aplicações. Aplicações como EP, CG, MG, SP e LU não mostram diferenças significativas no desempenho em relação ao tempo de execução quando foram executados em nossos 4 ambientes (1, 2, 3 e 4 NICs). Os experimentos foram executados 10 vezes em máquinas idênticas dedicadas e apresentaram um desempenho médio com o desvio padrão relacionado. Além disso, para medir o uso da rede, as estatísticas das NICs de rede foram coletadas usando o utilitário `ifstat`.

Na Figura 3.4 são plotados o tempo de execução dos aplicações BT 3.4(a) e SP 3.4(b) nos dois cenários (nativo e nuvem baseada em LXD) com os quatro ambientes (1 NIC como *baseline*, 2, 3 ou 4 NICs usando agregação de NICs). Como pode ser visto, BT e SP no cenário nativo obtiveram piores resultados de desempenho quando a agregação é aplicada. BT e SP são aplicações com execuções sob operações de bloqueio (por exemplo barreiras). Consequentemente, como resultado das características BT e SP, a agregação de rede não foi explorada pela baixa utilização, que é a razão para não melhorar o desempenho mesmo quando

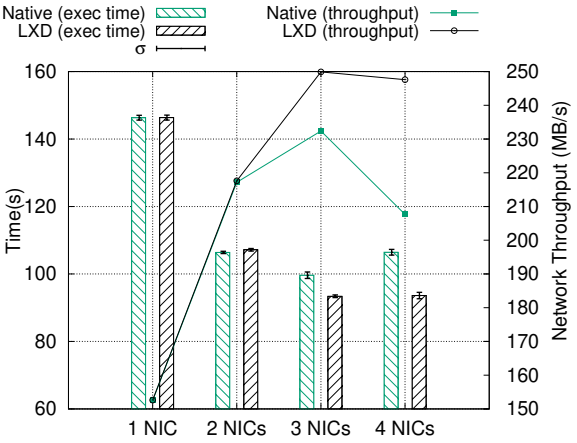


(a) Experimento usando o BT com 16 processos.

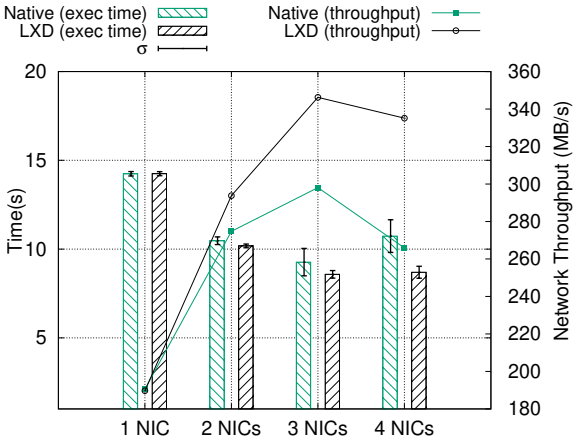


(b) Experimento usando o SP com 16 processos.

Figura 3.4: Tempo de execução das aplicações BT e SP com 16 processos.



(a) Experimento usando o FT com 16 processos.



(b) Experimento usando o IS com 16 processos.

Figura 3.5: Tempo de execução das aplicações FT e IS com 16 processos.

se usam várias NICs agregadas.

Os resultados do *kernel* FT são mostradas na Figura 3.5(a). Esta aplicação realiza operações de redução e comunicação de todos para todos, o que cria um padrão de utilização baseado extremamente na rede [8]. O FT pode melhorar significativamente seu tempo de execução até 36% com 3 NICs agregadas. Esse ganho de desempenho segue as melhorias de vazão com 3 NICs (cerca de 97 MB/s). Além disso, as diferenças de desempenho com NICs 3 e 4 entre cenários nativo e nuvem baseado em LXD são mais significativas. Observando o lado direito da Figura 3.5(a), fica explícito que a vazão desempenha um papel crucial no desempenho dessa aplicação.

A Figura 3.5(b) mostra os experimentos com IS, que é caracterizado por cálculos intensivos durante a maior parte de sua execução, utilizando intensivamente a rede durante operações de redução/acumulativa que reúnem resultados de todos os processos de execução. Em resumo, as execuções de IS usando agregação de rede melhoraram o desempenho ao comunicar e coletar resultados mais rapidamente. Os ganhos de desempenho podem ser vistos principalmente no cenário de nuvem baseado em LXD, em que o tempo de execução foi reduzido 38.95% com 4 NICS agregados.

Uma comparação feita entre os resultados da agregação de NIC nos cenários nativo e de nuvem baseados em LXD enfatiza que o LXD alcançou um maior desempenho. A otimização de

desempenho do LXD (por exemplo o isolamento) permitiu que esse cenário superasse o cenário nativo. A otimização de desempenho de contêineres que executam com aplicações distribuídas/multiprocessos é conhecida na literatura relacionada [34, 10]. O resultado inesperado veio do cenário nativo, que mostrou perdas de desempenho ao usar 3 e 4 NICs agregadas. O motivo do desempenho degradado no cenário nativo está relacionado às combinações das aplicações e a aspectos de baixo nível da implementação da rede.

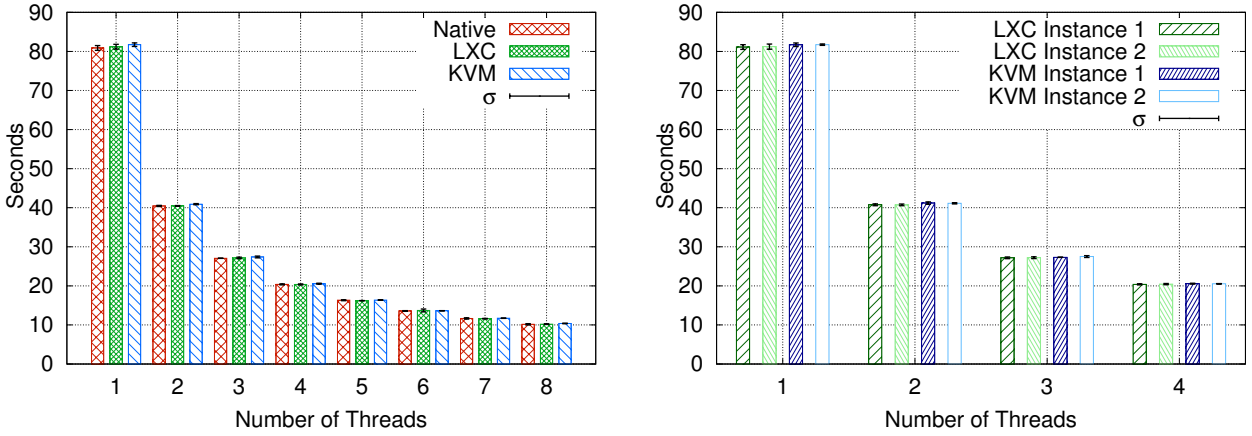
A sobrecarga relacionada à agregação de NIC é devido à retransmissão de pacotes. O modo de agregação balance-rr distribui os pacotes de rede entre as interfaces físicas, o que resultou em pacotes adicionais não ordenados. Consequentemente, o alto número de pacotes desordenados excedeu o limite do *buffer* de congestionamento, então o controle de congestionamento do TCP/IP liberou pacotes desordenados. Portanto, os pacotes liberados precisavam ser retransmitidos, causando lentidão no processamento devido ao atraso de comunicação e, portanto, reduzindo o desempenho da aplicação. A retransmissão também causou tráfego de rede adicional.

3.2 Comparando as virtualizações LXC/KVM em uma nuvem privada usando aplicações científicas

Esta avaliação teve como objetivo a caracterização de desempenho de aplicações que demandam sistemas de alta performance usando múltiplos tipos de virtualização, baseada em *kernel* (KVM) e baseada em sistema operacional (LXC), assim como o ambiente nativo. Além disso foram avaliados o ambiente *single-tenant* no qual apenas um usuário estaria executando a aplicação, e o ambiente *multi-tenant*, no qual múltiplos usuários estariam executando as aplicações, gerando uma concorrência pelos recursos computacionais. Para avaliar o desempenho e a viabilidade ao executar aplicações HPC em cenários de nuvem, escolhemos o NAS Parallel Benchmarks (NPB), um conjunto de aplicações projetadas para auxiliar o *benchmarking* do desempenho de supercomputadores paralelos, compilado na sua versão de programação paralela baseada em OpenMP (memória compartilhada).

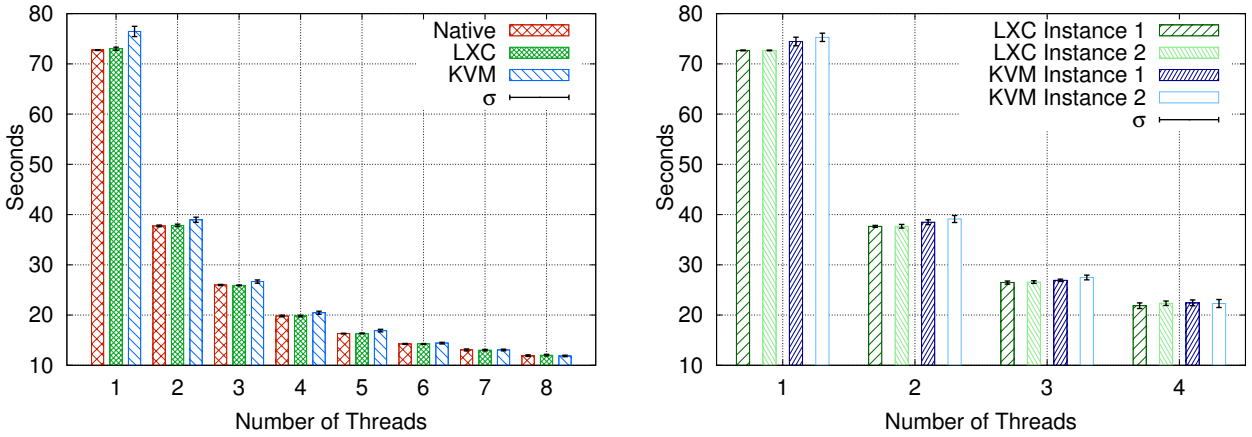
Os resultados obtidos (tempo de execução de três ambientes distintos (Nativo, LXC e KVM) e dois cenários (*single-tenant* e *multi-tenant*) são mostrados e descritos com uma cor específica nos gráficos abaixo. No cenário *single-tenant* (um único usuário), identificamos os ambientes com vermelho para o nativo, verde para o LXC e azul para o KVM. Por outro lado, no cenário *multi-tenant* (dois usuários concorrentes), os ambientes foram identificado com verde-escuro para a instância LXC 1, verde-claro para a instância LXC 2, azul escuro para a instância KVM 1 e azul claro para instância KVM 2, respectivamente. Nessa abordagem, cada *thread* tem um núcleo/CPU virtual disponível e foram considerados instâncias de nuvem com um número variante de CPUs virtuais e memória disponível. Além disso, as aplicações executadas dentro das instâncias usavam um número diferente de *threads*. À primeira vista, podemos observar que o cenário de nuvem baseado em KVM apresentou os piores resultados, indicando que esse tipo de virtualização impõe uma sobrecarga considerável em comparação com os ambientes LXC e nativo.

O *kernel* EP (Figura 3.6) tem suas operações totalmente independentes em cada configuração de número de *thread*. As operações envolvem as gerações de variantes aleatórias gaussianas. Esta aplicação apresenta o contexto real do processamento paralelo. Percebe-se que em cada número crescente de *threads*, o tempo de execução diminuiu consideravelmente. Isso significa que o ganho de desempenho segue em níveis próximos aos recomendados, com pequenas diferenças entre os ambientes. Observamos que nos três ambientes (nuvem baseada em LXC, nuvem baseada em KVM e nativo), tanto para *single-tenant* (Figura 3.6(a)) quanto *multi-tenant* (Figura 3.6(b)), esse *kernel* fornece resultados semelhantes.



(a) *Single-Tenant.* (b) *Multi-Tenant.*

Figura 3.6: Tempos de execução do EP.

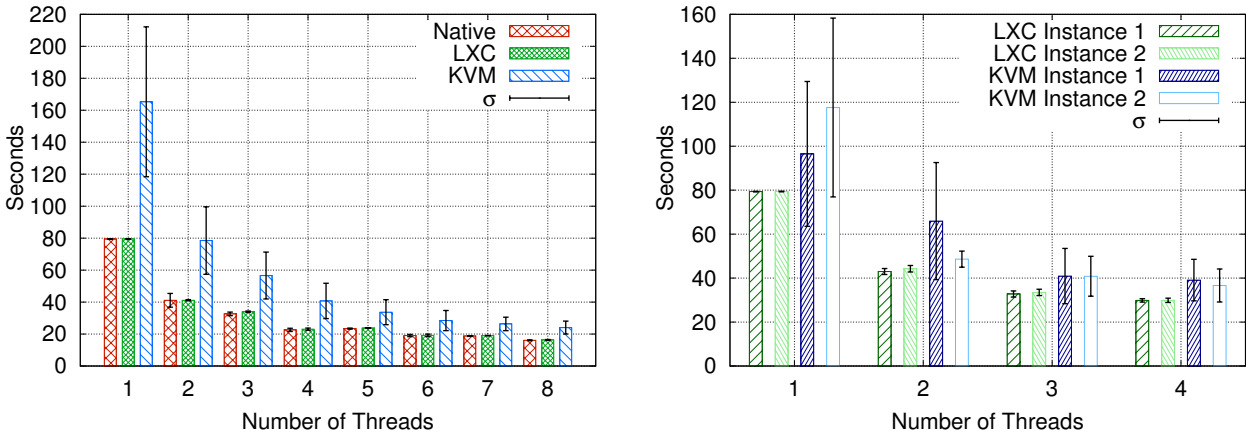


(a) *Single Tenant.* (b) *Multi-Tenant.*

Figura 3.7: Tempos de execução do FT.

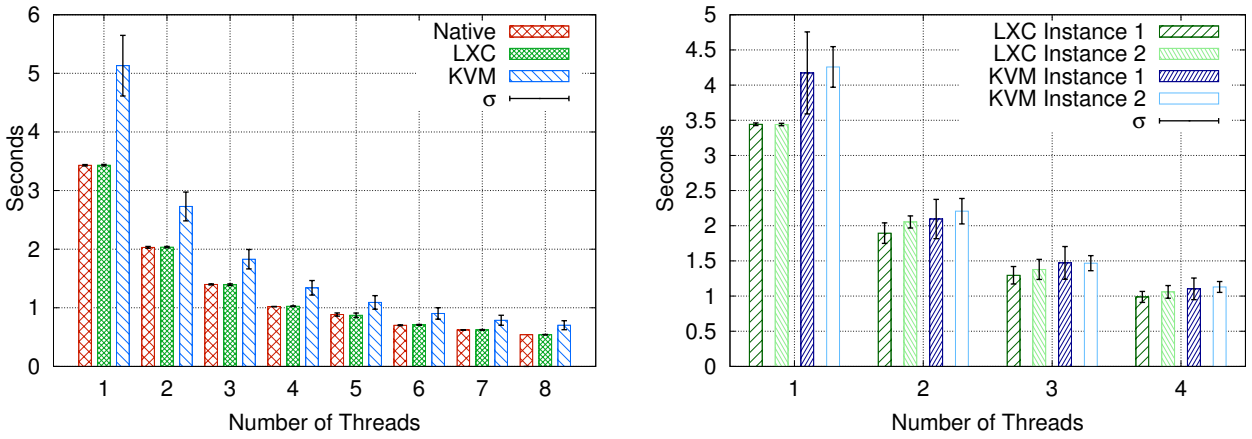
FT (Figura 3.7) é uma aplicação intensiva em memória, focada em "comunicação todos para todos". Ela apresenta algumas perdas de desempenho na primeira *thread* no ambiente virtualizado completo (baseado em KVM) com *single-tenant* (Figura 3.7(a)). Como esta aplicação usa uma grande quantidade de memória, a penalidade de virtualização é mais significativa. Além disso, podemos observar que no ambiente em que há competição por recursos entre os usuários (Figura 3.7(b)), a diferença entre a nuvem baseada em LXC e a nuvem baseada em KVM é reduzida. Esse comportamento é esperado porque o KVM fornece isolamento de recurso.

Da mesma forma, CG (Figura 3.8) e IS (Figura 3.9) apresentaram uma diferença de desempenho entre os cenários de nuvem em *multi-tenant* e *single-tenant*. Na aplicação CG, existem alguns vetores com operações de multiplicação de matrizes, que gera um grande volume de atualizações de memória ao final de cada iteração. Como esperado, a virtualização total teve resultados piores e uma sobrecarga considerável em relação aos contêineres. Além disso, a nuvem baseada em KVM fornece resultados com um desvio padrão significativo. No cenário com 2 usuários (Figura 3.8(b)), pode-se ver resultados diferentes entre estes. Além disso, segundo Regola et al., a penalidade de virtualização é mais significativa em *benchmarks* que exigem grandes quantidades de comunicação ou acesso à memória [24]. Com base nos resultados, também pode-se concluir que a nuvem baseada em LXC (*single-tenant* e *multi-tenants*) apresentou baixa sobrecarga de desempenho quando comparada ao ambiente nativo representado



(a) *Single Tenant.* (b) *Multi-Tenant.*

Figura 3.8: Tempos de execução do CG.



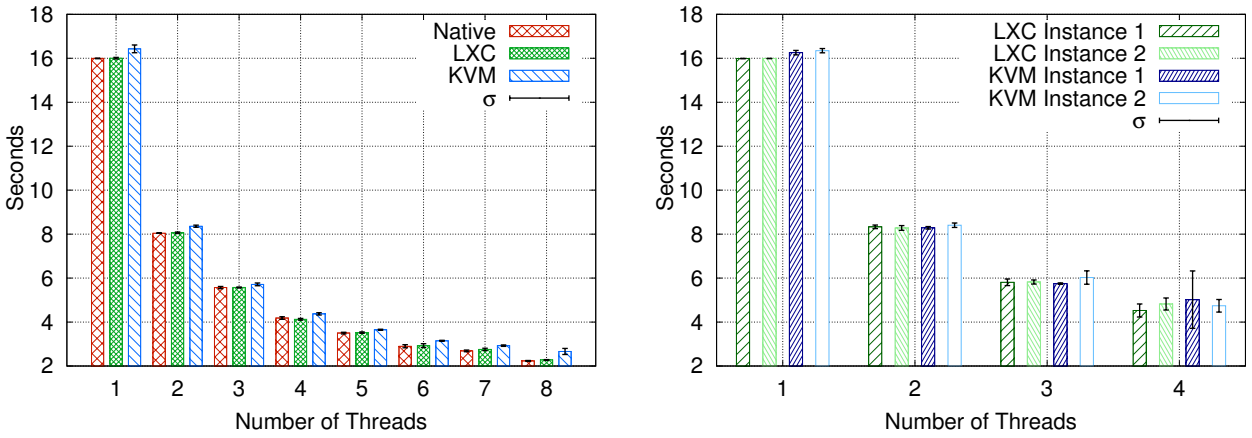
(a) *Single Tenant.* (b) *Multi-Tenant.*

Figura 3.9: Tempos de execução do IS.

em microssegundos.

No IS, o algoritmo *Bucket Sort* executa operações em diferentes posições de memória para classificar um vetor de valores inteiros. Esta aplicação possui algumas características específicas, como o menor conjunto de trabalho e o menor tempo de execução entre todas as aplicações do conjunto de *benchmarks* paralelos NAS OpenMP. Percebe-se que o IS executando no KVM, tanto o *single-tenant* quanto o *multi-tenant* obtiveram um resultado pior mesmo sem usar o paralelismo (com apenas uma *thread*) (Figura 3.9). Além disso, o resultado de desempenho dessa aplicação fornece uma quantidade significativa de desvio padrão na nuvem baseada em KVM. No cenário *multi-tenant* (Figura 3.9(b)), a competição por recursos faz com que os usuários obtivessem um desempenho diferente entre si. O mesmo comportamento do IS já foi relatado por Strazdins, et al. [31]. Além disso, podemos observar que a nuvem baseada em LXC tem um desempenho quase nativo com único usuário. No entanto, com 2 usuários (Figura 3.9(b)), a diferença entre a nuvem baseada em LXC e a nuvem baseada em KVM diminui à medida que o número de *threads* aumenta. Portanto, tendo em conta o desvio padrão, ambos têm resultados semelhantes com 4 *threads*.

A aplicação de MG fornece uma solução para uma equação de Poisson discreta tridimensional usando a abordagem Multi Grid. O impacto do *overhead* no KVM é menor em relação a outros métodos, como pode ser visualizado na Figura 3.10. Embora o MG seja um *kernel*



(a) *Single Tenant.* (b) *Multi-Tenant.*

Figura 3.10: Tempos de execução do MG.

intensivo de memória, o acesso à esta não parece impactar significativamente no desempenho. Com um único usuário (Figura 3.10(a)) a nuvem baseada em LXC teve resultados próximos ao ambiente nativo. Da mesma forma, no cenário com dois usuários (Figura 3.10(b)), as nuvens baseadas em LXC e KVM forneceram resultados semelhantes.

OS experimentos levaram a uma investigação mais profunda sobre as tecnologias que podem afetar os resultados obtidos. Foi identificado que o *Kernel Samepage Mergin* (KSM) aumenta a densidade de memória gerando páginas compartilhadas ao mesclar as páginas iguais. Consequentemente, haverá mais memória disponível para executar mais VMs ou aplicações no mesmo sistema [1]. Além disso, tanto a nuvem baseada em KVM quanto a baseada em LXC podem se beneficiar principalmente o ambiente *multi-tenant* no qual a memória disponível é dividida por dois usuários. Como esses usuários estão executando a mesma aplicação, a mesclagem do KSM é igual à requisição de páginas. Particularmente, esta tecnologia impacta diretamente no cenário *multi-tenant* das aplicações EP e MG, que usa a memória cache eficientemente e apresenta resultados próximos entre o LXC e nuvens baseadas no KVM.

Além disso, o suporte à virtualização Intel VT-x em nossas máquinas também pode afetar o desempenho. Neiger et al. enfatiza que essa tecnologia faz com que o KVM execute instruções com acesso nativo aos núcleos da CPU [21]. Essa tecnologia impacta sobre as mesmas aplicações que fazem uso eficiente da memória cache. As características da aplicação combinadas com a configuração experimental de nossos testes resultaram em um desempenho otimizado.

4. Conclusão

Considerando o primeiro trabalho que buscou otimizar aspectos relacionados a comunicação (rede), percebe-se que a comunicação pode causar *overheads* para aplicações de alto desempenho executadas em ambientes de nuvem. A realização do trabalho buscou atenuar esse problema propondo uma abordagem usando a agregação de NICs. Os resultados mostraram que a agregação da NICs integrada na nuvem melhorou significativamente o desempenho da rede, fornecendo maior vazão e reduzindo a latência. A agregação de NICs mostrou o potencial de ser facilmente integrada a outras tecnologias de virtualização que usam "bridges" de rede.

Em aplicações HPC, as melhorias no desempenho e pequenas perdas foram perceptíveis. Por exemplo, as aplicações IS e FT fazem uso intensivo da rede, o que permitiu melhorar o desempenho até 36% (FT com 3 NICs) e 38.95% (IS com 4 NICs) devido à comunicação mais rápida à medida que mais NICs foram agregadas. Por outro lado, as aplicações BT e SP apresentam piores resultados quando executados no cenário nativo. No entanto, o mesmo desempenho pior não é mostrado ao executar na nuvem baseada em LXD.

Os resultados obtidos mostram que, em cenários em que a infraestrutura de rede é escalonável em termos de NICs, cabos e portas de *switch*, implementações diretas com uma integração pronta para uso em um ambiente de nuvem podem fornecer melhorias significativas no desempenho. Embora os experimentos tenham sido executados apenas em dois nós computacionais, espera-se que as tendências de desempenho sejam semelhantes em ambientes de grande escala, especialmente ao executar aplicações que usam a rede intensivamente para explorar a otimização fornecida pela agregação de NICs. Argumenta-se que as otimizações de rede são relevantes para melhorar o desempenho de ambientes de computação intensiva em nuvem.

Como experiência, também utilizamos a metodologia MPTCP. Entretanto, após várias tentativas falhas de realizar a agregação e testes de desempenho, foi definido que esta não seria utilizada nos experimentos. Embora sua descrição objetiva a possibilidade da agregação, é provável que o tempo dispendido e a complexidade para sua utilização torne esta metodologia de difícil acesso. Em trabalhos futuros, plane-se: (I) otimizar o desempenho do algoritmo *Round-robin* para reduzir o congestionamento do TCP; (II) avaliar a interferência da rede entre instâncias de vários usuários simultâneos (concorrência por recursos); (III) avaliar os outros modos de agregação de links que o Bonding possuí, uma vez que neste trabalho apenas o modo 0 (*Balance-rr*) foi utilizado.

O segundo trabalho apresentou uma análise e avaliação de desempenho, utilizando uma abordagem estatística para identificar se os resultados são significativamente diferentes, abrangendo experimentos realizados em dois cenários (*single-tenant* e *multi-tenants*) em condições de nuvem privada implantadas com a plataforma CloudStack. Os ambientes suportaram as tecnologias de virtualização KVM e LXC, nas quais o conjunto de *benchmarks* paralelos NAS OpenMP foram experimentados. Percebeu-se que o LXC fornece a menor sobrecarga para esse tipo de aplicações. No primeiro cenário, ao comparar os tipos de instância, a virtualizações baseadas em contêiner supera a virtualização baseada em kernel em 90% sob condições de nuvem privada com uma plataforma CloudStack para nossas amostras. Apenas para um caso excepcional (FT com oito *threads*) essa instância baseada em KVM supera o LXC com a mínima diferença. Além disso, a nuvem baseada em LXC supera o KVM em 57,5% dos resultados do cenário de *multi-tenancy*.

Também descobriu-se que o alto uso de memória dessas aplicações afeta significativamente o desempenho de instâncias baseadas em KVM. Isso porque a virtualização completa adiciona mais instruções que precisam ser gerenciadas pela CPU. Consequentemente, este precisa tratar mais informações e bufferização que causam degradação de desempenho em comparação com o desempenho do ambiente nativo. Além disso, o KSM e o Intel VT-x são tecnologias permissivas

para melhor desempenho, o que pode afetar significativamente os aplicações que possuem uso eficiente da memória cache.

Finalmente, a nuvem baseada em LXC é a melhor opção para aplicações científicas com único usuário em nossos testes. Apesar de a nuvem baseada em LXC também ter obtido melhores resultados com dois usuários, o número de resultados que não há diferenças significativas entre os ambientes de nuvem é de aproximadamente 45%. Embora a nuvem baseada em LXC de *multi-tenants* seja a melhor opção, a nuvem baseada em KVM ainda pode ser considerada uma opção adequada. O aumento de desempenho da nuvem baseada em KVM com dois usuários é resultante ao isolamento de recursos e usuários. Por outro lado, o LXC tem os melhores resultados com um único usuário e um pior isolamento de recursos.

No futuro, para avançar a avaliação deste trabalho, planeja-se avaliar aplicações de diferentes domínios (ex. Processamento de *Stream*, *Big Data*, *Deep Learning* e *Machine Learning*) para descobrir comportamentos diferentes; incluir diferentes ferramentas de gerenciamento de IaaS (ex. OpenStack e OpenNebula); realizar experimentos com super-provisão de recursos ou mesmo multilocação, práticas amplamente usadas na nuvem; e realizar os experimentos com outras tecnologias de virtualização (ex. HyperV, VMWare, Xen e Docker).

Referências Bibliográficas

[1] Arcangeli, A., Eidus, I., Wright, C.: Increasing Memory Density by Using KSM. In: Proceedings of the Linux Symposium (OLS) (2009)

[2] Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al.: A View of Cloud Computing. Communications of the ACM (2010)

[3] Bailey, D.H., Barszcz, E., Barton, J.T., Browning, D.S., Carter, R.L., Dagum, L., Fatoohi, R.A., Frederickson, P.O., Lasinski, T.A., Schreiber, R.S., et al.: The NAS Parallel Benchmarks. International Journal of High Performance Computing Applications (IJHPCA) (1991)

[4] Buyya, R., Broberg, J., Goscinski, A.: Cloud Computing: Principles and Paradigms. Wiley (2010)

[5] Chandrasekaran, K.: Essentials of Cloud Computing. Chapman and Hall/CRC (2014)

[6] Chaufournier, L., Ali-Eldin, A., Sharma, P., Shenoy, P., Towsley, D.: Performance Evaluation of Multi-Path TCP for Data Center and Cloud Workloads. In: ACM/SPEC International Conference on Performance Engineering (ICPE) (2019)

[7] Dua, R., Raja, A.R., Kakadia, D.: Virtualization vs Containerization to Support PaaS. In: IEEE International Conference on Cloud Engineering (IC2E) (2014)

[8] Faraj, A., Yuan, X.: Communication Characteristics in the NAS Parallel Benchmarks. In: International Conference on Parallel and Distributed Computing and Systems (PDCS) (2002)

[9] Fundation, T.L.: Bonding (2019), https://wiki.linuxfoundation.org/networking/bonding, Último acesso em dezembro de 2019

[10] Griebler, D., Vogel, A., Maron, C.A.F., Maliszewski, A.M., Schepke, C., Fernandes, L.G.: Performance of Data Mining, Media, and Financial Applications under Private Cloud Conditions. In: IEEE Symposium on Computers and Communications (ISCC) (2018)

[11] Jawarneh, I.M.A., Bellavista, P., Foschini, L., Martuscelli, G., Montanari, R., Palopoli, A., Bosi, F.: QoS and Performance Metrics for Container-based Virtualization in Cloud Environments. In: International Conference on Distributed Computing and Networking (ICDCN) (2019)

[12] KVM: Kernel Virtual Machine (2019), http://www.linux-kvm.org, Último acesso em dezembro de 2019

[13] LXC: Linux Containers (LXC) (2019), http://linuxcontainers.org/, Último acesso em dezembro de 2019

[14] LXD: Linux Containers (2019), https://linuxcontainers.org/lxd/introduction/, Último acesso em dezembro de 2019

[15] Maliszewski, A.M., Griebler, D., Schepke, C., Ditter, A., Fey, D., Fernandes, L.G.: The NAS Benchmark Kernels for Single and Multi-Tenant Cloud Instances with LXC/KVM. In: International Conference on High Performance Computing & Simulation (HPCS) (2018)

[16] Maliszewski, A.M., Vogel, A., Griebler, D., Roloff, E., Fernandes, L.G., Navaux, P.O.A.: Minimizing Communication Overheads in Container-based Clouds for HPC Applications. In: Symposium on Computers and Communications (ISCC) (2019)

[17] Maron, C.A.F., Vogel, A., Griebler, D.: Caracterizando a Implantação e o Desempenho de Aplicações em Ambientes de Nuvem Privada com Recursos Compartilhados e Dedicados. Tech. rep., Laboratory of Advanced Research on Cloud Computing (LARCC) (2018)

[18] Maron, C.A.F., Griebler, D.: Avaliando o Desempenho das Ferramentas de Nuvem Privada OpenStack e OpenNebula. Tech. rep., Laboratory of Advanced Researches on Cloud Computing (LARCC) (2015)

[19] Maron, C.A.F., Vogel, A., Griebler, D.: Caracterizando o Desempenho de Rede e Aplicações Pipeline em Ambientes de Nuvem Privada. Tech. rep., Laboratory of Advanced Researches on Cloud Computing (LARCC) (2017)

[20] Mell, P., Grance, T., et al.: The NIST Definition of Cloud Computing. Tech. rep., Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD, United States (2011)

[21] Neiger, G., Santoni, A., Leung, F., Rodgers, D., Uhlig, R.: Intel Virtualization Technology: Hardware Support for Efficient Processor Virtualization. Intel Technology Journal (2006)

[22] Portnoy, M.: Virtualization Essentials. Wiley (2016)

[23] RedHat: O que é KVM? (2019), https://www.redhat.com/pt-br/topics/virtualization/what-is-KVM, Último acesso em dezembro de 2019

[24] Regola, N., Ducom, J.: Recommendations for Virtualization Technologies in High Performance Computing. In: IEEE International Conference on Cloud Computing Technology and Science (CloudCom) (2010)

[25] Roloff, E., Diener, M., Gaspary, L.P., Navaux, P.O.A.: HPC Application Performance and Cost Efficiency in the Cloud. In: Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP) (2017)

[26] Sabharwal, N.: Apache CloudStack Cloud Computing. Community experience distilled, Packt Publishing (2013)

[27] Scharf, A.L.B.L., Ford, C.: Multipath TCP (MPTCP) Application Interface Considerations (2006), https://www.rfc-editor.org/rfc/rfc6897.txt, Último acesso em dezembro de 2019

[28] Shrivastwa, A., Sarat, S.: Learning OpenStack. Packt Publishing (2015)

[29] Snell, Q.O., Mikler, A.R., Gustafson, J.L.: Netpipe: A Network Protocol Independent Performance Evaluator. In: International Conference on Intelligent Information Management and Systems (IASTED) (1996)

[30] Sotomayor, B., Montero, R.S., Llorente, I.M., Foster, I.: Virtual Infrastructure Management in Private and Hybrid Clouds. Internet computing, IEEE (2009)

[31] Strazdins, P.E., Cai, J., Atif, M., Antony, J.: Scientific Application Performance on HPC, Private and Public Cloud Resources: A Case Study Using Climate, Cardiac Model Codes and the NPB Benchmark Suite. In: International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW) (2012)

[32] Toraldo, G.: OpenNebula 3: Cloud Computing. Packt, Birmingham (2013)

[33] Vogel, A., Griebler, D.: Implantando, Avaliando e Analisando as Ferramentas para Gerenciamento de IaaS OpenStack, OpenNebula e CloudStack. Tech. rep., Laboratory of Advanced Researches on Cloud Computing (LARCC) (2016)

[34] Vogel, A., Griebler, D., Schepke, C., Fernandes, L.G.: An Intra-Cloud Networking Performance Evaluation on CloudStack Environment. In: Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP) (2017)

[35] Xavier, M.G., Neves, M.V., Rossi, F.D., Ferreto, T.C., Lange, T., De Rose, C.A.: Performance Evaluation of Container-based Virtualization for High Performance Computing Environments. In: Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP) (2013)