Caracterizando a Implantação e o Desempenho de Aplicações em Ambientes de Nuvem Privada com Recursos Compartilhados e Dedicados

Carlos A. F. Maron, Adriano Vogel, Dalvan Griebler

Três de Maio, Brasil

HiPerfCloud:

High Performance in Cloud

Avaliação de Ambientes de Nuvem IaaS

RT4: Relatório Técnico de Pesquisa (Atividades de 2017)

Reference: 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*. Laboratory of Advanced Research on Cloud Computing (LARCC), Technical Report, 2018.

ID do Documento:	LARCC-HiPerfCloud-RT4
Versão:	1
Autores:	Adriano Vogel, Carlos A. F. Maron, Dalvan Griebler
Objetivo:	Avaliar o Desempenho de diferentes implantações de nuvem, diferentes cargas de trabalho e diferentes modos de utilização (dedicado, compartilhado, superprovisionado)
Tarefa:	Caracterizar o desempenho de aplicações de mineração de dados, financeiras e multimídia em ambientes de nuvens. Implantar e avaliar o desempenho de uma nuvem privada OpenStack
Hardware:	2 *clusters* isolados foram criados usando 4 máquinas idênticas, um cluster de 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). No segundo cluster cada máquina possuí a configuração: 32GB Memória, 2 Xeon E5410 (quad-core 2.33 GHz), armazenamento de 500GB de armazenamento e conectados em uma rede Gigabit (10/1000).
Ambiente:	Sistema Operacional (Ubuntu Server 16.04), Virtualizador (KVM e LXC), OpenStack (vers. Kilo), CloudStack (vers. 4.8). Aplicações Paralelas usando memória compartilhada: PARSEC, LINPACK, STREAM e UPerf.
Softwares:	GNUPlot (Gráficos), Latex (Documentos).

Tarefa	Responsável	Instituição	Papel	Data
Criado por:	Dalvan Griebler	SETREM/PUCRS	Coordenador	12/12/2016
Editado por:	Adriano Vogel	SETREM/PUCRS	Pesquisador	22/11/2018
Editado por:	Carlos A. F. Maron	SETREM/PUCRS	Pesquisador	15/11/2018
Revisão de Conteúdo:	Dalvan Griebler	SETREM/PUCRS	Coordenador	20/11/2018
Revisão:	Adriano Vogel	SETREM/PUCRS	Pesquisador	03/12/2018
Revisão:	Carlos A. F. Maron	SETREM/PUCRS	Pesquisador	03/12/2018
Aprovado por:	Dalvan Griebler Ildo Corso	SETREM/PUCRS SETREM/ABASE	Coordenador Colaborador	 08/12/2018

Log de Mudanças do Documento

Versão	Autores	Instituição	Mudança	Data
1	Dalvan Griebler	SETREM/PUCRS	Versão inicial	12/12/2016
1	Adriano Vogel e Carlos Maron	SETREM/PUCRS	Versão parcial	31/10/2018
1	Dalvan Griebler	SETREM/PUCRS	Revisão, comentários e mudanças	20/11/2018
1	Adriano Vogel e Carlos Maron	SETREM/PUCRS	Correções e versão para revisão externa	23/11/2018
1	Adriano Vogel e Carlos Maron	SETREM/PUCRS	Correções e versão para revisão externa	29/11/2018
1	Adriano Vogel e Carlos Maron	SETREM/PUCRS	Correções e versão para revisão externa	03/12/2018
1	Adriano Vogel, Carlos Maron, Dalvan Griebler	SETREM/PUCRS	Versão final	03/12/2018

Lista de colaboradores internos e externos

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

- Adriano Vogel (SETREM/PUCRS)
- Carlos A. F. Maron (SETREM/PUCRS)
- Claudio Schepke (UNIPAMPA)
- 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 características de implantação e gerenciamento nas ferramentas disponíveis. Este documento apresenta a continuidade do RT1-2015 [12], RT2-2016 [25] e RT3-2017 [13] e mostra novos resultados em implantação de nuvem privada com OpenStack e a avaliação e caracterização de desempenho do ambiente de nuvem e aplicações representativas.

Contexto do Relatório

Este documento é o quarto Relatório Técnico *Caracterizando a Implantação e o Desempenho de Aplicações em Ambientes de Nuvem Privada com Cenários Compartilhados e Dedicados* relativo ao **Projeto HiPerfCloud** que apresenta resultados de infraestrutura onde são avaliados a implantação e o desempenho de aplicações em cenários compartilhados e dedicados.

Estrutura do Relatório

Este documento inicialmente apresenta a estrutura geral. Posteriormente, o contexto da computação em nuvem é contextualizado com as ferramentas de gerenciamento OpenStack e CloudStack. Também são contextualizados as aplicações representativas de um cenário real. Ao final, os experimentos e resultados são discutidos e a conclusão é apresentada.

Sumário

1. Introdução

Este capítulo apresenta um olhar genérico e introdutório do que será discutido nesse documento, elencando o conteúdo dos capítulos e termos relevantes desse estudo.

1.1 Visão Geral

Neste documento são apresentados os resultados de desempenho de nuvens OpenStack e CloudStack utilizando diferentes ferramentas de virtualização. Também são apresentados resultados de desempenho de aplicações de mineração de dados, financeira e multimídias em nuvens privadas CloudStack.

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.
- **OpenStack**: Ferramenta de código aberto para gerenciamento de infraestrutura de nuvem IaaS.
- **CloudStack**: Ferramenta de código aberto para gerenciamento de infraestrutura de nuvem IaaS.
- **PARSEC**: Suíte de *benchmarks* que representam aplicações reais e emergentes.
- **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.

1.3 Estrutura deste Documento

Este documento está organizado em cinco capítulos:

- Capítulo 1: Apresenta um visão geral deste documento.
- Capítulo 2: Nesta seção, encontra-se o referencial sobre a computação em nuvem e seus serviços, seguido pela apresentação de ferramentas e dos métodos usados para avaliar o desempenho em ambientes implantados.
- 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 e trabalhos futuros.

2. Contexto

Este relatório técnico consiste em realizar novos estudos envolvendo a computação em nuvem. Dessa forma, são apresentados resultados sobre dois aspectos importantes para uma infraestrutura de nuvem, que são a implantação e o desempenho de aplicações na nuvem. Uma nuvem privada possuí inúmeras formas de implantação. Na Seção 3.2 é discutido e apresentado a implantação de uma nuvem OpenStack. Outro aspecto é o desempenho das aplicações em nuvem. Por isso, na Seção 2.3 são apresentados os resultados da caracterização de desempenho de aplicações de mineração de dados, financeiras e multimídias.

As ferramentas utilizadas neste relatório estão descritas detalhadamente nas pesquisas anteriores e podem ser encontradas nas versões anteriores do Relatório Técnico [12, 25, 13].

Este relatório técnico é um complemento de outras pesquisas que foram publicadas em eventos internacionais.

2.1 Plataformas de Nuvem Privada e Tecnologias de Virtualização

Neste estudo foram utilizadas as ferramentas de gerenciamento de infraestrutura de nuvem OpenStack e CloudStack. **OpenStack**[1] é uma ferramenta robusta com código aberto, desenvolvida em *Python* para criação de ambientes em nuvem. É composto por um núcleo de tecnologias e APIs que fornecem recursos (*e.g.* processamento, armazenamento e rede) para aplicações através de um *datacenter* [17]. É uma solução modular e distribuída, que necessita de uma conexão de rede eficiente para operar.

Apache CloudStack[2] é um software de código aberto flexível e desenvolvido em Java. Ele permite a utilização de vários recursos para implantar ambientes de nuvem [5]. A arquitetura permite implementações de redes virtuais com suporte a recursos avançados, como VLANs *(Virtual LANs)*, VPN *(Virtual Private Network)*, e GRE *(Generic Routing Encapsulation)* [21]. Na pesquisa onde são avaliadas as aplicações de mineração, financeiras e multimídia, são considerados aspectos de desempenho em nuvem privada composta por ferramentas de código aberto [26] Apache CloudStack para gerenciar a infraestrutura de nuvem [22]. Pois, esta ferramenta oferece gerenciamento de recursos computacionais e acesso destes recursos como forma de serviço. As instâncias da nuvem foram implantadas com virtualização baseada em *hipervisor* (KVM) e virtualização em nível de SO (LXC). As diferenças entre o KVM e os contêineres LXC é que no LXC os processos dos usuários são executados diretamente no sistema operacional do *host*, enquanto o KVM fornece hardware virtualizado no qual o sistema operacional convidado está instalado.

No KVM, os dispositivos de hardware podem ser virtualizados com o emulador de dispositivo QEMU. O KVM usa um módulo do *kernel* para interceptar solicitações de E/S do Linux e transfere para o QEMU, que traduz essas solicitações em chamadas do sistema. O KVM também suporta *drivers* de para-virtualização, que geralmente alcançam melhor desempenho. As instâncias com KVM recebem um ambiente virtual e isolado.

Diversos recursos são encontrados no KVM, como *Qemu Monitor Protocol* (QMP), *KVM Paravirtual Clock*, *Balloon Memory Driver*, *Kernel Samepage Merging* (KSM), entre outros. Dentre estes, KSM se destacou durante as pesquisas e influenciou positivamente os resultados de desempenho. O KSM [3] é um recurso utilizado pelo KVM para aplicar uma otimização de

[1] https://www.openstack.org/
[2] https://cloudstack.apache.org/
[3] Documentação KSM: https://www.kernel.org/doc/Documentation/vm/ksm.txt

memória no ambiente, compartilhando entre as máquinas virtuais dados idênticos alocados em memória. Esse compartilhamento evita a duplicação de dados em memória, melhora o desempenho de determinadas aplicações (ex: *cache-miss*, alocação de memória, acesso à memória, *etc*) e a utilização de recursos.

No LXC, os processos dos usuários são executados diretamente no *kernel* do Linux sem uma camada de virtualização e compartilham caches e outras estruturas de dados no nível do sistema operacional. Buscando melhorias de desempenho, o LXC oferece menos isolamento entre os contêineres e as operações dentro de um contêiner podem afetar o desempenho de outros contêineres por meio de estruturas de dados compartilhadas. Os recursos são controlados por um mecanismo chamado *cgroup*, que determina os limites dos recursos para cada contêiner.

2.2 CloudStack e Alta Disponibilidade

Um aspecto importante em ambientes de nuvem é a alta disponibilidade de aplicações executadas na infraestrutura. Diversas aplicações não podem sofrer com interrupções ou falhas enquanto executadas. Por isso, técnicas de redundância e tolerância a falhas são implantadas no nível da infraestrutura e nas aplicações. Nas infraestruturas de nuvem, existe uma crescente demanda para manter online servidores e máquinas virtuais, porém, nas ferramentas *open source* de IaaS é notável os desafios para alta disponibilidade dos ambientes.

Um minicurso foi ministrado com o objetivo de introduzir o tema computação em nuvem no modelo IaaS usando a ferramenta de gerenciamento de código aberto CloudStack. O minicurso teve enfoque prático ao implantar uma nuvem privada usando CloudStack e evidenciar as funcionalidades presentes na ferramenta para alta disponibilidade em ambientes de nuvem. Mais detalhes podem ser encontrados no RT3-2016 [13].

2.3 Desempenho de Aplicações de Mineração de Dados, Multimídia e Financeira em Nuvens Privadas

A computação em nuvem é um paradigma capaz de fornecer Infraestrutura como Serviço (IaaS), no qual usuários/clientes podem realizar o provisionamento de recursos computacionais sob demanda (CPU, memória, armazenamento), seja para modelos privados ou públicos de nuvens [26]. IaaS é o modelo de serviço base para um ambiente de computação em nuvem, pois suporta os modelos de serviços dos níveis superiores, tais como Plataforma como Serviço (PaaS) e Software como Serviço (SaaS) [15]. No nível mais baixo de uma nuvem, existe a camada de virtualização que permite o dinamismo para os modelos de serviços. Portanto, os usuários da nuvem tem um provisionamento flexível de recursos e pagam apenas pelo uso [4].

Diversas aplicações estão migrando para as nuvem devido as características destas infraestruturas. As aplicações representam diversos domínios, como aplicações corporativas e de alto desempenho. No entanto, é difícil prever o desempenho da maioria das plataformas de nuvens existentes, devido as variações de desempenho e flutuações de carga causadas pelo ambiente que é compartilhado com outros usuários e pelas pilhas de softwares (virtualização e *drivers*)[2, 27, 20]. Além disso, a literatura carece de investigações mais profundas e empíricas sobre como é a melhor forma de explorar e otimizar os benefícios das nuvens.

Analisar e estudar o desempenho das aplicações em diferentes implantações de nuvem é fundamental para fornecer *insights* para provedores de nuvem e usuários em potencial. Para isto, foi escolhido avaliar o desempenho de um conjunto de *benchmarks* da suíte PARSEC sob condições de nuvem privada. Estes *benchmarks* representam aplicações reais, emergentes, e são implementadas com diferentes modelos de paralelismo. Como os *benchmarks* desta suíte

foram selecionados através de uma metodologia específica [18], muitas outras aplicações com características semelhantes podem seguir as tendências de desempenho dos experimentos realizados neste trabalho. Desse modo, usuários e provedores de nuvem podem se beneficiar dos resultados para identificar a melhor maneira de implantar uma nuvem com base na aplicação. Por exemplo, o provedor pode solicitar aos usuários as características de suas aplicações e, em seguida, oferecer implantações personalizadas para garantir um desempenho otimizado.

Trabalhos anteriores sobre desempenho em ambientes de nuvem analisaram os benefícios da migração das aplicações corporativas para provedores de nuvem pública [10]. Outros caracterizaram o desempenho da nuvem por meio de avaliações de desempenho [9, 7]. Além disso, estudos avaliaram a viabilidade de infraestruturas de nuvem para execução de aplicações científicas [24]. Tradicionalmente, estes trabalhos usam os *benchmarks* da suíte NAS [11, 14]. Embora sejam extremamente importantes para a área de computação de alto desempenho (HPC), os resultados com estes *benchmarks* não são representativos para aplicações corporativas.

2.4 Suíte PARSEC

Foi escolhido a suíte PARSEC [18] para avaliar o desempenho e a viabilidade ao executar aplicações reais em ambientes de nuvem. A suíte é composta por 13 *benchmarks* diferentes (10 aplicações e 3 *kernels*) que simulam o comportamento de aplicações reais, tais como: visão computacional, codificação de vídeo, análise financeira, animação física, processamento de imagens, compressão e deduplicação de arquivos.

Embora a suíte PARSEC tenha sido desenvolvida para avaliar o desempenho dos processadores modernos, seu conjunto de *benchmarks* pode ser utilizado como uma carga de trabalho representativa das aplicações reais em ambientes de nuvem. Os *benchmarks* escolhidos incluem o domínio Financeiro devido à sua relevância para ambientes corporativos que exigem recursos e disponibilidade computacionais. Também foi considerado o domínio de mineração de dados devido à alta demanda atual por essa classe de aplicações, tanto em pesquisa quanto na indústria. Além disso, o domínio de processamento de mídia também foi incluído, porque esta área tem se destacado e seu uso exige recursos computacionais especializados para processamento de imagem e vídeo. Os *benchmarks* escolhidos são descritos em seguida.

Blackscholes é uma aplicação de análise financeira que representa o campo Equações Diferenciais Parciais (PDE). Esta aplicação é usada em cálculos financeiros e avalia a quantidade de pontos flutuantes que um processador pode executar. O conjunto de entrada *native* foi utilizado nos testes, que tem 10.000.000 valores de referência utilizado para os cálculos financeiros [18].

Swaptions é outra aplicação do domínio análise financeira. Ela utiliza a estrutura Heath-Jarrow-Morton (HJM) para calcular a taxa de juros em evolução do gerenciamento de risco e o passivo de patrimônio para uma classe predefinida de modelos. Swaptions utiliza o método de Monte Carlo (MC), que é utilizado para calcular a probabilidade numérica com base em um grande número de amostras aleatórias. A entrada *native* utiliza 128 *swaptions* e 1.000.000 de simulações.

Freqmine é usada para simular a tarefa de mineração de dados para Frequent Item Set Mining (FIMI), muito comum em aplicações que executam sequências de proteínas, dados de mercado e análise de logs. Sua principal característica é encontrar um padrão frequente em um grande banco de dados, minerando um conjunto de padrões usando o crescimento fragmentado. Este método é chamado de FP-growth que identifica os padrões que ocorrem com frequência e armazena as transações relevantes do banco de dados em uma estrutura de dados compacta. Para executar este *benchmark* a entrada *native* foi utilizada, composta por uma coleção de 250.000 documentos da Web em formato HTML. A implementação paralela é feita com o OpenMP.

O **Streamcluster** é um aplicação usada para simular um *cluster* de fluxo de dados (domínio de mineração de dados). O fluxo representa dados contínuos que são recebidos pela aplicação. Esse fluxo pode ser dados multimídia, transações financeiras e registros telefônicos. A operação de *clustering* de fluxo organiza os dados em condições de tempo real. Assim, o programa gasta a maior parte do tempo avaliando o ganho inicial de um novo centro e as formas de reduzir custos. Essa operação usa um paralelismo com o particionamento estático de pontos de dados e os dados originais são ligados à memória, se tornando cada vez mais intensivos em computação à medida que a dimensão aumenta. O Streamcluster calcula como reduzir custos abrindo um novo centro, realizando uma comparação entre analisar o custo de fazer um novo centro ou reatribuir o trabalho a partir do ponto já existente. A entrada *native* utilizada tem 1.000.000 pontos, 200.000 pontos de tamanho de bloco e 128 pontos de dimensões, 10-20 centros e tem até 5.000 centros intermediários permitidos.

Vips pertence ao domínio de processamento de mídia. Consiste em um sistema de processamento de imagens que inclui operações fundamentais de imagens como *affine transformation* e convolução. O VIPS pode avaliar a imagem utilizando uma abordagem paralela, combinando as operações que não necessitam de espaço em disco para imagens intermediárias e nenhuma E/S de disco. As transformações de imagem que VIPS realiza são tarefas comuns em computadores *desktop* e são organizadas em 18 etapas. Esta aplicação também cria um *pipeline* de processamento de imagem *multithread* e transparente em tempo de execução. A entrada *native* para o VIPS usa uma imagem com 18.000 x 18.000 pixels. O VIPS usa *memory-mapped_IO* para carregar sob demanda as partes de uma imagem de entrada.

O **Raytrace** é uma aplicação de renderização que simula a renderização de vídeo e cenas 3D. O resultado do processamento é uma imagem foto realista que usa um modelo de sombreamento. Esta técnica emprega informações globais para calcular as intensidades das sombras na imagem e comumente usada em animações em tempo real, como filmes e jogos de computador. A entrada *native* foi utilizada e consiste em uma imagem com resolução HDTV de 1920 X 1080 pixels.

2.5 Desempenho em ambientes de nuvem OpenStack

Avaliar o desempenho contribuiu para a evolução de diversas tecnologias. Por exemplo, os processadores foram os componentes computacionais que passaram por diversas avaliações de desempenho. Deste modo, por meio dos resultados gerados nessas etapas de pesquisa, atualmente existem os processadores multi-cores, multiprocessadores, co-processadores, entre outros. O processo de avaliar o desempenho consiste em entender os resultados que são alcançados. Somente obter determinada métrica de desempenho não significa totalmente que seu ambiente/aspecto avaliado está de acordo com as necessidades desejadas. Geralmente, o bom desempenho é quando software e hardware estão em sintonia, por isso, a necessidade da avaliação de desempenho e entender se esta sintonia está acontecendo.

Uma prática comum na computação é utilizar *benchmarks* para avaliar o desempenho de diferentes aspectos computacionais. Um *benchmark* é uma aplicação sintética desenvolvida com traços de uma aplicação real. Seu propósito é obter diferentes métricas de desempenho das execuções em um determinado ambiente computacional. Existem muitos *benchmarks* que avaliam diferentes tipos de aspectos computacionais, tanto do hardware como do software.

A escolha de um *benchmark* depende dos aspectos que precisam ser avaliados no cenário computacional e também o comportamento do *benchmark* durante as execuções. Em uma nuvem IaaS, o objetivo é analisar o desempenho dos recursos computacionais como processador, memória, rede e armazenamento das instâncias da nuvem. Então, sabendo que o *benchmark* é uma carga sintética com traços de uma aplicação, o *benchmark* deve utilizar principalmente os

recursos anteriormente citados quando executado em uma instância.

Existem outros aspectos que também podem ser avaliados em uma nuvem IaaS. Alguns deles são o tempo de lançamento de instâncias, migração entre *hosts* do *cluster*, tempo de *snapshot*, entre outros.

As métricas de desempenho estão relacionadas aos recursos computacionais avaliados. Os recursos avaliados serão processador, memória, disco e rede. Consequentemente, as métricas serão *bytes* por segundo (processamento), RAM *memory bandwidth*, OPS (*operation per second*), *latency* e *throughput*. Estas métricas serão obtidas por meio das execuções dos *benchmarks*, que estão relacionados a seguir:

- **LINPACK:** algoritmo que resolve sistemas lineares de multiplicação de matrizes.
- **STREAM:** avalia a velocidade das memórias através de taxas de transferência de dados.
- **UPerf:** avalia diferentes métricas de desempenho de rede.

Para os testes serão utilizados até dois servidores idênticos. Inicialmente, os *benchmarks* foram executados no cenário Nativo, onde não existe as camadas de virtualização. Posteriormente, os testes serão executados na nuvem com o virtualizador KVM.

No decorrer dos testes, o cuidado na alocação das VMs foi tomado, para que todos os *benchmarks* sejam executados no mesmo *host*, independentemente dos virtualizadores. Esse procedimento é necessário para evitar qualquer *outliers* na posterior análise dos resultados.

Na Figura 2.1 está ilustrado o cenário de testes. Essa figura ajuda a entender como foi planejado as alocações das VMs e quais aspectos avaliados. Inicialmente, partindo do cenário isolado (*dedicated*), 1 VM com capacidade de utilizar todos os recursos do *host* foi utilizada para a execução dos *benchmarks*. Os *benchmarks* serão executados isoladamente do restante dos *benchmarks* neste cenário. Por exemplo, ao iniciar as execuções com o *benchmark* de processador, até que ele seja finalizado nenhum outro será executado, evitando assim possíveis interferências. Neste cenário, as aplicações *multithread* irão variar de 1 a até 8 *threads* em paralelo.

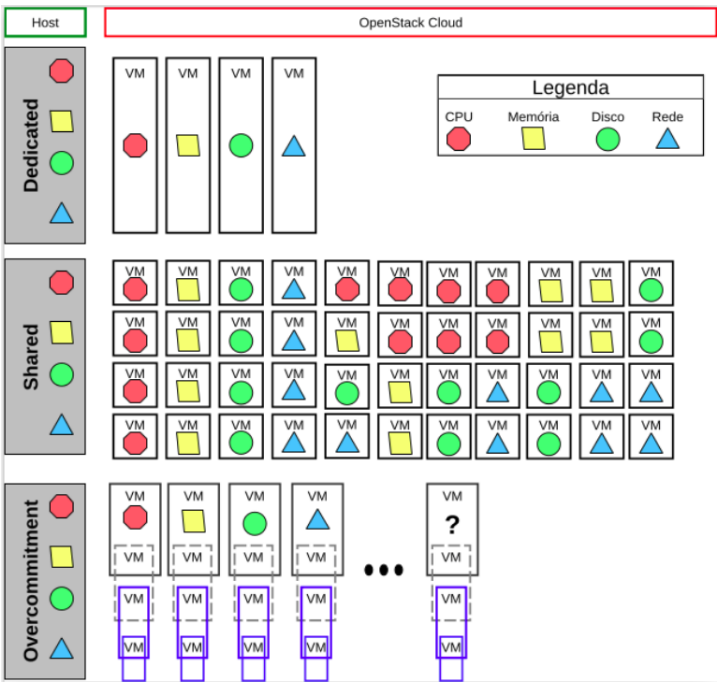


Figura 2.1: Cenário dos Testes

Já no cenário compartilhado (*shared*), no mínimo 4 VMs serão utilizadas durante cada etapa de execução. Neste cenário, as VMs estão dividindo a capacidade do *host* com outras VMs. Desse modo, até a conclusão desta etapa, o host sempre terá 4 VMs em execução com a distribuição e combinação dos *benchmarks* mostrados na Figura 03. Cada VM tinha disponível no mínimo 2 vCPU, 4GB de RAM, 250GB de disco, e 1 interface de rede 10/1000Gb/s. No cenário compartilhado, as aplicações *multithread* irão variar de 1 a até 2 *threads* em paralelo.

Cada *benchmark* foi executado 10 vezes em todos os ambientes. Porém, como os *benchmarks* têm comportamentos diferentes, os tempos de execuções de cada um também podem ser diferentes. Pensando nisso, as VMs com os *benchmarks* de execução mais rápida ainda ficarão executando os *benchmarks* até que todos estejam completos. As execuções além das 10 não serão calculadas nos gráficos de desempenho.

O cenário *overcommitment* demandou um planejamento e execução de testes mais cuidadoso. Para isto, é necessário os resultados de desempenho da etapa anterior (shared), para que o melhor cenário seja utilizado nesta etapa de testes. Por exemplo, no cenário compartilhado podemos encontrar casos isolados de *overcommitment* no recurso de rede. Este tipo de comportamento ainda será investigado no decorrer dos testes.

A carga de trabalho em cada *benchmark* deverá ser adequada ao ambiente de execução. Em casos específicos se tem apenas o algoritmo que deverá utilizado. Portanto, a carga sempre será a mesma, variando o número de *threads* em execução paralela. No STREAM, a quantidade de *threads* também será ajustada, variando em ordem crescente, respeitando os limites do cenário. No entanto, a variação existente será no tamanho do vetor alocado em memória. Esse tamanho será de acordo com a infraestrutura, para que nenhum dado seja alocado em memória cache do processador.

3. Resultados

Como apresentado no Capítulo 2, as ferramentas para implantação de ambientes de nuvem suportam diversas tecnologias e configurações. Nessa seção, ambientes de nuvem usando ferramentas distintas foram implantados e o objetivo principal é comparar os resultados e apresentar um viés inovador sobre eles. Na Seção 3.1, são mostrados os resultados de desempenho das aplicações sobre nuvens privadas com diferentes tecnologias de virtualização, os quais foram publicados no ISCC-2018 [8]. Ainda, na Seção 3.2 são apresentados resultados de uma implantação OpenStack comparada com o ambiente nativo.

3.1 Desempenho de Aplicações em Nuvens Privadas

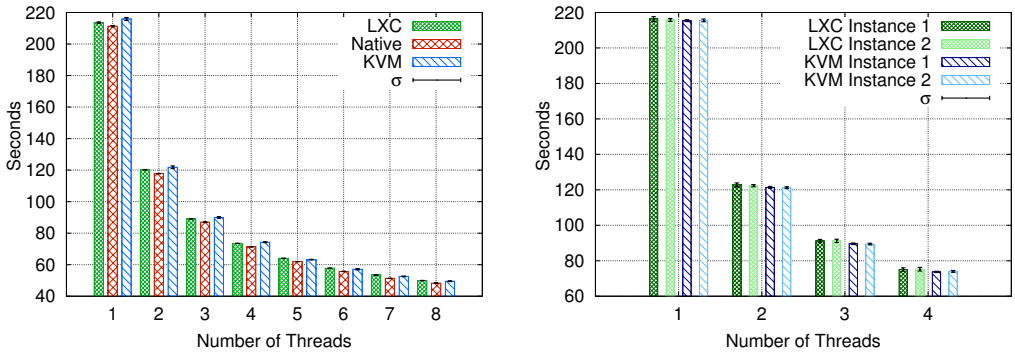
A metodologia experimental baseia-se em medições passivas para avaliar o desempenho e o comportamento dos *benchmarks* em nuvens privadas. Foram escolhidos seis *benchmarks* do PARSEC 3.0 (descritos na Seção 2.4) para representar as aplicações reais no cenário de uma nuvem. Duas nuvens privadas foram implantadas com as mesmas configurações de máquina utilizando a ferramenta CloudStack com os virtualizadores KVM (v.2.0.0) e LXC (v.1.0.8). Os recursos do ambiente de experimentos foram organizados para uso de maneira dedicada e compartilhada. Nos experimentos com recursos dedicados, o desempenho foi coletado de uma única e isolada instância. Por outro lado, o ambiente compartilhado de recursos representa um ambiente de nuvem real com vários usuários compartilhando os recursos físicos. Neste ambiente foi alocado duas instâncias que executaram na mesma máquina física. O número de *threads* utilizado foi limitado ao número de vCPUs disponíveis. Por exemplo, no ambiente compartilhado, o PARSEC foi executado com até 4 *threads*. Consequentemente, cada instância do ambiente compartilhado foi dimensionada com 4 vCPUs e 12 GB de RAM (metade do recurso de host total).

Existem alguns requisitos de arquitetura para implantar eficientemente uma nuvem para ambientes de produção [19]. Nos experimentos, foram adotadas as implementações padrões de nuvens típicas e populares para avaliar o desempenho das aplicações em condições de nuvem privada [26, 27]. Foi utilizado máquinas idênticas para normalizar o desempenho entre as nuvens implantadas. Cada máquina tem 24 GB de RAM, dois processadores Intel Xeon X5560 quad-core de 2,80 GHz com o *Hyperthreading* intencionalmente desabilitado e usando discos SATA II. As instâncias foram criadas com o Ubuntu Server 14.04 (kernel 3.19.0) que também foi utilizado para o ambiente nativo. Um *host* foi configurado como o gerenciador da nuvem usando a versão 4.8 da plataforma CloudStack e outros três hosts como nós de computação.

Os resultados dos experimentos com cada *benchmark* são apresentados em seguida na forma de gráficos. Foram plotados os tempos de execução para cada *benchmark* usando até 8 *threads*, para que de tal forma cada *thread* utilizasse um core/vCPU disponível. Os *benchmarks* executados dentro das instâncias tiveram o número de *threads* escalável de acordo com a configuração de cada instância. O tempo total de execução para todos os *benchmarks* não considera apenas o tempo da região paralela do *benchmark*.

Blackscholes (Figura 3.1) utiliza intensivamente a CPU para cálculos de pontos flutuantes. Essas operações são realizadas com 10.000.000 opções para carteiras financeiras e são calculadas utilizando um portfólio com 1.000 opções, que são pré-inicializados na aplicação. Os portfólios de entrada são lidos de um arquivo no disco e totalmente alocados na memória antes do início do processamento. As operações de E/S desta aplicação não influenciaram o tempo de execução. Foi observado que as instâncias do KVM nos cenários dedicados e compartilhados tiveram a maior degradação de desempenho, porém próximo dos resultados do ambiente nativo. Isto

acontece porque o KVM é capaz de executar nativamente as operações de ponto flutuante sem a interferência dos *drivers* KVM. Desse modo, com a granularidade de trabalho pequena, cada *thread* em execução nas instâncias da nuvem exploram eficientemente a memória cache do *host* físico. Os resultados também revelaram desempenho melhor nas instâncias LXC, onde o desempenho foi quase nativo devido à sua camada de virtualização.

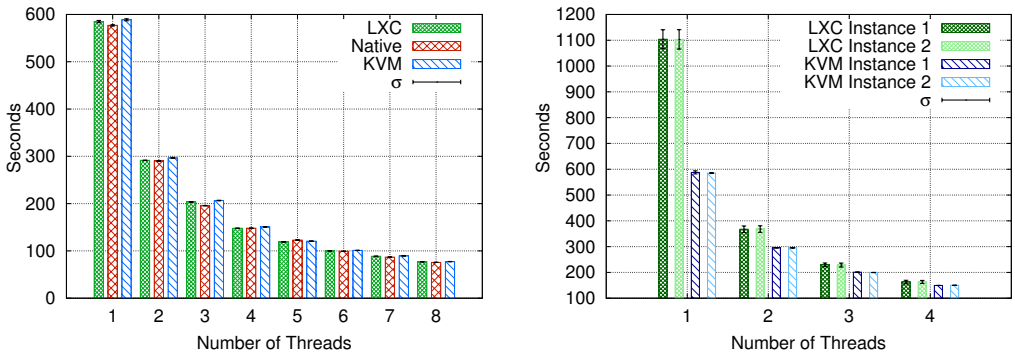


(a) Dedicated Machine Resources. (b) Shared Machine Resources.

Figura 3.1: Blackscholes Execution Times.

O Freqmine (Figura 3.2) processa uma grande entrada de transações de um banco de dados. Cada transação é composta de um conjunto de cliques em uma página web com tamanhos diferentes. Devido ao tamanho da entrada, a alocação na memória é gradual, de acordo com a frequência dos valores processados. O processamento usa intensivamente a memória, onde os acessos e as cargas são constantes. Esse comportamento é mais de leitura dos dados em memória invés da gravação [3]. Em consequência disso, Freqmine exige um alto compartilhamento de dados porque as *threads* processam os mesmos dados. As Figuras 3.2(a) e 3.2(b), mostram os resultados de desempenho da execução do Freqmine. Em geral, o tempo de execução apresentou resultados semelhantes para o ambiente dedicado no LXC, KVM e Native.

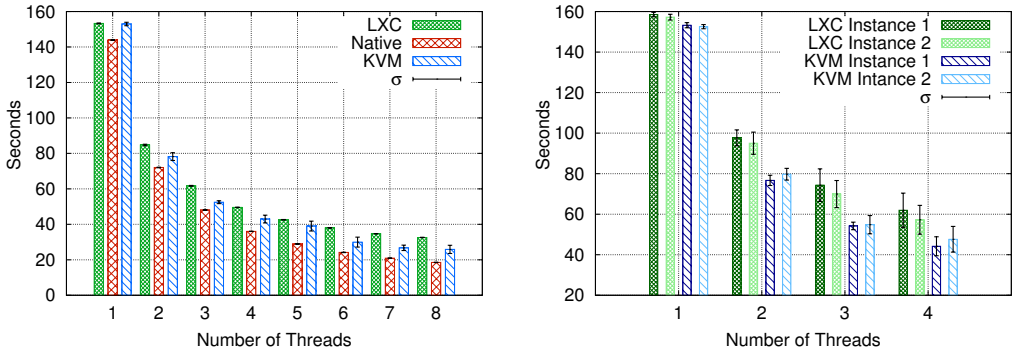
O Freqmine apresentou o mesmo desempenho nas instâncias de nuvem KVM no ambiente de recursos compartilhado. Em contraste, os resultados também revelaram a maior degradação de desempenho no ambiente compartilhado com as instâncias LXC. De acordo com [3], esta aplicação possui *overheads* de paralelismo gerado pelo alto número de *locks* (primitivas de sincronização). Este comportamento afeta as aplicações nas instâncias LXC devido ao alto compartilhamento de memória. Esse problema também foi relatado por [28], que demonstrou que o *cgroups* aplica um isolamento de memória ineficiente.



(a) Dedicated Machine Resources. (b) Shared Machine Resources.

Figura 3.2: Freqmine Execution Times.

O desempenho da aplicação VIPS é mostrado na Figura 3.3. O VIPS é uma aplicação de processamento de mídia com paralelismo de dados e grande granularidade, onde existe baixo compartilhamento de dados entre as *threads*. Os resultados da execução do VIPS no ambiente dedicado mostraram a degradação do desempenho em instâncias de nuvem devido ao grande número de operações de E/S que a aplicação executa. Essas operações não são executadas nativamente no *host* físico por causa da camada adicional de software que os virtualizadores aplicam sobre as VMs. Portanto, ele induz a sobrecarga de desempenho nas aplicações com E/S intensivos. Além disso, o VIPS carrega os dados do disco para a memória principal e as *threads* da aplicação se comunicam durante as fases de processamento. Comportamentos assim caracterizam um uso intensivo de memória e cache. Nas execuções com os recursos da máquina são compartilhados, as instâncias KVM tiveram um melhor desempenho. O desempenho inferior no LXC ocorre porque existe a sobrecarga adicional causada pelo isolamento insuficiente de recursos sob a comunicação intensiva entre *threads* e o uso da memória. Mesmo aspecto discutido anteriormente na aplicação Freqmine.



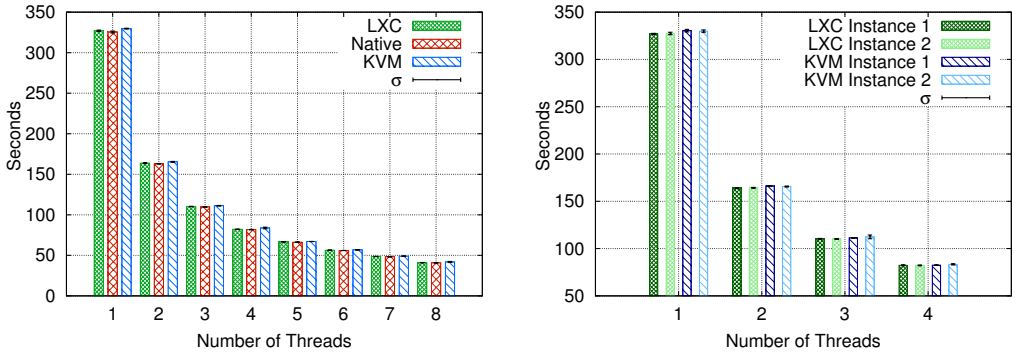
(a) Dedicated Machine Resources. (b) Shared Machine Resources.

Figura 3.3: Vips Execution Times.

Swaptions (Figura 3.4) é uma aplicação que realiza principalmente operações de ponto flutuante, onde a granularidade de trabalho é pequena e exige baixo compartilhamento entre as *threads*. Esta é uma aplicação com características similares ao Blackscholes. Swaptions divide a entrada em pequenos *chunks* e os armazena em uma matriz de portfólios. Em seguida, partes da matriz são igualmente distribuídas entre as *threads* em execução. Os resultados apresentados na Figura 3.4(a) e Figura 3.4(b) mostram que em ambos os experimentos (ambiente dedicado e compartilhado) os tempos de execução foram semelhantes entre os tipos de instância de nuvem e cenário nativo. No entanto, é observado que essa aplicação tem um desempenho um melhor nas instâncias do LXC. Como Blackscholes e Swaptions possuem características muito similares, a discussão do desempenho do Blacksholes se estende para o Swaptoins.

O Streamcluster (Figura 3.5) é uma aplicação de mineração de dados com características de granularidade média e baixo compartilhamento de dados. No ambiente dedicado, os tempos de execução apresentaram contraste entre os tipos de instâncias de nuvem com altos desvios padrão. Isso é resultado do ambiente e das características da aplicação. A execução do Streamcluster com uma única *thread* é diferente porque o tráfego na cache é caracterizado apenas por operações de leitura privadas e sem leituras compartilhadas [3]. Portanto, nessa situação, houve um uso de cache baixo ou inexistente. Consequentemente, as vantagens acima mencionadas disponíveis nas instâncias do KVM não foram exploradas ao executar o Streamcluster com uma única *thread*.

Os experimentos com o Streamcluster no ambiente compartilhado (Figura 3.5(b)) também revelaram ganhos de desempenho nas instâncias da nuvem. Foi identificado que esta aplicação usa a memória cache eficientemente para em arquiteturas *multithreading* [3]. Além disso, tanto

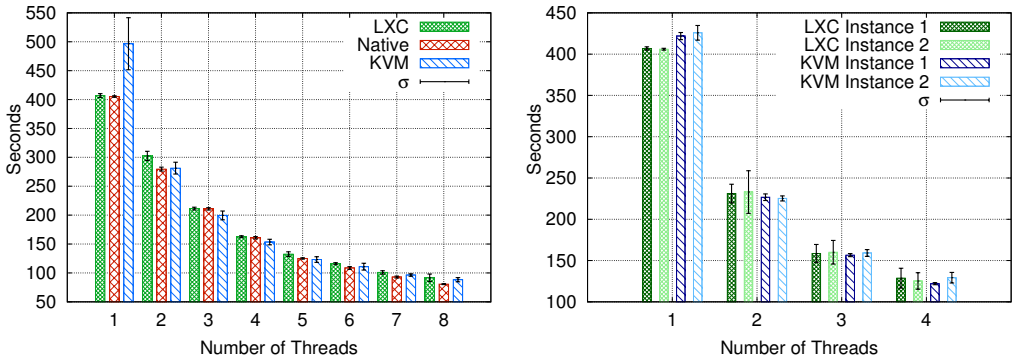


(a) Dedicated Machine Resources. (b) Shared Machine Resources.

Figura 3.4: Swaptions Execution Times.

o KVM quanto o LXC podem tirar proveito do Kernel Samepage Mergin (KSM)[1] que agrupa dados idênticos alocados em diferentes instâncias de máquinas virtuais em execução no mesmo *host* [1]. Portanto, o Streamcluster apresentou melhor desempenho nas instâncias de nuvem devido ao KSM.

Outro aspecto relacionado ao desempenho superior no ambiente compartilhado é que as máquinas possuem suporte à virtualização Intel VT-x [16]. Essa tecnologia permite que instâncias do KVM executem instruções com acesso nativo à CPU. Essa tecnologia impacta positivamente no Streamcluster porque a taxa de *cache-miss* é mínima, pois os dados utilizam a cache e permanecem nela durante a execução, pois a maioria das operações são carregadas da cache. Na verdade, executar instruções com acesso nativo à CPU permite um uso eficiente da cache. Além disso, as características desta aplicação combinadas com as configurações dos experimentos resultaram em desempenho otimizado.



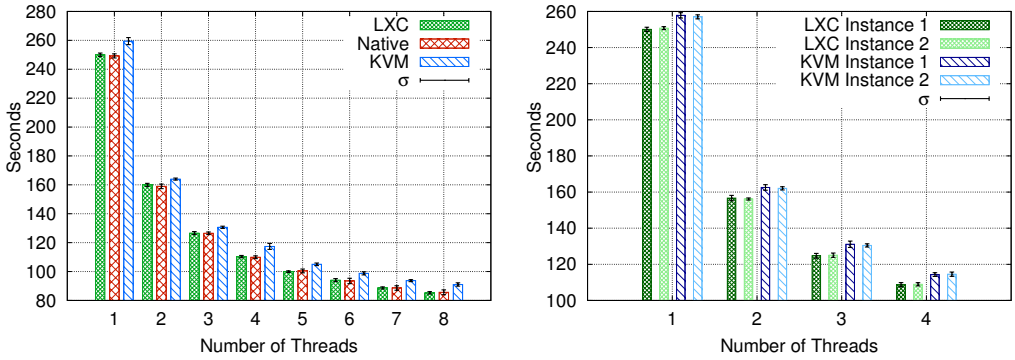
(a) Dedicated Machine Resources. (b) Shared Machine Resources.

Figura 3.5: Streamcluster Execution Times.

A Figura 3.6 apresenta os tempos de execução do Raytrace. Esta aplicação é caracterizada por paralelismo de dados e tem alto compartilhamento de dados e baixa troca de dados entre as *threads*. Esta aplicação apresentou pouca escalabilidade nos experimentos, uma vez que uma parte significativa do processamento é executado sequencialmente. Os resultados do Raytrace mostram um bom desempenho do LXC, com desempenho nativo no ambiente dedicado e melhor do que o KVM no ambiente compartilhado. Por outro lado, no KVM o baixo desempenho foi porque as instâncias não conseguiram tirar proveito da otimização de memória e cache que

[1]Esse recurso foi originalmente criado para *hosts* que executam máquinas virtuais. Recentemente, foi incorporado no kernel do Linux. Assim, o LXC também é capaz de tirar proveito disso

são oferecidas pela infraestrutura. Além disso, não há contraste significativo entre ambientes dedicados e compartilhados. Assim, esta aplicação provou mostrar características adequadas para ser executada em ambientes compartilhados em nuvens computacionais.



(a) Dedicated Machine Resources. (b) Shared Machine Resources.

Figura 3.6: Raytrace Execution Times.

3.2 Avaliação de implantação OpenStack

Diversas tecnologias são usadas em implantações de ambientes de nuvem, tanto em clientes quanto em provedores. Um dos aspectos mais relevantes é o desempenho da CPU, memória e rede em ambiente de nuvem. Por isso, nessa seção são apresentados resultados de avaliação e comparação do desempenho de infraestrutura em implantações de nuvem.

A metodologia usada nos testes é apresentada na Seção 2.5. Para comparar os resultados foi utilizado hardware idêntico e o desempenho de rede foi medido entre duas instâncias (máquinas virtuais) em cada um dos ambientes. O contexto da ferramenta OpenStack e tecnologias de virtualização é mostrado na Seção 2.1.

3.2.1 Ambiente Dedicado

A forma mais difundida e amplamente aceita para medir o desempenho de processadores é avaliar a capacidade de cálculo de FLOPS *FLoating-point Operations Per Second* (operações de ponto flutuante por segundo). Essa mesma métrica é também utilizada para avaliar o desempenho de diferentes configurações e ambientes em que os processador são usadas. O *benchmark* que calcula a capacidade de cálculos de pontos flutuantes em processadores é o LINPACK, como mostrado na Figura 3.7, que apresenta o resultado de desempenho no ambiente nativo (sem virtualização) e o resultado em uma instância de nuvem usando o virtualizador KVM.

Na Figura 3.7 é apresentado o comparativo entre o desempenho do ambiente nativo e KVM. O resultado no ambiente nativo é minimamente melhor quando comparado ao KVM, o que é um resultado *overhead* causado pela camada de virtualização do KVM, resultado em um aumento no tempo total de execução dessa aplicação.

Na Figura 3.8 são apresentados os resultados de *bandwidth* de memória comparando os dois ambientes considerando diferentes operações de memória e *threads* de processamento. O ambiente nativo apresenta, em geral, um desempenho levemente maior. Porém, o ambiente KVM teve um desempenho muito próximo ao ambiente nativo. Dessa forma, esse resultado mostrou que o *overhead* causado pelo KVM no acesso a memória é mínimo.

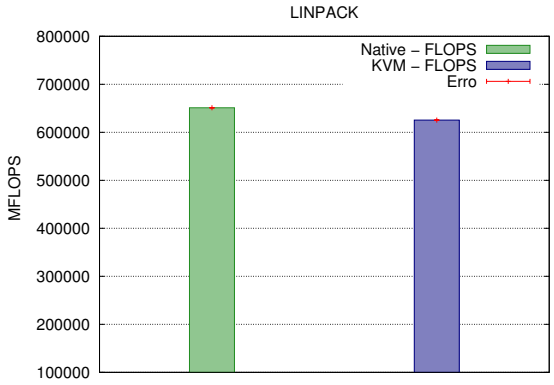


Figura 3.7: LINPACK - CPU

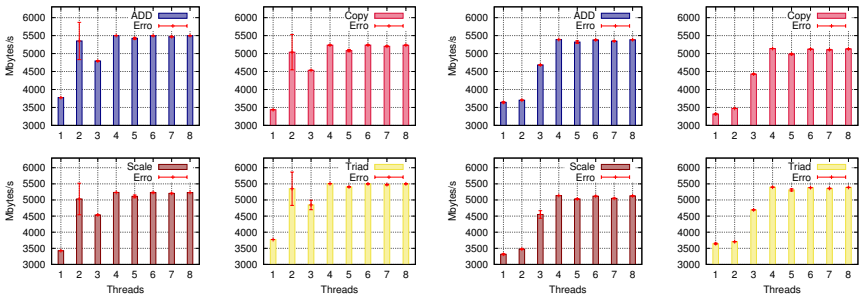


Figura 3.8: STREAM Memória - Nativo (Esquerda) e KVM (Direita)

Na Figura 3.9 é apresentado o resultado do desempenho de rede. É apresentado o *throughput* da rede nos dois ambientes com diferentes números de *threads*. É importante destacar que nesse experimento do *Uperf*, o objetivo não é atingir o máximo da vazão de rede, mas avaliar o comportamento considerando diferentes aspectos, como latência, consumo de CPU e transmissão de pacotes.

O resultado da Figura 3.9 mostra que o desempenho do KVM foi similar ao ambiente nativo. Com 5 e 6 *threads*, o ambiente nativo foi levemente superior ao KVM. Por outro lado, o ambiente KVM foi melhor com 7 e 8 *threads*. Isso mostra que o KVM consegue atingir um isolamento de recursos superior ao nativo quando várias *threads* competem pelos recursos computacionais.

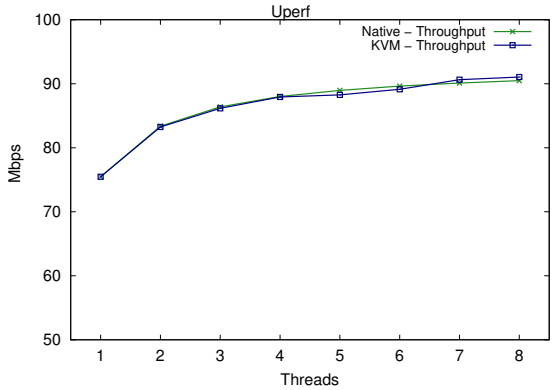


Figura 3.9: Uperf - Desempenho de Rede

3.2.2 Ambiente Shared (Compartilhado)

Os ambientes nativo e KVM também foram avaliados em um cenário compartilhado, que se refere a um teste onde os *benchmarks* não são executados em um ambiente dedicando, de tal forma que existe mais de um *benchmark* executado ao mesmo tempo. Na Figura 3.10 são apresentados os resultados do LINPACK em um ambiente compartilhado com 4 execuções simultâneas. Esse resultado evidencia a média de desempenho do ambiente KVM superior ao ambiente nativo. Isso ocorre porque o KVM tem um melhor isolamento de recursos quando múltiplos processos competem por recursos.

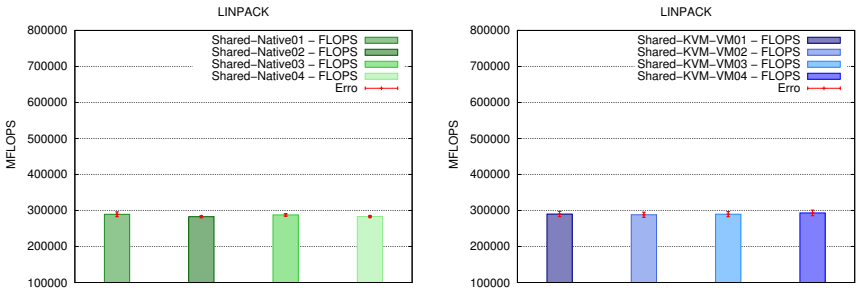


Figura 3.10: LINPACK CPU - Nativo (Esquerda) e KVM (Direita)

Na Figura 3.11 são mostrados resultados do *bandwidth* de memória no STREAM da execução 1. Nesse cenário compartilhado fica evidente uma maior variação no desempenho. No ambiente compartilhado, o STREAM teve 4 execuções simultâneas, por limitações de espaço e clareza são apresentados resultados apenas de duas execuções, nas Figuras 3.11 e 3.12.

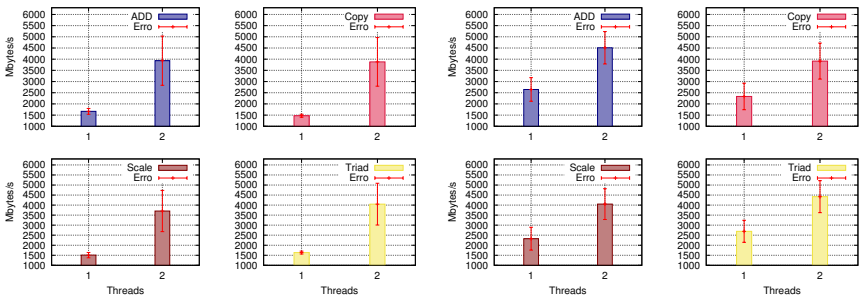


Figura 3.11: Instância 1: STREAM - Nativo (Esquerda) e KVM (Direita)

Enquanto na Figura 3.11 o desempenho entre os ambientes é semelhante, na Figura 3.12, o desempenho do KVM é superior ao ambiente nativo, sendo o KVM mais efetivo para isolar o desempenho das múltiplas instâncias e evitar degradação de desempenho.

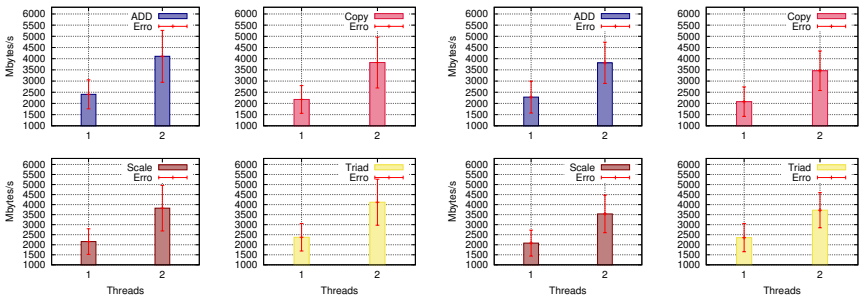


Figura 3.12: Instância 4: STREAM - Nativo (Esquerda) e KVM (Direita)

3.2.3 Ambiente *Overcommitted* (Superprovisionado)

No cenário *overcommitted* existe um provisionamento excessivo de recursos, de tal forma que é criada uma quantidade maior de processos do que CPUs disponíveis no nodo. Isso resulta em uma concorrência adicional bem como competição pelos recursos. Novamente são apresentados os resultados de cálculos em FLOPS usando o LINPACK, mas dessa vez são 6 *threads* executadas de forma simultânea. De modo geral, o desempenho é melhor e mais estável no KVM, pois este possui camadas adicionais que melhoram o isolamento de recursos.

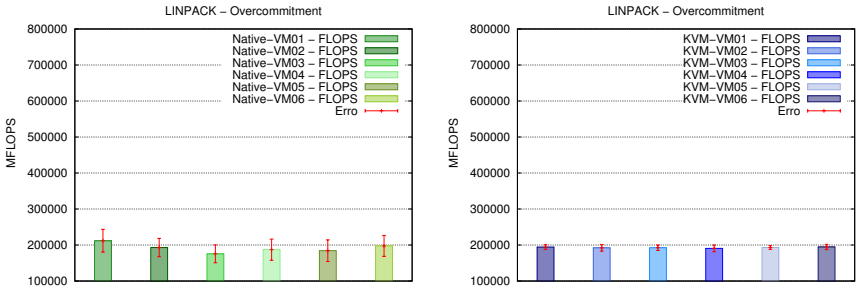


Figura 3.13: LINPACK CPU - Nativo (Esquerda) e KVM (Direita)

Os testes de memória também foram executados no cenário *overcommitted*, como mostrado nas Figuras 3.14 e 3.15, que apresentam resultados de algumas instâncias representativas desse cenário. Novamente se repete a tendência de uma maior variação nos resultados, devido a concorrência adicional. Na instância 3, o ambiente nativo é melhor tendo um *bandwidth* de memória superior ambiente KVM. Por outro, na Figura 3.15 os resultados da instância 6 são melhores no KVM comparados ao ambiente nativo.

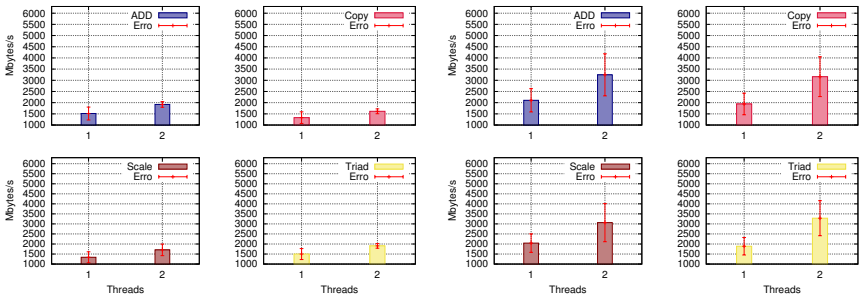


Figura 3.14: Instância 3 - STREAM - Nativo (Esquerda) e KVM (Direita)

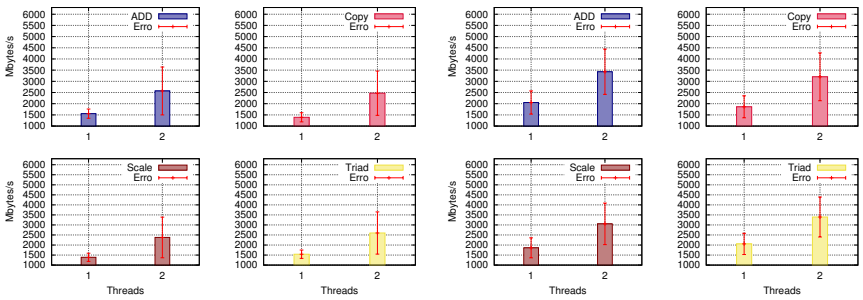


Figura 3.15: Instância 6 - STREAM - Nativo (Esquerda) e KVM (Direita)

4. Conclusão

Neste relatório técnico foram apresentados experimentos relacionados a computação em nuvem, os quais envolvem a avaliação da implantação de uma nuvem privada e a avaliação de aplicações de mineração, multimídia e financeiras em nuvens computacionais. Para isto, na Seção 2.1 foram contextualizados as ferramentas de computação em nuvem as tecnologias de virtualização. Na Seção 2.4, foram detalhadas as aplicações utilizadas nos experimentos, as quais fazem parte da suíte PARSEC.

A implantação do ambiente de nuvem privada usando OpenStack com a tecnologia de virtualização KVM atingiu bons resultados de desempenho. Embora no cenário dedicado, o ambiente nativo obteve o melhor desempenho. No cenário compartilhado e *overcommitted*, o ambiente implantado de nuvem baseado em OpenStack/KVM obteve os melhores resultados comparado ao ambiente nativo, tendo também uma menor variação nos testes.

Os ambientes de nuvem privada são alternativas atraentes para aplicações com uso intensivo de recursos, como os *benchmarks* do PARSEC. Durante a execução dos testes, o *overcommitting* de recursos foi evitado ao executar mais de uma instância de nuvem em um único *host*, pois o administrador do sistema recebe o controle da nuvem privada para garantir um melhor desempenho. Em contraste, estudos relacionados abordaram o desempenho relativo ao *overcommitting* excessivo da CPU em ambientes de nuvem pública.

Embora o PARSEC seja um conjunto de *benchmarks*, ele pode ser utilizado para representar cargas de trabalho de aplicações reais. Por exemplo, Raytrace pode ser encontrado em aplicações de modelagem 3D e Blackscholes em modelagem financeira para várias aplicações conhecidas [23]. Assim, analisar as descobertas sobre o desempenho dessas aplicações em infraestruturas de nuvem privada pode ser extrapolado para o comportamento de outras aplicações com características semelhantes. Os provedores públicos também podem se beneficiar dos resultados apresentados, já que nos testes foram utilizados as mesmas tecnologias de virtualização.

Os resultados mostram que o desempenho de uma aplicação em um ambiente de nuvem varia de acordo com seu comportamento, o ambiente e as tecnologias de virtualização utilizadas. Para algumas aplicações, as instâncias de nuvem tiveram um bom desempenho, em alguns casos quase nativo. Por outro lado, algumas aplicações com características específicas apresentaram sobrecarga nas instâncias de nuvem. No ambiente dedicado, as instâncias do LXC tiveram, em média, um desempenho um pouco melhor do que as instâncias do KVM.

Os resultados de Blackscholes, Swaptions e Freqmine revelam um desempenho quase nativo, devido à sobrecarga insignificante em ambos os cenários de nuvem. Por exemplo, comparando com o desempenho nativo em Blachscholes com 2 threads, existe uma pequena diferença de 3.2% em instâncias do KVM e 2.0% em instâncias do LXC.

Os resultados do Raytrace e do VIPS nas instâncias do KVM demonstram o impacto positivo causado pelo compartilhamento de dados na nuvem (tecnologia KSM). Por outro lado, existe uma sobrecarga de desempenho nas instâncias do KVM na aplicação Raytrace (até 6,3 % com 4 *threads*) devido ao alto compartilhamento de dados entre as *threads* em execução. O Vips é caracterizado pelo baixo compartilhamento de dados, assim, as instâncias do KVM apresentaram desempenho melhor do que as instâncias do LXC (ex, 3.1% com uma *thread* no ambiente compartilhado). O KVM tem desafios em relação ao compartilhamento de dados entre as *threads* dentro de instâncias [6], embora as características do Vips tenham resultados de desempenho melhores nas instâncias do KVM.

Por fim, o LXC é uma opção para cargas de trabalho sensíveis ao desempenho, pois os contêineres costumam apresentar melhor desempenho do que as tecnologias de virtualização completa ou de para-virtualização. De fato, o desempenho das aplicações do PARSEC em

instâncias LXC (em comparação com instâncias KVM) foram melhores no ambiente dedicado na maioria dos testes. Em alguns cenários, o KVM superou o LXC em um ambiente compartilhado. Portanto, a virtualização de *hipervisor* provou ser mais eficaz no isolamento de recursos. Além disso, do ponto de vista do gerenciamento de recursos, os contêineres nem sempre é a alternativa mais adequada, pois carecem de funcionalidades e possuem limitações de flexibilidade e compatibilidade.

Referências Bibliográficas

[1] Arcangeli, A., Eidus, I., Wright, C.: Increasing Memory Density by Using KSM. In: Proceedings of the Linux Symposium. pp. 19–28 (2009)

[2] Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M.: A View of Cloud Computing. Commun. ACM 53(4), 50–58 (Apr 2010)

[3] Bienia, C.: Benchmarking Modern Multiprocessors. Ph.D. thesis, Princeton University, Princeton, NJ, USA (2011)

[4] Buyya, R., Vecchiola, C., Selvi, S.: Mastering Cloud Computing: Foundations and Applications Programming. ITPro collection, Elsevier Science (2013)

[5] Cloudstack: Cloudstack (2016), https://cloudstack.apache.org/, last access Aug, 2016

[6] Felter, W., Ferreira, A., Rajamony, R., Rubio, J.: An Updated Performance Comparison of Virtual Machines and Linux Containers. In: 2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS). pp. 171–172 (March 2015)

[7] Folkerts, E., Alexandrov, A., Sachs, K., Iosup, A., Markl, V., Tosun, C.: Benchmarking in the Cloud: What It Should, Can, and Cannot Be. In: Nambiar, R., Poess, M. (eds.) Selected Topics in Performance Evaluation and Benchmarking: 4th TPC Technology Conference, TPCTC 2012, Istanbul, Turkey, August 27, 2012, Revised Selected Papers. pp. 173–188. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)

[8] 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: 23rd IEEE Symposium on Computers and Communications (ISCC). IEEE, Natal, Brazil (June 2018)

[9] Iosup, A., Prodan, R., Epema, D.: IaaS Cloud Benchmarking: Approaches, Challenges, and Experience. In: Li, X., Qiu, J. (eds.) Cloud Computing for Data-Intensive Applications, pp. 83–104. Springer New York, New York, NY (2014)

[10] Khajeh-Hosseini, A., Greenwood, D., Sommerville, I.: Cloud Migration: A Case Study of Migrating an Enterprise IT System to IaaS. In: 2010 IEEE 3rd International Conference on Cloud Computing. pp. 450–457 (July 2010)

[11] Kourai, K., Nakata, R.: Analysis of the Impact of CPU Virtualization on Parallel Applications in Xen. In: 2015 IEEE Trustcom/BigDataSE/ISPA. vol. 3, pp. 132–139 (Aug 2015)

[12] 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)

[13] 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)

[14] Mehrotra, P., Djomehri, J., Heistand, S., Hood, R., Jin, H., Lazanoff, A., Saini, S., Biswas, R.: Performance Evaluation of Amazon EC2 for NASA HPC Applications. In: Proceedings of the 3rd Workshop on Scientific Cloud Computing. pp. 41–50. ScienceCloud '12, ACM, New York, NY, USA (2012)

[15] Mell, P., Grance, T., et al.: Sp 800-145. 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)

[16] Neiger, G., Santoni, A., Leung, F., Rodgers, D., Uhlig, R.: Intel Virtualization Technology: Hardware Support for Efficient Processor Virtualization. Intel Technology Journal 10(3) (2006)

[17] OpenStack: OpenStack Roadmap (2016), http://openStack.org/software/roadmap/, last access Aug, 2016

[18] PARSEC: Princeton Application Repository for Shared-Memory Computers <http://parsec.cs.princeton.edu/overview.htm> (2017), last access Oct, 2017

[19] Rimal, B.P., Jukan, A., Katsaros, D., Goeleven, Y.: Architectural Requirements for Cloud Computing Systems: An Enterprise Cloud Approach. J. Grid Comput. 9(1), 3–26 (Mar 2011)

[20] Rista, C., Griebler, D., Maron, C.A.F., Fernandes, L.G.: Improving the network performance of a container-based cloud environment for hadoop systems. In: 2017 International Conference on High Performance Computing Simulation (HPCS). pp. 619–626 (July 2017)

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

[22] Sabharwal, N.: Apache CloudStack Cloud Computing. Packt Publishing (2013)

[23] Shinde, A., Takale, K.: Study of Black-Scholes Model and its Applications. Procedia Engineering 38(Supplement C), 270–279 (2012), international Conference on Modelling Optimization and Computing

[24] Vecchiola, C., Pandey, S., Buyya, R.: High-performance cloud computing: A view of scientific applications. In: Pervasive Systems, Algorithms, and Networks (ISPAN), 2009 10th International Symposium on. pp. 4–16. IEEE (2009)

[25] 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)

[26] Vogel, A., Griebler, D., Maron, C.A.F., Schepke, C., Fernandes, L.G.: Private IaaS Clouds: A Comparative Analysis of OpenNebula, CloudStack and OpenStack. In: 24th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP). pp. 672–679. IEEE, Heraklion Crete, Greece (Febuary 2016)

[27] Vogel, A., Griebler, D., Schepke, C., Fernandes, L.G.: An Intra-Cloud Networking Performance Evaluation on CloudStack Environment. In: 25th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP). p. 5. IEEE, St. Petersburg, Russia (March 2017)

[28] Zhuang, Z., Tran, C., Weng, J., Ramachandra, H., Sridharan, B.: Taming Memory Related Performance Pitfalls in Linux Cgroups. In: 2017 International Conference on Computing, Networking and Communications (ICNC). pp. 531–535 (Jan 2017)