UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Adriano José Ferruzzi

Análise do Comportamento de Aplicações Paralelas em Ambientes de Computação de Alto Desempenho Virtualizados

Campinas

2021

UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Adriano José Ferruzzi

Análise do Comportamento de Aplicações Paralelas em Ambientes de Computação de Alto Desempenho Virtualizados

Tese apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica, na Área de Engenharia de Computação.

Orientador: Prof. Dr. Christian Rodolfo Esteve Rothenberg

Co-orientador Prof. Dr. Leandro Martinez

Este exemplar corresponde à versão final da tese defendida pelo aluno Adriano José Ferruzzi, orientada pelo Prof. Dr. Christian Rodolfo Esteve Rothenberg, e coorientada pelo Prof. Dr. Leandro Martinez.

Campinas

2021

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Rose Meire da Silva - CRB 8/5974

F418a

Ferruzzi, Adriano José, 1988-
Análise do comportamento de aplicações paralelas em ambientes de computação de alto desempenho virtualizados / Adriano José Ferruzzi. – Campinas, SP : [s.n.], 2021.

Orientador: Christian Rodolfo Esteve Rothenberg.
Coorientador: Leandro Martinez.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Computação de alto desempenho. 2. Virtualização. I. Esteve Rothenberg, Christian Rodolfo, 1982--. II. Martinez, Leandro, 1979-. III. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: Analysis os parallels applications in virtualized high performance computing evironments
Palavras-chave em inglês:
High performance computing
Virtualization
Área de concentração: Engenharia de Computação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora:
Christian Rodolfo Esteve Rothenberg [Orientador]
William Roberto Wolf
José Raimundo de Oliveira
Data de defesa: 29-03-2021
Programa de Pós-Graduação: Engenharia Elétrica

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0002-1268-7132
- Currículo Lattes do autor: http://lattes.cnpq.br/5813261133047060

COMISSÃO JULGADORA - DISSERTAÇÃO DE MESTRADO

Candidato: Adriano José Ferruzzi RA: 180112

Data da Defesa: 29 de março de 2021

Título da Tese: Análise do comportamento de aplicações paralelas em ambientes de computação de alto desempenho virtualizados.

Prof. Dr. Christian Rodolfo Esteve Rothenberg (Presidente)
Prof. Dr. William Roberto Wolf
Prof. Dr. José Raimundo de Oliveira

A ata de defesa, com as respectivas assinaturas dos membros da Comissão Julgadora, encontra-se no SIGA (Sistema de Fluxo de Dissertação/Tese) e na Secretaria de Pós Graduação da Faculdade de Engenharia Elétrica e de Computação.

Dedico à minha esposa Andressa Ferruzzi.

Agradecimentos

Primeiramente gostaria de agradecer a paciência e apoio da minha esposa Andressa Ferruzzi que sempre me apoiou durante o período de estudo para concluir o mestrado, além de me apoiar ela também serviu de inspiração para continuar estudando. Agradeço muito aos meus pais e meu irmão que me apoiaram todos esses anos.

Sou muito grato também ao orientador prof. Christian Rothenberg e o co-orientador prof. Leandro Martinez que me apoiaram muito com insights e conselhos. Ambos me deram uma visão melhor do meu projeto sempre me orientando para atingir os resultados desejados. Agradeço ao prof. Munir Skaf que me incentivou a continuar meus estudos mesmo com a grande carga de trabalho do nosso grupo. Agradeço também às pessoas do nosso grupo, tanto aqueles que já estavam aqui quanto aqueles que chegaram depois, pois eles tornaram os dias de trabalho mais divertido. Agradeço a todos do departamento de Informática que sempre me ajudaram com as minhas dúvidas do dia a dia.

O presente trabalho foi realizado com apoio do Centro de Computação em Engenharia e Ciências (CCES - Fapesp 2013/08293-7). Além disso, O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001. Finalmente, gostaria de agradecer aos membros do grupo Intrig que me apoiaram em todo meu desenvolvimento. Espero continuar a contribuir com todos os grupos, pois eles me ajudaram a crescer profissionalmente e pessoalmente.

"Em Deus nós confiamos; todos os outros devem trazer dados."
(William Edwards Deming)

Resumo

A computação em nuvem tem se tornado mais frequente nas empresas e universidades, principalmente devido ao fato da nuvem computacional ser mais facilmente administrada uma vez que seu ambiente esteja devidamente configurado. Entretanto, há resistência na implantação de ambientes de computação de alto desempenho em nuvem e isso pode estar associado ao fato da nuvem precisar de um `hypervisor` para a administração das máquinas virtuais, resultando em uma perda de desempenho causado pela camada de virtualização. Por outro lado, as tecnologias de virtualização estão em constante avanço, tornando essa perda de desempenho menor com o passar dos anos. Sendo assim, este trabalho faz uma análise do desempenho de aplicações paralelas, utilizando benchmarks e aplicações paralelas através de um estudo de caso voltado principalmente à aplicações da área da físico-química, visando testar a infraestrutura voltada à computação de alto desempenho, avaliar as características dos `hypervisors` XenServer e KVM com o propósito de comparar o desempenho de cada ambiente e identificar o `hypervisor` com o melhor desempenho a ser utilizado na criação de um ambiente de computação de alto desempenho virtualizado.

Palavras-chaves: virtualização; nuvem; alto desempenho.

Abstract

Cloud computing has become increasingly common in companies and universities. This happened because cloud computing has made management easier once your environment has been completely configured. However, there is resistance in deploying high-performance cloud computing environments and it is associated with the fact that the cloud needs a hypervisor to manage the virtual machines. It results in a performance loss caused by the virtualization layer. On the other hand, virtualization technologies are constantly advancing and make the performance loss smaller over the years. Therefore, this work makes an analysis of the performance of parallel applications, using benchmarks and parallel applications and a case study focused on applications in the area of physical chemistry, aiming to test the infrastructure aimed at high performance computing and evaluate the characteristics of the hypervisors XenServer and KVM to compare the performance of each environment and identify the hypervisor with the best performance to be used in the creation of a high performance virtualized computing environment.

Keywords: Virtualization; cloud computing; high performance computing.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Com a disponibilidade de alto processamento, memória e armazenamento, a virtualização e a computação em nuvem se tornaram cada vez mais acessíveis. Usando computação em nuvem pode-se deixar de investir em grandes servidores para investir em aplicações, possibilitando uma economia de recursos e custos com infraestrutura. Entretanto, esse cenário não ocorre quando o assunto é computação de alto desempenho. Uma das razões é o fato da nuvem computacional necessitar de `hypervisors` para fazer a virtualização das máquinas, gerando mais uma camada para processamento e, consequentemente, diminuindo a performance das aplicações.

Este trabalho foca precisamente no comportamento de uso de cpu, memória e disco feito pelos `hypervisors` e pretende analisar o comportamento de aplicações científicas. O trabalho tem como objetivo mapear o comportamento e o impacto da virtualização voltada a ambientes de computação de alto desempenho e consequentemente identificar o `hypervisor` com melhor desempenho.

O estudo será desenvolvido com preceitos na literatura, pesquisa bibliográfica e estudo de caso com ambientes reais. Para atingir tal propósito o trabalho será dividido em oito capítulos, onde o primeiro se trata deste preâmbulo que apontou a justificativa da temática, objetivos e metodologia. O capítulo 2 apresenta os objetivos gerais e específicos deste trabalho. Por sua vez, o capítulo 3 faz uma revisão bibliográfica que disserta a respeito da computação de alto desempenho, virtualização de maneira breve. O capítulo 4 mostra uma revisão dos trabalhos relacionados com o mesmo tema. O capítulo 5 apresenta a metodologia com a proposta de desenvolvimento dos ambientes e seus softwares de análise. Em seguida, no capítulo 6, são analisados os resultados obtidos nos `benchmarks` e discute-se qual `hypervisor` obteve melhor desempenho. No capítulo 7, é feito um estudo de caso com softwares de cálculo científico, dessa forma, é possível avaliar o desempenho das máquinas virtuais utilizando aplicações reais. Finalmente, no capítulo 8, é feita a conclusão do trabalho que ressalta os resultados alcançados, além de discorrer sobre possíveis trabalhos futuros.

2 Objetivos

Este trabalho propõe apresentar uma comparação entre ferramentas de virtualização, explicar a maneira como funcionam e suas particularidades. Em seguida serão apresentadas as análises dos `hypervisors` e feita a comparação entre eles, servindo como um guia para a implementação da virtualização voltada à computação de alto desempenho. Em uma segunda etapa, será feito um estudo de caso com aplicações de dinâmica molecular e, dessa forma, concluir qual o `hypervisor` possui o melhor desempenho.

2.1 Objetivos específicos deste trabalho

- Configurar quatro tamanhos de ambientes virtuais distintos utilizando os `hypervisors` gratuitos XenServer e o KVM, além do ambiente nativo;
- Fazer uma análise de performance de cada ambiente em comparação com o ambiente nativo;
- Analisar as particularidades de comportamento de cada ambiente, baseando-se em estudos de desempenho anteriores e nos resultados;
- Identificar qual `hypervisor` possui melhor desempenho para a construção de uma nuvem dedicada a computação de alto desempenho.

3 Revisão Bibliográfica

Neste capítulo será apresentada uma revisão bibliográfica dos conceitos relacionados a essa dissertação de mestrado, onde nas seções seguintes serão apresentados os conceitos de computação de alto desempenho, virtualização e nuvem computacional. Esses conceitos formam a base teórica para reforçar a proposta apresentada.

3.1 Computação de Alto Desempenho

Computação de alto desempenho se refere ao uso de um conjunto de computadores para resolução de grandes tarefas e esse conjunto de computadores é chamado de cluster. Esse ambiente é caracterizado por tarefas paralelas e complexas sendo executadas em um mesmo instante. Essas tarefas normalmente são simulações numéricas que necessitam de grande quantidade de recursos computacionais para sua resolução (COLVERO TAIS A.; DANTAS, 2005). De forma simplificada, cluster é um sistema que utiliza dois ou mais servidores para trabalharem de maneira conjunta com o objetivo de executar uma tarefa de alto custo computacional. Além disso, existem outros tipos de cluster, como o cluster de alta disponibilidade e o cluster de balanceamento de carga. O cluster de alta disponibilidade faz uso de dois ou mais servidores com o objetivo de tornar um serviço disponível o maior tempo possível. O cluster de balancemento de carga utiliza vários servidores simultaneamente para garantir que as requisições dos clientes sejam distribuidas entre vários servidores, neste caso, o objetivo é garantir que nenhum dos servidores fique sobrecarregado. Este trabalho utiliza apenas a arquitetura de cluster de computação de alto desempenho.

Os clusters de computação de alto desempenho são voltados à aplicações que necessitam de muito processamento como sistemas de previsão meteorológica, simulação de proteínas, genômica, solução de escoamentos turbulentos e combustão, e muitos outros problemas que envolvam uma grande quantidade de cálculos. Para conseguir tal feito, os nós do cluster de alto desempenho são interligados com utilização de redes, seja rede Ethernet ou Infiniband de alta performance. Essa configuração da rede no cluster é de extrema importância, pois é através dela que são disponibilizados os mesmos arquivos para todos os nós. Isso possibilita a partição de um grande problema em vários nós ao mesmo tempo para que seja realizado a computação em paralelo permitindo alcançar o seu principal objetivo que é gerar resultados da maneira mais rápida possível.

Normalmente os clusters de computação de alto desempenho são configurados

com sistema operacional Linux ou Unix e utiliza algumas bibliotecas para computação paralela como OpenMP e MPI. Além disso, esse tipo de cluster é configurado de forma homogênea, ou seja, vários computadores com a mesma arquitetura e com mesmo sistema operacional que trabalham em uma rede de alta performance, visando calcular grandes problemas científicos o mais rápido possível.

3.2 Virtualização

A ideia de virtualização é antiga e surgiu na década de 1960 quando a IBM utilizava essa tecnologia em mainframes desde 1965. Atualmente essa tecnologia está disponível para plataforma x86 desde 2001 (CACIATO, 2009). A virtualização é uma tecnologia que possibilita a execução de vários sistemas operacionais e aplicativos simultaneamente no mesmo servidor. Na prática, podem ser criados vários servidores virtuais dentro de um único servidor físico. Algumas das soluções mais famosas na virtualização são a VMWare, Microsoft Hyper-V, Xen, KVM e Virtualbox. A figura 3.1 exemplifica a virtualização de quatro máquinas virtuais em um servidor físico.

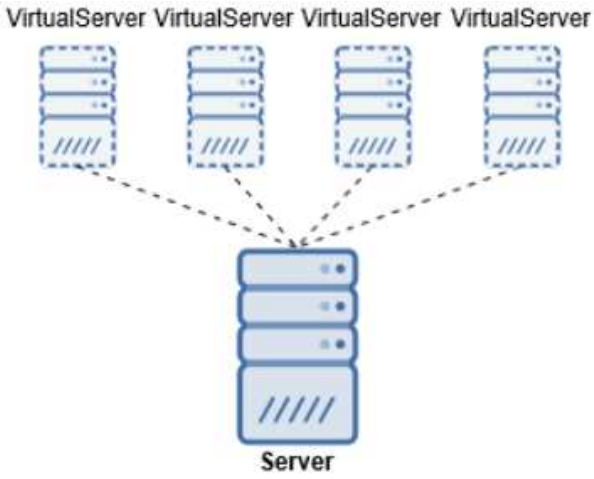


Figura 3.1 – Arquitetura de virtualização de quatro máquinas virtuais em um servidor físico.

Cada máquina virtual oferece todos os recursos de computação e trabalha de forma transparente, dessa forma é possível utilizar todos os recursos de hardware do servidor físico. Com a virtualização é possível executar dois ou mais sistemas distintos no mesmo servidor físico, aproveitando possíveis recursos ociosos e evitando a compra de novos equipamentos (CACIATO, 2009). Essa atividade de virtualizar antigos servidores físicos de storage, aplicações e outros em servidores virtuais é chamada de consolidação de datacenter.

A figura 3.2 ilustra a consolidação de servidores de um datacenter, passando de três servidores para apenas um. A virtualização permite particionar um único sistema

Figura 3.2 – Consolidação de servidores.

computacional em vários outros denominados de máquinas virtuais (CARISSIMI, 2008). Cada máquina virtual oferece um ambiente completo muito similar a uma máquina física. Dessa forma, é possível consolidar o trabalho que antes era alocado em três servidores para apenas um servidor físico.

3.2.1 Tipos de Virtualização

As máquinas virtuais podem ser distinguidas de acordo com a localização da camada de virtualização. Se a camada de virtualização estiver diretamente sobre a camada de hardware ela é chamada de Tipo 1 ou Nativa. Caso a camada de virtualização estiver em cima de um sistema operacional, é chamada de Tipo 2 ou Modell Hosted Virtualization (CARISSIMI, 2008) e (SANTOS, 2009). Além desses 2 tipos de virtualização ainda existem dois tipos básicos de virtualização, a virtualização completa e a paravirtualização. Trata-se de virtualização completa aquela onde toda a plataforma física é virtualizada, como: CPU, memória e disco. No caso da paravirtualização, as máquinas virtuais podem se comunicar diretamente com o hardware sem passar pelo hospedeiro, ganhando um maior desempenho e maior performance em I/O do que as máquinas completamente virtualizadas, por outro lado a paravirtualização requer sistemas operacionais hóspedes modificáveis, que praticamente elimina o sistema operacional Windows da Microsoft.

3.2.2 Vantagens da Virtualização

Virtualizar vários sistemas operacionais e aplicativos em apenas um servidor pode trazer uma série de vantagens como agilidade, flexibilidade, melhor dimensionamento da infraestrutura de TI, ao mesmo tempo que permite uma economia significativa de ener-

gia, espaço físico e hardware. A implantação de servidores é mais rápida, a disponibilidade é maior e as operações se tornam automatizadas. As principais razões para se virtualizar um datacenter são (VMWARE, 2019): Redução dos custos operacionais e de capital; Alta disponibilidade de aplicativos; Minimizar ou eliminar o tempo de inatividade: Agilizar e simplificar o fornecimento de aplicativos e recursos; Aumentar a capacidade de resposta de TI; Suporte a continuidade de negócios e recuperação de desastres; Possibilita um gerenciamento centralizado;

No entanto, é importante notar que, embora existam várias vantagens, a virtualização não pode ser interpretada como uma solução para tudo. Mesmo atendendo a uma série de necessidades a virtualização, dependendo da situação, pode ter algumas desvantagens. Uma das principais desvantagens seria a sobrecarga, pois se o servidor for mal dimensionado irá causar lentidão em todas as máquinas virtuais que estão no mesmo hardware. Por sua vez, a segurança no sistema operacional hospedeiro é de extrema importância, pois uma vulnerabilidade de segurança no servidor hospedeiro pode afetar todas as máquinas virtuais. Outro fator seria o desempenho, pois a virtualização pode não ter um bom desempenho para todas as aplicações.

3.3 Nuvem Computacional

A computação em nuvem surgiu com o intuito de funcionar como um serviço, semelhante ao de telefonia, por exemplo, onde o usuário não precisa se preocupar com a infraestrutura em si. O objetivo era oferecer acesso a software, plataformas e infraestrutura como serviços disponíveis na Internet.

Com o passar dos anos veio o aumento da capacidade computacional e a capacidade de prover uma infraestrutura de computação como serviço, que surgiu pela evolução e amadurecimento do conceito de virtualização, isso possibilita um alto grau de portabilidade e flexibilidade para aplicações e sistemas operacionais diferentes. Usando técnica de virtualização é possível criar e definir redes de interconexão e sistemas de armazenamento virtuais, ou seja, na prática a computação em nuvem é o uso massivo da virtualização. Dessa forma a computação em nuvem se tornou um modelo de negócio, onde o usuário paga apenas o que consome e o provedor mantém a infraestrutura física amortizando os custos de manutenção e investimentos com o aluguel de seus recursos a diferentes usuários. Esse uso da virtualização e dos recursos por demanda permite que o usuário solicite, e pague, por mais recursos, caso necessite. Da forma semelhante, ele pode devolver os recursos que não precisa e deixar de pagar por eles (CARISSIMI, 2015).

Utilizando a computação em nuvem é possível migrar as máquinas virtuais de máquinas físicas pouco utilizadas para outras máquinas físicas, balanceando a carga

e desligando as de baixa utilização. Tais ações reduzem os custos de energia elétrica, refrigeração, suporte e manutenção a vários sistemas. A economia de energia gerada pela virtualização e pela computação em nuvem, faz com que elas sejam mecanismos para atingir o que se denomina computação verde (green computing).

Para identificar claramente o que é a computação em nuvem o National Institute of Standards and Technology (NIST) define um modelo para computação em nuvem com três camadas: atributos de serviços, modelo de serviços e modelos de implementação. A figura 3.3 ilustra como são as camadas da computação em nuvem (CARISSIMI, 2015).

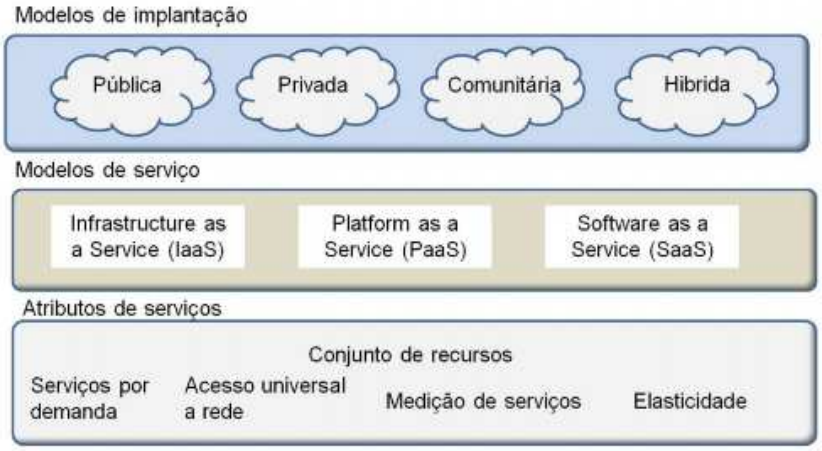


Figura 3.3 – Modelo NIST de computação em nuvem.

3.3.1 Atributos de Serviço

O NIST estabeleceu cinco características essenciais que a computação em nuvem deve ter: serviços por demanda, acesso universal, conjunto de recursos, contabilização de recursos e a elasticidade (CARISSIMI, 2015).

Serviços por demanda é a capacidade de alocar recursos conforme a necessidade de maneira automática;

Acesso universal da nuvem se dá com a capacidade de estar disponível em diferentes plataformas e sistemas operacionais;

Conjunto de recursos, são recursos fornecidos pelo provedor de forma que atendam a vários clientes simultaneamente, onde os recursos físicos e virtuais são alocados e liberados de forma dinâmica;

Contabilização de recursos em um sistema de computação em nuvem é a capacidade de medir e disponibilizar relatórios de uso a seus clientes de acordo com alguma métrica estabelecida. Além disso, o sistema deve ter a possibilidade de ter o uso, o consumo e a tarifação de recursos auditados;

Elasticidade é a técnica utilizada para alocar e liberar recursos conforme a necessidade do cliente. Sendo essa uma das características mais marcantes da computação em nuvem, pois com essa técnica é possível alocar e retirar recursos computacionais conforme a necessidade do cliente.

Na prática, trata-se de um balanceador de cargas que gera instâncias de máquinas virtuais conforme é necessário distribuir o processamento e assim que o processamento termina ou diminui essas instâncias são desfeitas. Para o usuário final isso funciona de forma transparente e serve para economizar custos e recursos, pois as máquinas são criadas e utilizadas somente em momentos de necessidade. (RIGHI, 2013)

3.3.2 Modelos de Serviço

Existem três modelos de serviços mais comuns definidos na computação em nuvem: Software as a Service (SaaS), Platform as a Service (PaaS) e a Infrastructure as a Service (IaaS).

SaaS: Nesse tipo de serviço o usuário final basicamente utiliza a aplicação já funcional, sem a necessidade de aquisição, configuração e manutenção. Um exemplo prático do uso do SaaS é o Google Docs, que permite ao usuário a criação de textos, planilhas e slides sem que ele se preocupe com a instalação de um software, armazenamento ou backup dos arquivos. Na verdade, o usuário final não sabe qual a configuração de hardware que está sendo utilizada ou em quantos servidores a aplicação está rodando, basta ter um navegador compatível com a aplicação e tudo funcionará.

PaaS: No modelo PaaS os clientes finais são desenvolvedores, pois nesse modelo eles obtêm um ambiente computacional completo com compiladores e bibliotecas, pronto para o desenvolvimento de aplicações. Um exemplo é a plataforma IBM Bluemix.

IaaS: É o modelo de serviço dedicado a uma equipe de tecnologia da informação (TI). Nesse modelo o analista escolhe a quantidade de processadores, memória e armazenamento e o sistema operacional, ficando com a responsabilidade de instalar e configurar o que necessita. Alguns exemplos de provedores IaaS são o Windows Azure, Amazon Elastic Compute Cloud (EC2), Citrix e a Eucalyptus.

3.3.3 Modelos de Implantação

Seguindo as definições de nuvem, o NIST define quatro modelos de implementação de nuvem computacional: pública, privativa, comunitária e híbrida (CARISSIMI, 2015).

Nuvem Pública: Nesse modelo de implementação as empresas fornecem recur-

sos que podem ser provisionados através da Internet, alocando apenas o necessário ao cliente. Normalmente as nuvens públicas oferecem os três modelos de serviços, IaaS, PaaS e SaaS. Alguns fornecedores desse tipo de nuvem são o Google, Windows Azure, Dualtec, entre outras.

Nuvem Privada: Nesse modelo a empresa constrói uma nuvem que somente ela teria acesso, contudo ela arca com todos os gastos de implementação, infraestrutura física, energia, refrigeração e gerenciamento. Esse tipo de nuvem traz a vantagem da alta disponibilidade, porém não se enquadra na categoria "paga pelo que usa" e também não há muitas vantagens com o uso da elasticidade, pois há o limite da infraestrutura física e para ter uma grande elasticidade a empresa precisaria comprar mais recursos físicos. Nuvem Comunitária: As nuvens comunitárias são aquelas onde organismos, departamentos ou empresas, compartilham a mesma infraestrutura física. O serviço mais comum encontrado em nuvens comunitárias é o SaaS, na forma de portais web. Nuvem Híbrida: Nuvens híbridas são nuvens compostas por duas ou mais nuvens de tipos diferentes conectadas entre si. A figura 3.4 representa a arquitetura em camadas da computação em nuvem (CARISSIMI, 2015).

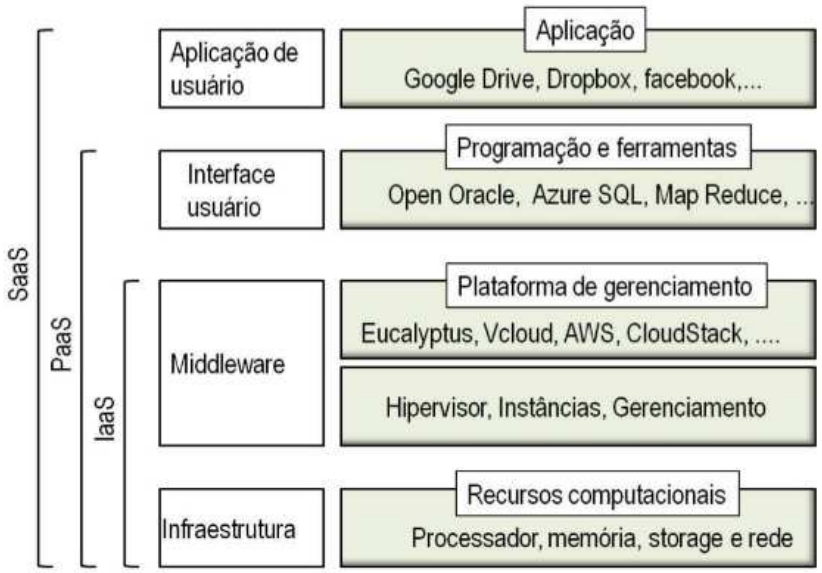


Figura 3.4 – Arquitetura em camadas.

4 Trabalhos Relacionados

O seguinte capítulo apresenta os trabalhos que foram realizados na área desta dissertação, fornecendo mais detalhes técnicos para auxiliar no desenvolvimento da pesquisa.

4.1 Performance Evaluation of Container Based Virtualization for HPC Computing Environments

Neste artigo os autores (XAVIER, 2013) comentam que não é comum a utilização de virtualização para ambientes de computação de alto desempenho e que isso se deve ao fato dos hypervisors criarem uma sobrecarga, especialmente em termos de I/O (XAVIER, 2013). Desse modo, o autor propõe a implementação de ambientes HPC com a utilização de `containers` como `Linux-VServer, OpenVZ e Linux Containers (LXC)`. No trabalho é utilizado o `benchmark NAS Parallel Benchmarks (NPB)` para fazer os testes de performance e o `Isolation Benchmark Suite (IBS)` para fazer os testes de isolamento e segurança.

Durante os experimentos foram criados ambientes de contêineres utilizando Linux-VServer, OpenVZ e Linux Containers (LXC) e para comparação foi criado um ambiente utilizando o hypervisor Xen. Foram utilizados quatro servidores com a seguinte configuração: Dell PowerEdge R610 com dois processadores 2.27GHz Intel Xeon E5520 (de 8 núcleos cada), 8M de L3 cache por núcleo, 16GB de RAM e um NetXtreme II BCM5709 Gigabit Ethernet adapter. Todos os nós estavam interconectados por um switch Dell PowerConnect 5548 Ethernet. O sistema operacional utilizado foi o Ubuntu 10.04 LTS (Lucid Lynx). Os resultados mostram que os ambientes com containers obtiveram uma performance similar ao ambiente nativo, não existindo diferenças estatísticas significantes. Entretanto, no caso do Xen, houve uma sobrecarga de aproximadamente 4,3%.

A respeito da performance da memória constatou-se que os containers usam a memória de uma melhor forma não resultando em diferenças com o sistema nativo. Por outro lado o Xen demonstrou uma diferença significante no uso da memória apresentando uma sobrecarga de aproximadamente 31% em comparação com o sistema nativo.

Referente à performance na utilização dos discos, os containers LXC e Linux-VServer obtiveram resultados semelhantes no quesito leitura, ambos com resultados próximos ao sistema nativo. Entretanto para a escrita foi observado um pequeno ganho de performance no VServer, uma perda de performance no container OpenVZ e novamente

o pior resultado ficou com o hypervisor Xen.

Quanto ao uso da rede, o melhor container foi o Linux-VServer, que obteve um comportamento similar ao nativo, seguido do LXC e pelo OpenVZ. No Xen a média de uso da banda foi de 41% menor que o ambiente nativo.

Finalmente, ao utulizar apenas um nó, em todos os casos os containers apresentaram um desempenho semelhante ao ambiente nativo, contudo o OpenVZ obteve o pior desempenho e o Xen obteve um resultado bom quanto ao uso intensivo de CPU. Nos testes com utilização de vários nós as diferenças se tornaram mais evidentes e o Xen ficou com o pior desempenho devido ao uso da rede.

4.2 Performance Evaluation of Hypervisors for HPC Applications

Neste trabalho os autores compararam o desempenho de dois hypervisors, comparando também as abordagens utilizadas na virtualização, ou seja, comparando a abordagem de virtualização completa e de paravirtualização (BESERRA, 2016). O objetivo era descobrir qual hypervisor possui a melhor performance para fazer a virtualização com foco em computação de alto desempenho.

O ambiente de pesquisa foi constituído por oito computadores HP Compaq 6005 com processadores AMD Athlon II X2 220 operando em 2.8 GHz. Todos os computadores também possuíam 8 GB DDR3 RAM de 1066 MHz. A rede utilizada para interconexão foi Gigabit Ethernet network adapters e switches. O ambiente nativo utilizado para a comparação foi construído utilizando o Rocks Cluster 6.1 64bits OS (Sistema baseado em Linux OS desenvolvido para facilitar a configuração de clusters de computação de alto desempenho). E todos os nós foram configurados com CentOS 6.6 64bits.

O autor adotou uma metodologia que baseia a análise em cinco pontos principais, onde a primeira era fazer a análise da arquitetura dos hypervisors tanto KVM quanto Virtualbox. Em seguida na segunda etapa foi necessário estabelecer e planejar as métricas que seriam medidas. Nessa etapa também foi definido qual `benchmark` seria utilizado e como seria o armazenamento dos dados coletados. Na terceira etapa foi feita a coleta dos dados e medidas. Em seguida, na quarta etapa, foram feitos os tratamentos estatísticos e verificação de ajustes e erros. Por fim, na quinta etapa, foi feita a análise de performance. Para mensurar e comparar os ambientes foi utilizado o `High Performance Computing Challenge Benchmark (HPCC)`. Nos testes foram coletadas informações de performance de CPU, memória e rede. Em todos os casos o KVM se mostrou melhor para computação de alto desempenho.

4.3 Advanced Virtualization Techniques for High Performance Cloud Cyberinfrastructure

Neste trabalho os autores propõem a analisar pontos chave na virtualização para computação de alto desempenho, visando minimizar a sobrecarga causada pelo hypervisor (YOUNGE ANDREW J.; FOX, 2014). Primeiramente foram analisados os hypervisors e a viabilidade de uma implementação de computação de alto desempenho, analisando também a utilização de GPU para melhorar o desempenho dos cálculos científicos. Além disso, também foi testado o uso das máquinas virtuais na rede de alta transmissão, a rede infiniband. Para fazer essa comparação o autor utilizou a plataforma de nuvem OpenStack.

O trabalho discute as melhores práticas para a construção de um ambiente IaaS com a melhor performance possível. Para realizar os teste de performance o autor utilizou casos de uso reais no ambiente.

Os testes foram realizados utilizando Xen, KVM e Virtualbox em comparação com o ambiente nativo. Os primeiros resultados mostraram que o hypervisor com maior eficiência foi o KVM e que o Xen possui a maior variância entre os testes. Em seguida os testes com utilização de MPI e FFT mostraram que o ambiente com KVM e Virtualbox ficam muito próximo do ambiente real enquanto que o Xen perde uma performance considerável.

Por fim, nos testes que utilizaram GPU o Xen se torna uma melhor opção com a performance equivalente a do ambiente nativo.

4.4 Evaluation of HPC Applications on Cloud

Neste trabalho os autores propõem que a nuvem pode ser viável para computação de alto desempenho dependendo das características da aplicação (GUPTA ABHISHEK; MILOJICIC, 2012). Os resultados revelaram que a nuvem é uma plataforma viável principalmente em casos que não existe muita comunicação entre os processos. Para fazer o experimento foram utilizados `benchmarks` NPB em 3 plataformas diferentes:

1. Taub que é um cluster físico utilizando rede infiniband e scientific Linux

2. Open Cirrus test-bed que é um cluster físico utilizando rede gigabit tradicional e Vanilla Linux

3. Um ambiente de nuvem utilizando o Eucalyptus e que utiliza KVM como hypervisor

Além dos `benchmarks` NPB, também foram utilizados softwares de uso real como o NAMD e o NQueens. Durante a apresentação dos resultados o autor mostra que o NAMD não escalona de forma adequada com a utilização do ambiente Open Cirrus e do Eucalyptus. Entretanto ao utilizar a aplicação NQueens o resultado é muito próximo ao do ambiente físico com uma pequena queda de performance ao utilizar 128 núcleos e uma queda um pouco maior ao utilizar 256 núcleos.

4.5 Exploring the Performance Impact of Virtualization on an HPC Cloud

Neste trabalho os autores fazem testes de performance para verificar a viabilidade de um ambiente de nuvem para o uso em computação de alto desempenho (CHAKTHRANONT, 2014), utilizando máquinas físicas e virtuais para verificar o impacto da virtualização na performance dos cálculos. Para realizar os experimentos a equipe utilizou um cluster composto por 155 nós sendo que cada um deles possui dois processadores Intel Xeon E5-2680 v2 (Ivy Bridge EN) 2.8 GHz com Hyper threading desabilitado, 128 GB de memoria RAM (DDR3-1866), 600GB de armazenamento utilizando SSD, rede Mellanox ConnectX-3 FDR InfiniBand HCA, e um 10 Gigabit Ethernet NIC.

Para construir o ambiente de cloud foi utilizado Apache CloudStack e KVM como hypervisor. Para avaliar o impacto da virtualização foram utilizados `benchmarks` tradicionais e rotinas de trabalho de computação de alto desempenho. Nos testes com MPI foi constatado que a sobrecarga da virtualização tende a diminuir se o tamanho da mensagem for um pouco maior. Na média uma mensagem com 1 byte gera 25% de sobrecarga enquanto que uma mensagem de 1 MB gera uma sobrecarga menor que 3%. Entretanto os experimentos mostram que a virtualização dificulta o escalonamento do MPI.

4.6 Understanding the Performance and Potential of Cloud Computing for Scientific Applications

O objetivo deste trabalho foi avaliar a performance da nuvem pública Amazon bem como oferecer algum contexto para comparação contra uma solução de nuvem privada. Foram executados micro `benchmarks` no ambiente EC2 da Amazon AWS para avaliar a performance de rede, processador, memória e armazenamento. Esses resultados foram comparados com um ambiente de nuvem privada chamado FermiCloud que utiliza OpenNebula. Essa comparação visou verificar as vantagens e limitações da nuvem pública

para computação científica (SADOOGHI, 2015).

Quando comparados os dois ambientes percebeu-se que, embora tenham uma performance similar, houve uma grande diferença na velocidade e na latência da rede. Isso ocorreu devido ao fato da nuvem privada utilizar a rede infiniband enquanto que a rede do ambiente AWS era ethernet 10Gb. Nos resultados finais os autores comentam que a falta de rede infiniband na nuvem AWS gerou uma diferença bastante significativa e concluíram que a rede infiniband é de grande importância para computação de alto desempenho.

4.7 Performance Comparison Between Virtual Machines And Docker Containers

Containers surgiram como uma alternativa à virtualização, e neste trabalho os autores fazem uma análise de performance entre os contêineres e os hypervisors para avaliar o desempenho de ambos. Além da análise tradicional com `benchmarks` eles também fazem uma análise utilizando um caso de uso real. (YADAV, 2018)

Para a realização dos experimentos foram utilizados o hypervisor VMWare e o container Docker. As máquinas virtuais e containers utilizados possuíam processador Intel i3, 4GB de memória RAM e 50GB de disco. Além disso, todos estavam configurados com sistema operacional Ubuntu 16.04 LTS e utilizando o Sysbench como programa de benchmark. Na análise dos resultados foi constatado que ambas as tecnologias têm performance semelhante, com vantagem para o Docker Container, entretanto, foi observado que o aumento da carga de trabalho faz a taxa de consumo CPU e memória RAM ser mais acentuada no Docker Container.

4.8 Analysis of Virtualization Technologies for High Performance Computing Environments

O artigo faz uma análise entre diferentes tipos de hypervisors, com aplicabilidade em computação de alto desempenho para verificar as vantagens e desvantagens de cada um. Foi possível notar um leve impacto na performance, dependendo do tipo de hypervisor, e concluído que o KVM é uma ótima escolha para um ambiente de computação de alto desempenho virtualizado (YOUNGE, 2011).

O objetivo do trabalho foi comparar os diferentes hypervisors para verificar qual o melhor adaptado para computação de alto desempenho, dessa forma é possível implementar um ambiente de nuvem computacional tendo uma perda de performance

pequena, mas que gera vários benefícios como escalabilidade, qualidade de serviço, personalização dos ambientes, custo efetivo e interfaces com acesso simplificado.

Os experimentos foram feitos utilizando somente a arquitetura x86-64 no ambiente do cluster FutureGrid, utilizando os hypervisors Xen, KVM, Virtualbox e ambiente nativo. Os nós possuem a seguinte configuração: Processador Intel Xeon 5570 quad core com 2.93Ghz, 24 GBs de memória RAM e rede infiniband QDR, além disso todos os nós estavam configurados com Red Hat Enterprise Linux server 5.5 x86-64 e kernel 2.6.18-194.8.1.el5. Por sua vez, os `benchmarks` que foram utilizados foram o HPCC e o SPEC.

Como conclusão os autores dizem que é possível criar um ambiente de computação de alto desempenho virtualizado, pois escolhendo o hypervisor certo a perda de performance seria muito baixa. Nesse caso os hypervisors mais promissores foram o KVM e o XEN, pois o Virtualbox possui muitas limitações. A princípio o XEN se aparentava mais promissor que o KVM, pois ele é mais utilizado, especialmente na construção de nuvens computacionais acadêmicas, entretanto o KVM se mostrou mais performático e foi considerado melhor para a virtualização de ambientes de computação de alto desempenho, devido a sua melhor performance e maior estabilidade.

4.9 An updated performance comparison of virtual machines and Linux containers

A computação em nuvem faz uso intensivo da virtualização porque essa técnica permite que os processos e trabalhos sejam executados de forma isolada, onde os recursos podem ser utilizados por cada usuário individualmente. Neste trabalho, (FELTER, 2015), os autores exploram o poder de desempenho entre máquinas virtuais, ambientes nativos e containers com o objetivo de entender o overhead gerado em cada um. Todos os testes foram realizados em um servidor IBM System x3650M4 com dois processadores Intel Sandy Bridge-EP Xeon E5-2665 de 2.4GHz com um total de 16 cores (plus Hyper Threading) e 256 GB de RAM e sistema operacional Ubuntu 13.10 e kernel kernel 3.11.0.

Foi percebido que o contêiner e máquinas virtuais não geraram muito overhead no uso de CPUs e memória RAM, mas que geraram impacto no uso de I/O, leitura e escrita, e na interação com o sistema operacional.

A conclusão do trabalho é que ambas as tecnologias, virtualização e contêiner, são maduras e que possuem uma boa otimização para o uso de CPU e memória RAM, mas que geram intensivos overheads no uso de I/O, portanto, ambas as tecnologias devem ser usadas com cuidado.

4.10 Virtualization in HPC - An Enabler for Adaptive Co-Scheduling?

Neste capítulo do livro Co-Scheduling of HPC Applications, (PICKARTZ, 2017), os autores apresentam um estudo sobre o estado da arte da virtualização aplicada à computação de alto desempenho, que não só considera os aspectos de workloads, mas também discute aspectos qualitativos como o desenvolvimento da virtualização nesse contexto. Os resultados mostram que o desempenho de uma máquina virtual é muito próximo ao de uma máquina nativa, entretanto o ambiente precisa estar muito bem configurado.

O livro explica que com o uso dessa técnica é possível rodar processos que necessitam de drivers e bibliotecas distintas no mesmo nó, pois a virtualização traz alguns benefícios ao ambiente de computação. Os benefícios são: isolamento, consolidação e migração. Quanto ao isolamento, refere-se a possibilidade de diferentes processos ocuparem o mesmo nó e utilizarem drivers e bibliotecas distintas. Graças a isso é possível melhorar a segurança e a confiabilidade do ambiente onde o processo está sendo executado. Entretanto um problema que pode aparecer é a subutilização dos nós, pois o programa pode não utilizar todos os recursos disponíveis. Por sua vez, a consolidação segue a mesma ideia do isolamento, mas utiliza a técnica de co-scheduling e tem o objetivo de melhorar a utilização do hardware disponível. Finalmente, a migração permite uma melhor distribuição do trabalho por todo o cluster, servindo como um load balancing.

Neste trabalho, os autores também comentam a respeito do I/O, leitura e escrita, que ainda é um desafio para quem utiliza computação de alto desempenho, pois contornar a camada do sistema operacional não é uma opção performática. Além disso, ele explica que essa é uma das razões que não deixaram a virtualização ter um papel mais importante no passado para a computação de alto desempenho. Para resolver esse desafio o livro aborda três técnicas: Emulação do dispositivo feito pelo virtual switch, device pass-through ou usar o Single Root I/O Virtualization (SR-IOV). A emulação do dispositivo iria requerer a virtualização completa, ou seja, cada sistema convidado iria fazer requisições custosas para o gerenciador de máquinas virtuais. Por um lado essa abordagem gera um impacto significativo na performance, mas, por outro lado, a migração de máquinas virtuais de um host a outro se torna muito mais fácil, pois essa abordagem permite ao gerenciador ter controle completo sobre tráfego entre o sistema convidado e o dispositivo. Uma alternativa à emulação dos dispositivos seria aplicar técnicas de paravirtualização e com a utilização de drivers otimizados é possível diminuir o custo computacional entre o sistema hospedeiro e convidado. Entretanto essa abordagem requer uma configuração especial para cada sistema convidado.

Com a utilização da atribuição direta, ou pass-through, é possível atingir um

desempenho comparável ao desempenho nativo e isso é possível graças a tecnologia de virtualização direta da Intel (Intel VT-d) que permite o acesso direto à memória (DMA). Contudo essa técnica é limitada a um sistema convidado por vez em cada dispositivo.

Por fim, o SR-IOV permite o uso de duas funções novas, as funções físicas e funções virtuais, onde múltiplas funções virtuais podem responder a uma função física. Mesmo assim as técnicas de pass-through e SR-IOV sofrem um gargalo quando for necessário fazer uma migração. Os testes foram feitos com a utilização do `NAS Parallel Benchmarks` e cada cálculo foi executado de forma exclusiva e também `co-scheduling`.

Em sua conclusão os autores reconhecem que a virtualização completa pode chegar a resultados próximos aos resultados de desempenho de programas executados em máquinas nativas, tanto com a utilização de `co-scheduling` ou job exclusivo. Além disso, foi estimado o impacto do tamanho das máquinas virtuais no desempenho do cálculo e concluído que a granularidade da máquina virtual impacta diretamente no uso de memória compartilhada e na comunicação de rede. Sendo assim, o uso das máquinas virtuais depende das características de cada aplicação.

4.11 Function Virtualization in High Performance Computing: Opportunities and Challenges

O artigo menciona que a computação de alto desempenho é um ponto chave no desenvolvimento da China, pois ela pode ajudar no desenvolvimento de diversas áreas. Neste artigo, os autores discutem o desenvolvimento do HPC e suas possíveis virtualizações no futuro, também é abordado as vantagens e desvantagens da virtualização para HPC, onde o foco principal do artigo é discutir os avanços e desafios do HPC (JIANG YUJUAN; XIANGYANG, 2018).

Neste artigo são listados os desafios de segurança e desafios de usabilidade, mas conclui-se que a virtualização é o futuro da computação de alto desempenho, que fará uso total dos recursos de computação e reduzirá a complexidade da transformação do código.

4.12 Cloud Platform Optimization for HPC

A maioria das nuvens não estão totalmente otimizadas para HPC ou ainda não estão completas com os recursos no que diz respeito à experiência de supercomputação tradicional. Alguns dos principais gargalos estão nas áreas que visam minimizar e eliminar o impacto da camada de virtualização (hypervisor); representação semelhante ao ambiente

nativo para a carga de trabalho; e o ecossistema de software HPC. Enquanto o terceiro item se preocupa mais com a prontidão e usabilidade do ambiente, os dois primeiros itens impactam diretamente o desempenho das cargas de trabalho HPC (VERMA, 2020).

Como é muito interessante imaginar um ambiente de nuvem para HPC, os autores deste trabalho discutem os esforços feitos para minimizar e eliminar o impacto da virtualização em cargas de trabalho HPC na infraestrutura em nuvem e avançar em direção a uma melhor experiência com supercomputação.

O artigo tem como base de infraestrutura a plataforma de nuvem Microsoft Azure, utilizando Windows Server, o Hyper-V como virtualizador e imagens de CentOS 7 para os sistemas virtualizados. Nele são listados alguns métodos que podem otimizar o uso de HPC em nuvem, onde alguns métodos descritos visam eliminar o atraso na entrega de dados e melhorar os métodos de virtualização. Em sua conclusão, o artigo aborda que as otimizações feitas visando cargas de trabalho tradicionais de HPC também podem ser aproveitadas para demandas de big data e AI. Além disso, é abordado o desafio de conseguir alinhar as demandas de velocidade com o armazenamento.

4.13 An Efficient Implementation of GPU Virtualization in High Performance Clusters

Neste artigo é apresentado um protótipo de middleware que virtualiza um recurso de hardware como uma GPU em um cluster HPC. Este middleware provê uma replicação completa de toda a GPU para que toda ela seja emulada em um outro nó. No entanto, para aplicativos de computação intensiva, essa abordagem não é válida devido à sobrecarga de emulação. Os autores dizem que a melhor opção para atender aos aplicativos HPC é oferecer uma plataforma de hardware virtualizada, compartilhando o tempo real entre os usuários. Este middleware possibilita a execução de diferentes partes de um programa em diferentes GPUs dinamicamente alocadas (DUATO JOSé, 2010).

O middleware de virtualização de GPU disponibiliza GPUs remotas compatíveis com CUDA para todos os nós do cluster. O software é implementado em cima da interface de programação do aplicativo sockets, garantindo portabilidade em redes commodity, mas também pode ser facilmente adaptado para redes de alto desempenho.

O objetivo do programa é compartilhar ao máximo o uso de GPUs visto que existem restrições econômicas e/ou relacionadas à energia, pois em certos casos não é viável fornecer um coprocessador acelerador, como um processador gráfico (GPU) por nó.

Finalmente, o artigo mostra que essa abordagem pode fornecer um desempenho razoável para clusters conectados por meio de uma rede comum. Como a maior parte

do tempo é gasto em comunicações, é esperado que a degradação do desempenho seja significativamente menor no caso os nós sejam conectados por meio de uma rede de alto desempenho.

4.14 On Implementation of GPU Virtualization Using PCI Pass-Through

O foco deste artigo é na virtualização de GPU e a implementação de um sistema com ambiente de virtualização e usa o PCI pass-through, uma tecnologia que permite que as máquinas virtuais no sistema usem GPU para aumentar o poder de computação. (YANG, 2012) Neste artigo, os autores usam a tecnologia de PCI Pass-Through para fazer com que as máquinas virtuais em um ambiente virtual sejam capazes de usar uma GPU, que usa o CUDA para programação paralela. Isso faz com que a máquina virtual não tenha apenas a CPU virtual, mas também a GPU real para computação. O artigo mede o desempenho das diferenças entre máquinas virtuais utilizando hypervisor Xen e máquinas físicas usando CUDA.

Em sua conclusão, o trabalho mostra que o desempenho da GPU é o mesmo na máquina nativa e virtual, independente de quantas CPUs existem na máquina virtual, a GPU fornece o mesmo desempenho por passagem PCI.

4.15 Tabela Comparativa Entre os Trabalhos Estudados

Tabela 4.1 – Trabalhos Estudados.

Artigo	Área	Abordagem	Problemas	Utilidade
(XAVIER, 2013)	Virtualização e performance	`Benchmark` de ambientes HPC com virtualização baseada em conteineres	Compara três softwares de contêiner com apenas um hypervisor	Comparação no desempenho de um ambiente virtualizado com XenServer em contraste com um ambiente em container

4.1				
Artigo	Área	Abordagem	Problemas	Utilidade
(BESERRA, 2016)	Virtualização e performance	Performance entre paravirtualização e virtualização completa com foco em HPC	Faz teste utilizando o Virtualbox, hypervisor que não é utilizado para aplicações de alta escala	Faz testes com hypervisors e descreve qual o melhor
(YOUNGE ANDREW J.; FOX, 2014)	Virtualização e performance	Análise de pontos chave na virtualização para HPC, visando minimizar a sobrecarga causada pelo hypervisor	Não encontrado	Faz `benchmarks` utilizando `GPU, infiniband e OpenStack`. Por fim propõe uma arquitetura para atingir a melhor performance possível
(GUPTA ABHISHEK; MILOJICIC, 2012)	Performance na nuvem	Testa performance de ambientes HPC na nuvem	Teste de apenas um ambiente de nuvem com um hypervisor em comparação com o ambiente nativo	Faz testes de performance com softwares de uso reais da dinâmica molecular

4.1				
Artigo	Área	Abordagem	Problemas	Utilidade
(CHAKTHRANONT, 2014)	Performance na nuvem	Testa performance de ambientes HPC na nuvem sem hyperthreading	Teste de apenas um ambiente de nuvem com um hypervisor em comparação com o ambiente nativo	Faz testes de performance com softwares de `benchmark`
(SADOOGHI, 2015)	Performance na nuvem	Faz testes de performance comparando a nuvem pública AWS e nuvem privada com OpenNebula	Apenas uma comparação entre duas nuvens sem testar os vários tipos de hypervisor	Conclui que a nuvem privada com OpenNebula é melhor devido a rede `infiniband`
(FELTER, 2015)	Virtualização e performance	Compara o desempenho de vários hypervisors com a utilização de `benchmarks HPCC e o SPEC`	Não foi utilizado testes com problemas reais	Compara diferentes tipos de hypervisor e define qual o melhor para um ambiente HPC
(PICKARTZ, 2017)	Virtualização e performance	Apresenta o estado da arte na aplicação da virtualização para HPC	Não encontrado	Exemplifica formas de migração para uso de virtualização em HPC com `co-scheduling`

Este trabalho propõe a verificação da performance de CPU, memória e disco comparando ambientes virtualizados com KVM e XenServer em comparação com o ambiente nativo. Os testes são feitos com a utilização do software `Sysbench` e o `dd`, além disso, o diferencial deste trabalho em comparação aos trabalhos analisados é que também são feitos estudos de caso com aplicações reais utilizadas na pesquisa em dinâmica molecular.

5 Proposta e Desenvolvimento

Este capíitulo apresenta os detalhes da proposta do projeto de mestrado. Ele é baseado nos artigos estudados e apresentados nos capítulos anteriores.

A proposta deste projeto é examinar as caracteristicas da virtualização e seus impactos de desempenho em ambientes de computação de alto desempenho. Para isso, quatro máquinas virtuais distintas foram configuradas, utilizando os `hypervisors` gratuitos mais utilizados, sendo eles o XenServer e o KVM, além do ambiente nativo. Os testes realizados nesses ambientes são comparados aos testes executados em ambiente nativo, ou seja, sem virtualização. Uma vez concluída as analises será possível dizer qual `hypervisor` é mais apropriado para a construção de uma nuvem computacional para computação de alto desempenho e que seja capaz de atender as demandas das aplicações.

5.1 Metodologia

Para a construção dos `benchmarks` são utilizados os softwares `sysbench`, `dd` e `hdparm`, cujo o objetivo é gerar uma medição de CPU, memória RAM e leitura/escrita de disco. Além disso, os `benchmarks` são feitos com granularidades diferentes de máquinas virtuais, pois, segundo (PICKARTZ, 2017), existe uma diferença de performance entre elas quando a granularidade da máquina virtual é diferente. Nesse caso, a granularidade significa a quantidade de recursos computacionais que a máquina virtual está utilizando.

Sendo assim, são construídas quatro categorias de máquinas virtuais, onde a primeira possui as 40 `threads` disponíveis e 60 GB de memória. A segunda tem 20 `threads` e 30 GB de memória. A terceira tem 10 `threads` e 15 GB de memória e a quarta é a máquina virtual com menos recursos e possui 4 `threads` e 5 GB de memória. O disco de todas elas possui o armazenamento de 100 GB. A Tabela 5.1 apresenta as configurações de forma resumida.

Máquina Virtual	Threads	Memória	Disco
vm-1	40	50GB	100GB
vm-2	20	30GB	100GB
vm-3	10	15GB	100GB
vm-4	4	5GB	100GB

Tabela 5.1 – Configurações das Máquinas Virtuais.

As máquinas virtuais possuem estas configurações visando os seguintes objetivos: a vm-1 tem o objetivo de observar o comportamento das aplicações ao utilizar uma

máquina virtual que utiliza todo o recurso de computação do servidor hospedeiro; a vm-2, é utilizada para verificar o comportamento das aplicações quando duas máquinas virtuais dividem todo o recurso computacional; a vm-3 mostra como as aplicações se comportam dividindo os recursos computacionais entre duas máquinas virtuais e sem utilizar o total disponível; por fim, são utilizadas cinco máquinas virtuais vm-4 simultaneamente no `benchmark` de uso de CPUs visando ocupar todos os recursos de processamento e, dessa forma, verificar se o desempenho das máquinas virtuais é impactado pelo `hypervisor` quando existem várias máquinas virtuais para serem gerenciadas ao mesmo tempo.

O `hardware` utilizado para a montagem dos ambientes possui a seguinte configuração: 2 processadores Intel Xeon E5-2670 v2 de 2.50GHz de 20 núcleos físicos e 40 `threads` com `Hyperthreading`; 64 GB de memória RAM DDR4 2133 MHz (8 x 8 GB); 1 disco SATA 1000 GB 7200 RPM e conexão de rede Gigabit Ethernet. Além de ser configurado com sistema operacional Suse Linux Enterprise Server 12.1, que é o mesmo sistema operacional das máquinas virtuais, dessa forma, o desempenho é verificado entre sistemas operacionais iguais.

Todos os `benchmarks` são executados 30 vezes, em seguida, são calculados as médias de desempenho e o intervalo de confiança de 95% dos resultados obtidos. A variação dos resultados é avaliada com a utilização do gráfico de caixas ou `boxplot`, que permite verificar se houve alguma anomalia. Além disso, é avaliado o coeficiente de variação que é uma medida padronizada de dispersão da distribuição dos resultados, podendo assim certificar a variação em porcentagem.

5.2 Ferramentas utilizadas

5.2.1 XenServer

O Xenserver é uma plataforma de virtualização para servidores empresariais, que oferece os recursos necessários para qualquer implementação de virtualização de servidores e datacenters (XENSERVER, 2019). Trata-se de um hypervisor que pode ser instalado diretamente no servidor, sem a configuração de um sistema operacional hospedeiro. Essa arquitetura visa garantir mais velocidade e desempenho para as aplicações. Além disso, a arquitetura de gerenciamento do XenServer distribui os dados de gerenciamento em todos os servidores do conjunto de recursos para garantir que não haja um único ponto de falha no gerenciamento. (BARHAM, 2003)

Por sua vez, o XenCenter é uma ferramenta que acompanha o Xenserver. Trata-se de uma ferramenta de administração que fornece uma interface para auxiliar na gestão das máquinas virtuais, na monitoração e na administração geral. Para a construção

dos `benchmarks` propostos é utilizado o XenServer 7.6 com máquinas virtuais Suse Linux Enterprise Server 12.1.

5.2.2 KVM e QEMU

O KVM é um hypervisor de código aberto que faz virtualização completa e utiliza Linux. Ele é voltado para servidores x86 que possuem as instruções Intel VT or AMD-V. Trata-se de um módulo carregável do kernel, o `kvm.ko`, que fornece a infraestrutura de virtualização e um módulo específico do processador, `kvm-intel.ko` ou `kvm-amd.ko` (KVM, 2019). Com o KVM, pode-se executar várias máquinas virtuais com imagens Linux ou Windows. Cada máquina virtual possui `hardware` privado virtualizado, ou seja, possui seu próprio processador, placa de rede e todos os outros componentes. O KVM é um software de código aberto, onde o componente kernel do KVM está incluído no Linux. Os ambientes criados usam o QEMU KVM 2.3.1.

5.2.3 SysBench

`SysBench` é um software multi plataforma e `multi-threaded` que é utilizado para testar a performance de sistemas operacionais com uma carga intensiva de trabalho. A proposta dessa ferramenta é fornecer uma avaliação rápida da performance dos sistemas, sem a necessidade de configurações complexas. Seu `design` é muito simples, onde é possível limitar a quantidade de requisições o tempo ou ambos, (ORACLE, 2020). Neste trabalho a limitação é apenas na quantidade de requisições e quantidade de CPUs a partir desses parâmetros é possível verificar a performance de execução dos CPUs dos ambientes propostos.

5.2.4 dd - Data Duplicator

Trata-se de um software do Linux cuja a função é duplicar dados. Este software é muito útil para criar réplicas de discos e nesse trabalho ele é utilizado para verificar a performance de leitura e escrita do disco e também utilizado para verificar a performance de memória RAM, pois trata-se de um software simples, muito eficaz e que faz a réplica bit a bit. (RUBIN PAUL; MACKENZIE, 2019)

5.2.5 hdparm

Este utilitário, desenvolvido por Mark Lord, (LORD, 2020; KERRISK, 2018) é capaz de fornecer dados de performance e diagnóstico de funcionamento do disco, além de oferecer a possibilidade de visualização e alteração de parâmetros do disco ou até mesmo apagar o disco de forma segura. Neste trabalho o utilitário é usado para verificar

os parâmetros do disco e para ajustar os parâmetros que se encontram diferentes no momento do `benchmark`.

5.2.6 Softwares de Cálculos Científicos

Neste trabalho são utilizados softwares científicos para verificação de desempenho visando fazer estudos de caso com softwares utilizados na área da físico-química. Os softwares utilizados são NAMD, Gromacs, Amber, Lammps, Gaussian e Gamess. O software NAMD é utilizado para realizar testes reais de cálculos de dinâmica molecular. O NAMD é desenvolvido pelo grupo `Theoretical and Computational Biophysics Group` no Instituto Beckman de Ciência Avançada e tecnologia da Universidade de Illinois em Urbana e Champaign (PHILLIPS, 2005). Gromacs (GROningen MAchine for Chemical Simulation), trata-se de um software de dinâmica molecular utilizado para cálculos de moléculas utilizando física clássica (BERENDSEN, 1994). Amber é um conjunto de programas de simulação biomolecular. O software começou no final dos anos 1970 e é mantido por uma comunidade de desenvolvimento ativa (CASE, 2020).

O Lammps é um software que começou a ser desenvolvido em meados de 1990 sob um acordo cooperativo de pesquisa e desenvolvimento (CRADA) entre dois laboratórios DOE (Sandia e LLNL) e 3 empresas (Cray, Bristol Myers Squibb e Dupont). O objetivo era desenvolver um código de Dinâmica Molecular clássico paralelo em grande escala (PLIMPTON, 1995). Por sua vez, Gaussian é um pacote de software de química computacional de uso geral lançado inicialmente em 1970 e desenvolvido por John Pople e seu grupo de pesquisa na Universidade Carnegie Mellon (AL., 2016). Finalmente, o Gamess `Gamess (General Atomic and Molecular Electronic Structure System)` um software de processamento paralelo utilizado para pesquisas de dinâmica quântica (SCHMIDT, 1993).

5.2.7 Uso de GPU nos benchmarks

Neste trabalho não foi utilizado GPU em seus experimentos devido a alguns problemas na virtualização da GPU, pois não foi possível habilitá-las para uso nas máquinas virtuais. Foram tentadas duas técnicas para habilitar a virtualização da GPU, a técnica de PCI Pass-Through e a técnica de utilização de GPU remota através do middleware rCUDA. Quanto ao PCI Pass-Through, mesmo habilitando as instruções Intel VT-d do processador e placa-mãe, a GPU era disponibilizada, mas não funcionava quando adicionada ao hypervisor.

Sobre o middleware rCUDA, a instalação do rCUDA foi feita, entretanto não foi obtido sucesso na comunicação, pois o middleware não dá suporte a algumas aplicações

científicas utilizadas neste artigo. As aplicações que foram testadas conseguiram enxergar a GPU entretanto o cálculo ficou parado e nenhum resultado foi produzido (DUATO, 2020; PRADES JAVIER; REAñO, 2018). Após inúmeras tentativas, o uso de GPU foi deixado para uma pesquisa futura, pois seu uso de forma virtualizada ainda é um desafio.

5.2.8 Reproducibilidade dos Experimentos e dos Dados Obtidos

Todos os dados e códigos usados para os testes iniciais estão disponíveis de forma pública no repositório: https://github.com/adrianoferruzzi/vmdata/ Os diretórios incluem os scripts utilizados, dados obtidos e os gráficos gerados.

6 Avaliações e Resultados Obtidos

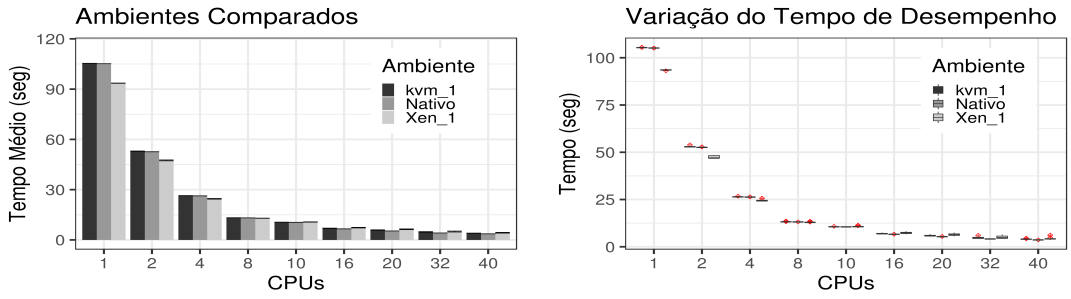
Neste capítulo os resultados são comparados e analisados de acordo com a arquitetura e com a performance atingida. Após uma análise detalhada do ocorrido é possível propor um `hypervisor` que tenha o melhor comportamento para a aplicações que utilizem majoriateriamente CPU.

6.1 CPU

O software `Sysbench` é utilizado em todos os ambientes para obter a estimativa de desempenho dos processadores. O parâmetro de comparação foi o tempo necessário, em segundos, para completar 100.000 eventos, onde cada evento é o resultado do cálculo para descoberta de números primos entre 0 e 10.000. Esses valores de eventos são o padrão do `Sysbench` e são suficientes para simular o comportamento de uma aplicação `CPU-bound`. Referente às CPUs, o parâmetro `turbo boost` foi desativado para garantir a mesma frequência dos processadores durante todos os experimentos. Além disso, neste experimento, o `cache` foi levado em consideração, mas os primeiros experimentos tiveram um tempo similar a dos demais, ou seja, o `cache` não impactou no resultado obtido nessas simulações com o `Sysbench`.

6.1.1 Benchmark de uso de procesador de forma exclusiva

Ao analisar os resultados é possível perceber que não houve uma diferença significativa entre as diferentes de máquinas virtuais, portanto os gráficos a seguir comparam o ambiente Nativo com uma máquina virtual de cada configuração. A Figura 6.1 apresenta a comparação entre o ambiente kvm-1, Xen-1 e nativo.



(a) Desempenho Comparado

(b) Variação no Boxplot

Figura 6.1 – Desempenho e Variação dos Resultados

Na Figura 6.1 é possível perceber que ao utilizar poucas `threads` o Xen obteve melhor desempenho enquanto que o KVM executa o benchmark sempre com uma performance próxima a do ambiente nativo e ao utilizar todas as `threads` disponíveis o ambiente nativo obteve melhor desempenho. Isso já era esperado, pois no ambiente nativo os processadores não precisam processar a camada de virtualização.

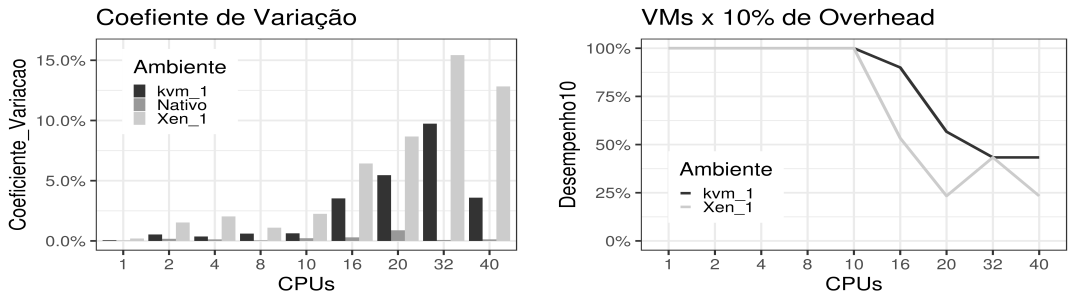
Após essas análises, o calculo de eficiência de processamento é feito entre 1 thread e 40 `threads`. Assim, pode-se verificar que o Xen-1 obteve uma eficiência de 54.58%, enquanto que o kvm-1 conseguiu 65.29%, ficando mais próxima ao resultado do ambiente nativo que foi 72.42%. Por outro lado, levando em consideração a média de desempenho final e o intervalo de confiança é possível notar que ambos os `hypervisors` obtiveram um resultado estatístico semelhante. O cálculo de eficiência é feito com a divisão do tempo de 40 `threads` pelo tempo de 1 `thread`, dessa forma, verifica-se quantas vezes mais rápido o cálculo é executado e em seguida esse valor é dividido pelo número total de `threads`, assim é possível saber a eficiência do proccessador.

Após o cálculo de eficiência de processamento o passo seguinte foi ver a dispersão dos dados obtidos que também estão apresentados na Figura 6.1. Com a análise do `boxplot` é possível identificar alguns `outliers`, ou anomalias. Essas anomalias são pontos fora da curva e representam os cálculos que não atingiram o desempenho esperado. Além disso, é possível ver o impacto do `hyperthreading` nos ambientes virtualizados, pois ao utilizar essa tecnologia foi observado que o coeficiente de variação sobe para mais de 15% no ambiente Xen-1 com 32 `threads`. Essa variação é causada pela distribuição entre núcleos físicos e virtuais.

Além disso, o cálculo de coeficiente de variação também é feito, visando expressar a variabilidade dos dados estatísticos excluindo a influência da ordem de grandeza da variável. Ele é utilizado para determinar quanto o resultado varia em porcentagem. Para chegar ao resultado do coeficiente de variação é preciso encontrar o desvio padrão e dividir pela média dos dados.

Ao analisar o coeficiente de variação de aproximadamente 15% ao utilizar 32 `threads`, na Figura 6.2, uma nova dúvida surgiu: Quantos processos de fato atingiram um desempenho que seja pelo menos 10% a mais que o tempo nativo? E após a criação de uma tabela com as comparações de tempos dos experimentos realizados foi possível verificar que o melhor `hypervisor`, ao utilizar 40 `threads`, foi o KVM. Ele atingiu em 45% das vezes um desempenho comparável ao ambiente nativo com mais 10% de overhead, conforme apresentado na Figura 6.2. Neste cálculo, o `overhead` utilizado para fazer a comparação é teórico, pois ele é resultado da soma do resultado nativo mais 10%.

O gráfico com o desempenho atingido mostra que ao utilizar o máximo do



(a) Coeficiente de Variação

(b) Desempenho Atingido com 10% de Overhead no Ambiente Nativo

Figura 6.2 – Coeficiente de Variação e Desempenho Atingido

nó, 40 `threads`, o tempo de processamento terá mais que 10% de `overhead` em pelo menos 55% das vezes e isso pode acontecer mesmo com o melhor `hypervisor` utilizado no trabalho.

É importante salientar que foi utilizado a tecnologia de `hyperthreading` nos benchmarks com máquinas virtuais. Portanto, nota-se um pequeno ganho de desempenho ao comparar os testes com 20 `threads` e 40 `threads` e esse fenômeno também pode ser observado nos resultados com 16 e 32 `threads`. Isso acontece porque o `hyperthreading` cria núcleos virtuais permitindo dobrar a quantidade de `threads` e, obviamente, os núcleos virtuais não têm o mesmo desempenho dos núcleos físicos, por isso o ganho de desempenho é pequeno, mesmo quando é usado o dobro de `threads`.

6.1.2 Benchmark de uso de processador de forma compartilhada

Na segunda etapa os benchmarks são feitos de forma simultânea, onde as máquinas virtuais estão hospedadas no mesmo hospedeiro e ambas executam os testes ao mesmo tempo. O objetivo foi verificar se existe alguma mudança de comportamento quando comparadas com as máquinas virtuais que executaram de forma exclusiva no hospedeiro.

A Figura 6.3 apresenta o resultado encontrado com cada tamanho de máquina virtual. Nesse caso, as máquinas virtuais kvm-2 e Xen-2 executam simultaneamente e ocupam todos os recursos. As kvm-3 e Xen-3 executam simultaneamente, mas não ocupam todos os recursos computacionais. Por sua vez, as kvm-4 e Xen-4 são as máquinas virtuais com menor tamanho e o objetivo é subir várias máquinas virtuais pequenas e fazer com que elas ocupem todo o hardware, dessa forma, pode-se verificar o impacto do tempo de gerenciamento do `hypervisor` quando existem muitas máquinas virtuais em execução. Ao analisar o gráfico é possível perceber que no geral o uso compartilhado não afeta a performance. A única exceção foi o resultado do Xen-4 que foi impactado pelo uso das

várias máquinas virtuais simultâneas, ou seja, o tempo nos resultados pode ser impactado quando existirem várias máquinas virtuais executando simultaneamente.

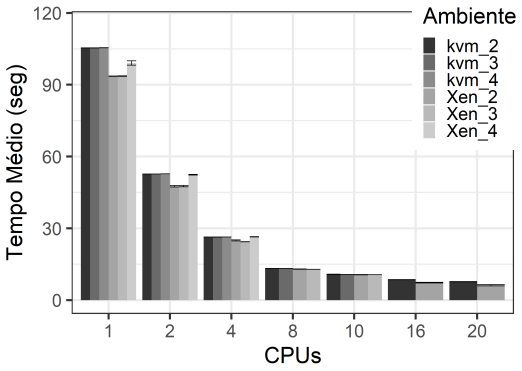
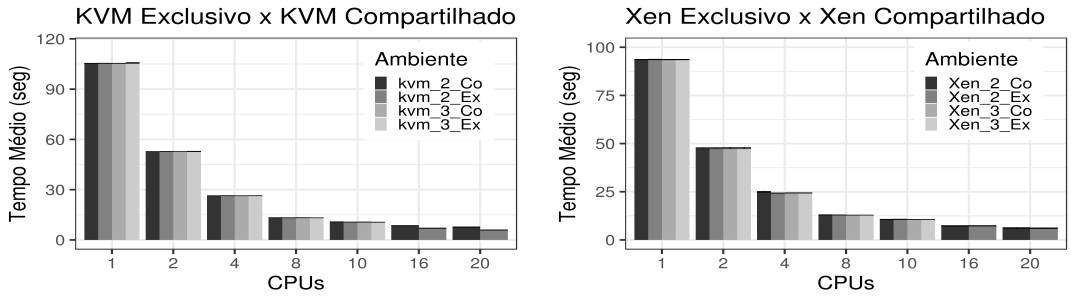


Figura 6.3 – Desempenho com uso Compartilhado

Com a análise da Figura 6.4 é possível perceber que o desempenho se mantem semelhante ao desempenho das máquinas virtuais que utilizaram os hospedeiros de forma exclusiva.



(a) KVM - Exclusivo x Compartilhado (b) Xen - Exclusivo x Compartilhado

Figura 6.4 – Comparação dos Resultados Atingidos

Nesses benchmarks o Xen se mostrou levemente mais performático que o KVM quando utiliza todos os recursos computacionais de forma compartilhada. Isso torna o Xen uma boa escolha caso seja utilizado em um ambiente onde os cálculos demandem majoritariamente processamento em ambiente compartilhado com poucas máquinas ao mesmo tempo. Por outro lado, o KVM é uma melhor escolha quando o objetivo for executar várias máquinas virtuais pequenas simultaneamente.

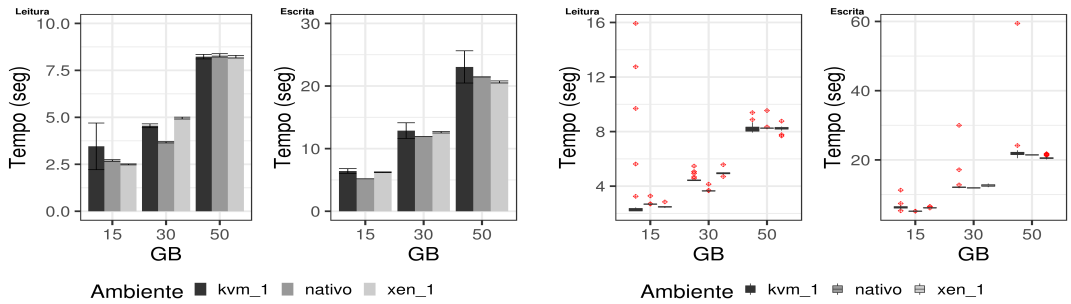
Finalmente, o resultado de desempenho compartilhado se mostrou dentro do esperado, pois o desempenho se manteve estável, isso acontece devido ao fato da virtualização garantir o isolamento dos processos e processadores virtualizados. Desta forma, uma máquina virtual não afeta o desempenho da outra, desde que estas não ultrapassem os limites computacionais do próprio hospedeiro.

6.2 Memória

Nessa etapa o software `dd` é utilizado em todos os ambientes para obter uma estimativa de performance da memória RAM com variações de tamanho dos arquivos. Com esse software pode-se averiguar o desempenho de todas as máquinas virtuais e do ambiente nativo no quesito de leitura e escrita sequencial. O parâmetro de comparação é o tempo necessário para ler ou escrever determinado arquivo em memória.

6.2.1 Benchmark de uso de memória de forma exclusiva

Nessa etapa o software `dd` é utilizado em todos os ambientes para obter uma estimativa de desempenho de memória no quesito de leitura e escrita sequencial com variações de tamanho dos arquivos. O parâmetro de comparação é o tempo necessário, em segundos, para ler ou escrever determinado arquivo em memória.



(a) Leitura/Escrita (b) Boxplot

Figura 6.5 – Leitura/Escrita e BoxPlot

A Figura 6.5 apresenta os gráficos com a comparação de velocidade de leitura e escrita, onde quanto menor o tempo mais eficiente é o sistema. Percebe-se que o Xen possui um melhor desempenho de leitura que os demais ambientes, principalmente nos testes de leitura de 15 GB, contudo quando o arquivo lido possui 30GB o ambiente nativo se torna melhor e com 50GB os ambiente se tornam estatisticamente semelhantes. Algo que chama a atenção é o tamanho da barra de intervalo de confiança do ambiente KVM durante a leitura de 15GB. A princípio esse caso foi interpretado como uma amostra com problemas, mas o mesmo fenômeno é encontrado ao refazer as simulações. Isso quer dizer que a variação nos tempos de leitura do KVM são grandes e que normalmente ocorrem.

Além disso, ao identificar esses `outliers` no ambiente KVM, é percebido que eles representam os primeiros cálculos realizados e mostram como o `hypervisor` trabalha com `cache` de memória, pois conforme os experimentos são feitos os tempos diminuem e esse comportamento é acentuado quando se observa o desempenho com 15GB.

De forma semelhante, é possível verificar que o desempenho de escrita de todos os ambientes são bem próximos e somente nos testes de 15G que o ambiente nativo consegue um desempenho estatisticamente melhor que os ambientes virtualizados. Nestes testes o Xen se mostra um `hypervisor` com menor variação nos resultados, além de manter um bom índice de desempenho, principalmente na escrita.

6.2.2 Benchmark de uso de memória de forma compartilhada

Os `hypervisors` apresentam um comportamento muito semelhante nos testes de desempenho com uso de memória de forma compartilhada. Nesta segunda etapa, os testes são feitos com duas máquinas virtuais, executadas de forma simultânea em um mesmo sistema hospedeiro, assim é possível verificar se existe queda de desempenho ou se ele continua semelhante ao uso exclusivo, além de ser possível determinar qual `hypervisor` se comporta de forma mais estável.

A Figura 6.6 apresenta um gráfico que exemplifica o bom uso de memória de ambos os `hypervisors`, e percebe-se que ambos conseguem trabalhar de forma consistente e estável com uma leve vantagem na taxa de leitura do KVM.

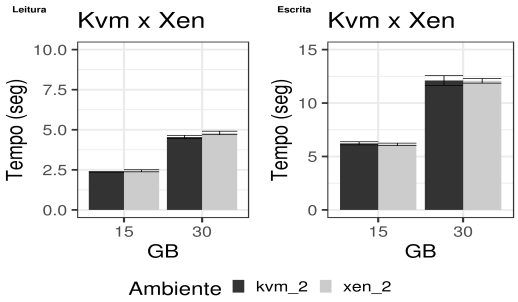


Figura 6.6 – Desempenho de Leitura/Escrita de Memória em uso Compartilhado

Por fim, uma observação importante a ser considerada é que o `hypervisor` Xen não permite o provisionamento de máquinas virtuais com 60GB de memória, mesmo com o servidor hospedeiro possuindo 64GB. Um dos motivos é a necessidade de uma parte da memória do hospedeiro ser resevada para gerenciar seus processos e para executar o `Control Domain`. Devido a esse fator, os benchmarks foram feitos com no máximo 50GB. Também é importante mencionar que esse problema não ocorre no caso do KVM, pois sua arquitetura de virtualização não necessita de uma máquina virtual como `Control Domain`.

6.3 Disco

Nessa etapa, antes da execução dos testes é feita uma avaliação do disco, cujos parâmetros de controle são listados utilizando o `HDparm`. Dessa forma, é possível garantir os mesmos valores em todos os ambientes, tornando os resultados mais confiáveis. Os parâmetros de controle avaliados e seus respectivos valores são: `CNominal Media Rotation Rate`: 10000; `R/W multiple sector transfer`: Max = 16 Current = 16; `Advanced power management level`: Max = 128 Current = 128; `DMA`: *udma6.

6.3.1 Benchmark de uso de disco de forma exclusiva

A primeira bateria de testes é feita no ambiente nativo e nos ambientes virtuais de forma exclusiva. A Figura 6.7 apresenta um gráfico com a comparação do tempo necessário para leitura ou escrita no disco.

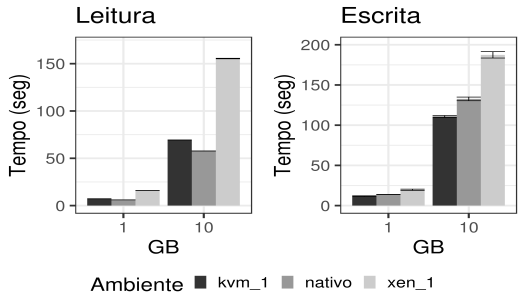


Figura 6.7 – Desempenho de Leitura/Escrita de Disco

Nesse experimento, o Xen apresenta um desempenho pior que os outros ambientes, enquanto que o KVM obtém um resultado mais próximo ao ambiente nativo, conseguindo até mesmo um desempenho melhor nos testes de escrita. Para explicar esse fenômeno do KVM é necessário avaliar os parâmetros de `cache`, nesse caso, o parâmetro de cache consta `writeback`. Esse parâmetro habilita o uso de `cache` durante escrita em disco, ou seja, o convidado faz um `cache` no hospedeiro antes de gravar no disco. Além disso, o parâmetro `IO mode` também é avaliado. Por padrão o `IO mode` usa modo `default (threads)`, cuja caracteristica é o uso de `I/O` assíncrono com opções de `I/O` direto.

A recomendação do Guia de Virtualização, Otimização e Tunning, no capítulo 7 Block I/O, para ambientes com alta densidade de `I/O` em disco é o `cache=none` e o `IO mode=native`. O parâmetro `cache=none` permite, ao sistema convidado, escrever direto no disco sem a necessidade de fazer `cache` no sistema hospedeiro, dessa forma, o sistema hospedeiro reduz a cópia de dados. De fato, isso melhora o desempenho ao mesmo tempo que aumenta a chance de perda de dados, pois em caso de queda de energia a probabilidade de corrupção de arquivos é muito grande. (HERRMANN, 2019)

Por sua vez, o Xen apresenta um desempenho pior que os outros ambientes. Sua recomendação para obter um melhor desempenho é o uso de `storage` com `RAID (Redundant Array of Independent Disks)` e `LVM (Logical Volume Manager)` com vários discos. No entanto, o ambiente utilizado possuia apenas um disco local e não permite essa configuração. Em seu manual de ajuste de performance não há opções de parâmetros de disco, mas existem recomendações para a alteração da memória alocada no `Control Domain`, pois a memória livre disponível é utilizada para `cache` de leitura. (CITRIX, 2019)

Finalmente, é importante salientar que, neste trabalho, são mantidos os parâmetros de forma padrão em ambos os `hypervisors`, pois a intenção é permitir que o ambiente atenda as difenrentes necessidades dos usuários. Além disso, a configuração utilizada no Xen foi a mesma do KVM, ou seja, um arquivo de dados criado em um disco de armazenamento.

6.3.2 Benchmark de uso de disco de forma compartilhada

Na segunda etapa, os testes são feitos com duas máquinas virtuais sendo executadas simultaneamente no sistema hospedeiro. Isso gera concorrência no uso de disco, pois o servidor hospedeiro possui apenas um disco físico. A Figura 6.8 apresenta os resultados obtidos.

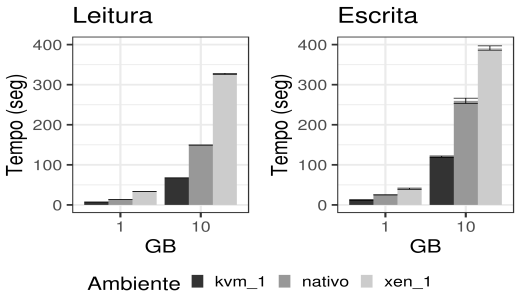


Figura 6.8 – Desempenho de Leitura/Escrita de Disco Compartilhado

Nestes gráficos da Figura 6.8, pode-se perceber que o uso compartilhado apresenta uma vantagem do KVM sobre o Xen. Notadamente, o KVM apresenta maturidade quanto ao uso das funcionalidades do `kernel` do Linux.

No final dos experimentos o resultado esperado era que o ambiente nativo fosse mais performático, pois os `hypervisors` usam a técnica de virtualização total, onde todos os dispositivos são virtualizados e a máquina virtual é tratada como um processo regular do Linux hospedeiro. Por isso o resultado demonstra a maturidade do KVM quanto ao uso de recursos e funcionalidades do `kernel` do Linux hospedeiro e, portanto, sendo uma melhor escolha para cálculos que demandam mais uso de disco.

7 Estudos de Caso com Programas de Cálculo Científico

Neste capítulo são avaliados os resultados obtidos a partir dos estudos de caso com softwares científicos que são utilizados na pesquisa teórica tanto na área de química teórica quanto na área de física teórica.

7.1 Análise de Desempenho do Software NAMD

Esta primeira análise foi feita utilizando o software NAMD 2.13 - Nanoscale Molecular Dynamics. Trata-se de um software que usa processamento paralelo projetado para simulação de alto desempenho de sistemas biomoleculares grandes, além disso, ele foi dimensionado para realizar as simulações utilizando centenas de núcleos. Neste trabalho, o NAMD foi testado em ambientes virtuais e nativos com um máximo de 20 núcleos e 40 threads se levar em consideração o hyperthreading. O benchmark foi executado levando em consideração as melhores práticas descritas pelos criadores do software, portanto a métrica a ser utilizada foi a quantidade de nanosegundos realizados de cálculo por dia. (PHILLIPS, 2005)

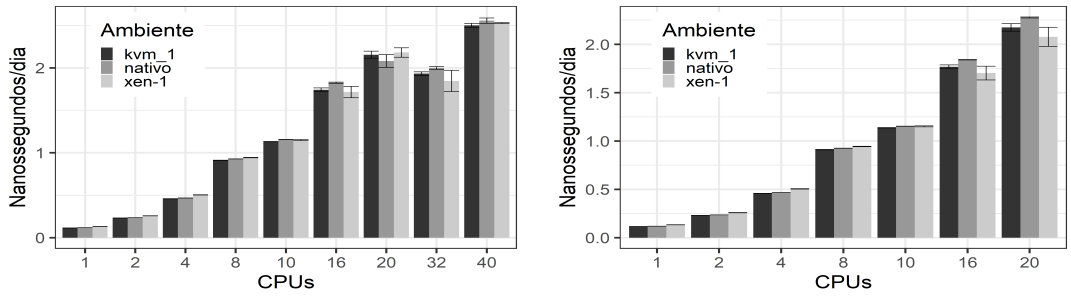
A proteína utilizada na realização desse benchmark foi a Apolipoproteína A-I. Trata-se de uma proteína que em humanos é codificada pelo gene APOA1, tendo um papel específico no metabolismo de lipídios. E ela foi escolhida por ser utilizada nos testes de benchmark realizados pelos desenvolvedores do software. (PHILLIPS, 2005) Seu tamanho é de 92.224 átomos e o cálculo é feito até completar 500 passos de integração das equações de movimento.

7.1.1 Resultados Obtidos com o NAMD

O comportamento do software se caracteriza pelo alto consumo de processamento e baixo uso de memória e disco. Isso o torna um software com alto potencial de ser usado em ambientes virtualizados, pois sua perda de desempenho pode ser baixa tornando as vantagens do ambiente virtualizado maiores que a desvantagem da perda de performance.

Após os testes foi observado que seu consumo de memória foi por volta de 350MB com 1 thread até 2933MB com 40 threads e o uso de disco foi de 186KB, comprovando o pouco uso de memória e disco. Sendo assim, ao observar a diferença entre a

quantidade de nanosegundos calculados por cada conjunto de CPUs e cada ambiente é possível perceber que o ambiente nativo possui melhor desempenho, como esperado.



(a) NAMD com hyperthreading (b) NAMD sem hyperthreading

Figura 7.1 – Desempenho com NAMD

Ao analisar os gráficos da Figura 7.1, percebe-se que a perda de performance é ínfima ao comparar os ambientes virtuais com o nativo. Portanto, trata-se de um ótimo sinal para quem utiliza esse software em ambientes virtualizados ou em nuvem computacional.

Além disso, é possível perceber a influência positiva do hyperthreading nos resultados, pois é possível observar um aumento, embora pequeno, no desempenho se comparado com os cálculos realizados sem essa tecnologia. Nesse caso, o uso de hyperthreading pode significar um ganho de aproximadamente 10% cálculos com o NAMD, desde que seja feito nas mesmas condições deste trabalho.

Em seguida, com a análise da Tabela 7.1 e da Tabela 7.2 é possível perceber a eficiência de escalabilidade do NAMD é muito boa. O cálculo de eficiência é feito com a divisão do resultado de 40 threads pelo resultado de 1 thread. Através desse cálculo é possível saber quantas vezes mais rápido o cálculo foi executado ao aumentar os recursos computacionais. No ambiente nativo, o ganho é de 21 vezes com o uso de hyperthreading e 19 vezes sem hyperthreading.

	1 Thread	40 Threads	Ganho
nativo	0.1192	2.5565	21.43 vezes
kvm-1	0.1152	2.5016	21.71 vezes
xen-1	0.1334	2.5290	18.95 vezes

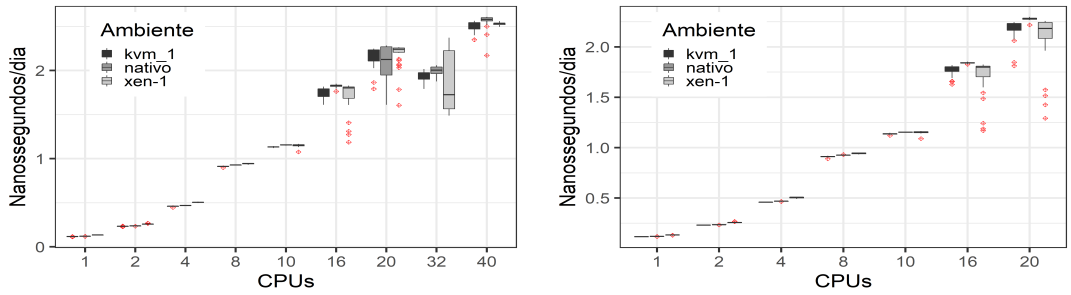
Tabela 7.1 – Nanossegundos calculados com Hyperthreading.

Pode-se perceber uma certa variação nos resultados quando é observado o gráfico boxplot da Figura 7.2. A variação acontece principalmente ao utilizar 16 threads ou mais no ambiente virtualizado com o Xen. Esse efeito também acontece no ambiente

	1 Thread	20 Threads	Ganho
nativo	0.1191	2.27	19.11 vezes
kvm-1	0.1165	2.17	18.67 vezes
xen-1	0.1339	2.07	15.51 vezes

Tabela 7.2 – Nanossegundos calculados sem Hyperthreading.

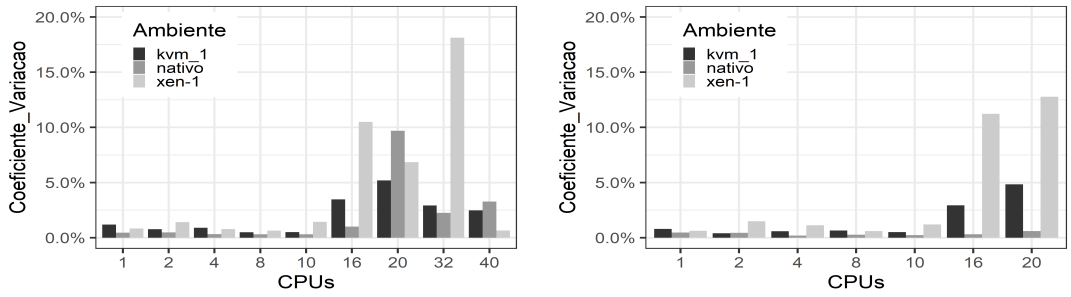
nativo e no KVM. Ao utilizar todos os 40 threads o ambiente Xen se mostra menos variável.



(a) Boxplot com hyperthreading (b) Boxplot sem hyperthreading

Figura 7.2 – Boxplot com NAMD

Com a análise dos gráficos de coeficiente de variação na Figura 7.3, pode-se notar uma maior variação no ambiente Xen, principalmente ao utilizar 16 e 32 threads no ambiente com hyperthreading e 16 e 20 threads no ambiente sem o hyperthreading. Entretanto, ao utilizar todas as 40 threads no Xen com hyperthreading, a variação fica próxia de 1% tornando o ambiente Xen a melhor escolha quando for utilizar todos os recursos computacionais de um nó.



(a) Coeficiente de Variação com hyperthreading (b) Coeficiente de Variação sem hyperthreading

Figura 7.3 – Coeficiente de Variação do NAMD

Finalmente, o desempenho sem hyperthreading se mostrou menos variável no ambiente nativo e no ambiente KVM, assim como o tempo de cálculo se manteve similar. A primeira vista os resultados com 40 threads nos ambientes virtuais são estatisticamente

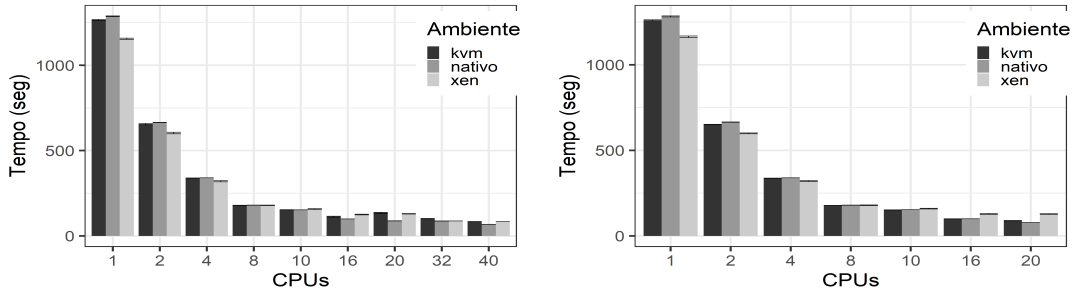
semelhantes, entretanto, ao analisar a variação dos resultados, percebe-se uma variação menor no ambiente com Xen. Ao mesmo tempo, percebe-se um ganho de aproximadamente 10% nos resultados com o uso de todos os recursos com hyperthreading, tornando uma boa opção para os cálculos com NAMD ter o ambiente virtualizado e com Xen e o processador com hyperthreading.

7.2 Análise de Desempenho do Software Gromacs

Nessa sessão o software utilizado foi o Gromacs (GROningen MAchine for Chemical Simulation), trata-se de um software de dinâmica molecular utilizado para cálculos de moléculas utilizando física clássica. (BERENDSEN, 1994) Assim como o NAMD, o Gromacs utiliza majoritariamente CPU. Para a utilização do Gromacs foi necessário fazer a compilação do código, pois ele é disponibilizado como código fonte. Na compilação do Gromacs foi utilizado o compilador GCC 9.2.0, o OpenMPI 3.1.3 e o cmake 3.15. O sistema utilizado no cálculo foi encontrado no site HECBioSim que possui sistemas simples para benchmarks. (HECBIOSIM, 2020) A métrica para a avaliação de desempenho foi o tempo, em segundos, necessário para executar 10000 passos das equações de movimento.

7.2.1 Resultados Obtidos com o Gromacs

Ao analisar os resultados obtidos com os cálculos do Gromacs é possível notar que um pequeno ganho de performance 40 threads no ambiente nativo. Além disso, pode-se perceber uma pequena melhora no desempenho ao utilizar o hyperthreading nos ambientes virtualizados também. Os gráficos da Figura 7.4 ilustram os resultados obtidos. Nesse caso, o uso de hyperthreading foi vantajoso em todos os ambientes e o ambiente virtualizado com o melhor tempo foi o Xen.



(a) Gromacs Desempenho com hyperthreading (b) Gromacs Desempenho sem hyperthreading

Figura 7.4 – Gráficos de Desempenho do Gromacs

As Tabelas 7.3 e 7.4 exibem de forma mais precisa os resultados alcançados pelos cálculos. Ao ver as duas tabelas é possível notar o ganho de desempenho com o uso

do hyperthreanding.

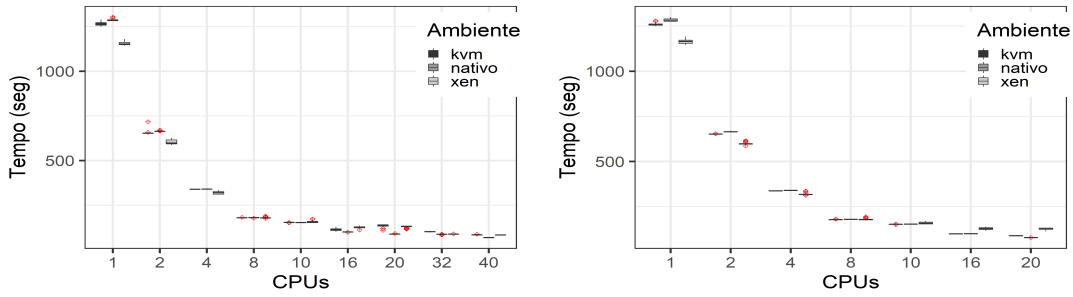
	1 Thread	40 Threads	Ganho
nativo	1287.23	68.35	18.83 vezes
kvm-1	1265.34	83.76	15.10 vezes
xen-1	1155.51	83.47	13.84 vezes

Tabela 7.3 – Tempo Médio do Cálculo com Hyperthreading.

	1 Thread	20 Threads	Ganho
nativo	1282.50	78.49	16.33 vezes
kvm-1	1259.67	89.16	14.12 vezes
xen-1	1163.77	127.34	9.13 vezes

Tabela 7.4 – Tempo Médio do Cálculo sem Hyperthreading.

Por sua vez, ao analisar a variação dos cálculos na Figura 7.5 é possível perceber alguns pontos de anomalia ao usar 20 threads nos ambientes virtualizados com hyperthreading. Nesses casos as anomalias não são os primeiros cálculos, então isso pode estar acontecendo devido a distrobuição de processos entre núcleos físicos e virtuais.

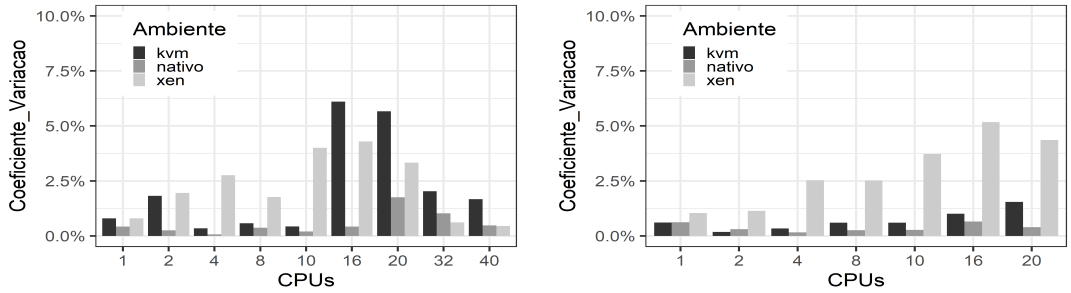


(a) Gromacs Boxplot com hyperthreading (b) Gromacs Boxplot sem hyperthreading

Figura 7.5 – Gráficos de Boxplot do Gromacs

Para melhorar a análise de variação dos resultados, pode-se verificar o coeficiente de variação nos gráficos da Figura 7.6. Ao utilizar hyperthreading a variação apresentada é maior no ambiente com KVM mesmo assim não chega a 7%. Por outro lado, quando o uso de hyperthreading é desligado a variação do ambiente KVM cai abaixo de 2%, enquanto que a do ambiente com Xen chega a 5% com 16 threads.

Finalmente, é possível determinar que o ambiente com Gromacs fica menos variável e com um bom desempenho no ambiente KVM sem hyperthreading, pois não perde muito desempenho quando comparado ao ambiente nativo e o resultado alcançado é estatisticamente melhor que o Xen. Entretanto, ao utilizar hyperthreading o ambiente Xen consegue um melhor tempo e menor variação, tornando ele a melhor escolha quando o objetivo for melhor desempenho.



(a) Gromacs Coeficiente de Variação com hyperthreading

(b) Gromacs Coeficiente de Variação sem hyperthreading

Figura 7.6 – Gráficos de Coeficiente de Variação do Gromacs

7.3 Análise de Desempenho do Software Amber

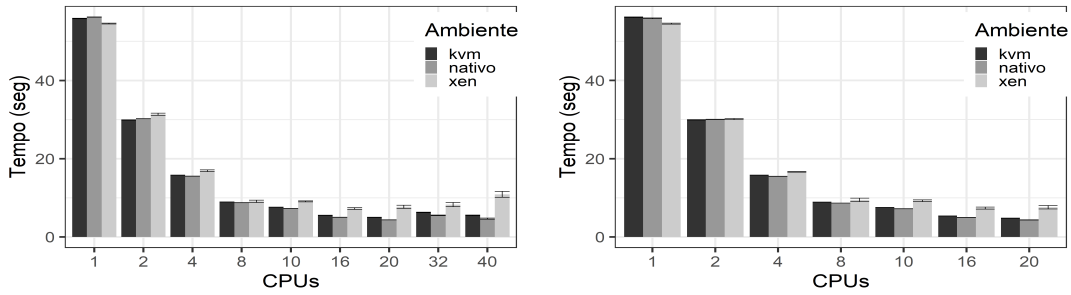
Nesta sessão o software Amber é utilizado para fazer as avaliações de desempenho. Trata-se de um conjunto de programas de simulação biomolecular. O software começou no final dos anos 1970 e é mantido por uma comunidade de desenvolvimento ativa. O termo "Amber"se refere a duas coisas. Em primeiro lugar, é um conjunto de campos de força mecânica molecular para a simulação de biomoléculas (esses campos de força são de domínio público e são usados em uma variedade de programas de simulação). Em segundo lugar, é um pacote de programas de simulação molecular que inclui código-fonte e demonstrações. (CASE, 2020). Este software tem como característica o uso majoritário de CPU, pois ele não faz um grande alocação memória ou alto uso leitura e escrita em disco.

Neste estudo de caso com o Amber, os resultados apresentam bastante estabilidade entre o ambiente virtual com KVM e o ambiente nativo, entretanto apresenta uma certa variação no ambiente virtualizado com Xen. O sistema utilizado no benchmark possui 20 mil átomos e a métrica utilizada nos testes foi o tempo, em segundos, para o Amber conseguir rodar 500 passos das equações de movimento. Este sistema utilizado no benchmark foi obtido no HECBioSim que consiste em um conjunto de benchmarks simples para uma série de softwares de dinâmica molecular populares, cada um dos quais é definido em uma contagem de átomos diferente. (HECBIOSIM, 2020).

7.3.1 Resultados Obtidos com o Amber

A Figura 7.7 mostra os gráficos com os resultados de tempo entre os ambientes com hyperthreading e sem Hyper-Threading habilitados no processador. Mesmo ao analisar somente as barras de erro dos gráficos já é possível perceber uma maior variação por parte do Xen. Além disso, percebe-se que o hyperthreading não melhorou o desempenho, mas pelo contrário, ele piorou o desempenho quando comparado aos benchmarks

equivalentes, ou seja, o cálculo com 16 threads rodou com mais desempenho do que o de 32 threads e também o cálculo com 20 threads obteve um melhor resultado do que o com 40 threads.



(a) Amber com hyperthreading (b) Amber sem hyperthreading

Figura 7.7 – Desempenho do Amber

Pode-se observar a eficiência de escalabilidade do Amber ao analisar a Tabela 7.5 com os tempos de execução médio dos cálculos com hyperthreading. O ambiente virtualizado com KVM mostra uma grande vantagem ao observar a Figura 7.7 e a Tablela 7.6.

	1 Thread	40 Threads	Ganho
nativo	56.21	4.68	12.01 vezes
kvm-1	55.83	5.51	10.13 vezes
xen-1	54.55	10.89	5.00 vezes

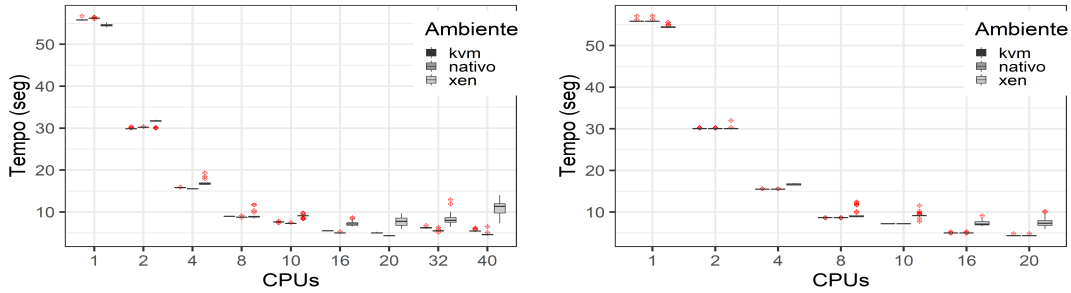
Tabela 7.5 – Tempo Médio do Cálculo com Hyperthreading.

	1 Thread	20 Threads	Ganho
nativo	55.88	4.34	12.87 vezes
kvm-1	56.15	4.78	11.74 vezes
xen-1	54.51	7.60	7.17 vezes

Tabela 7.6 – Tempo Médio do Cálculo sem Hyperthreading.

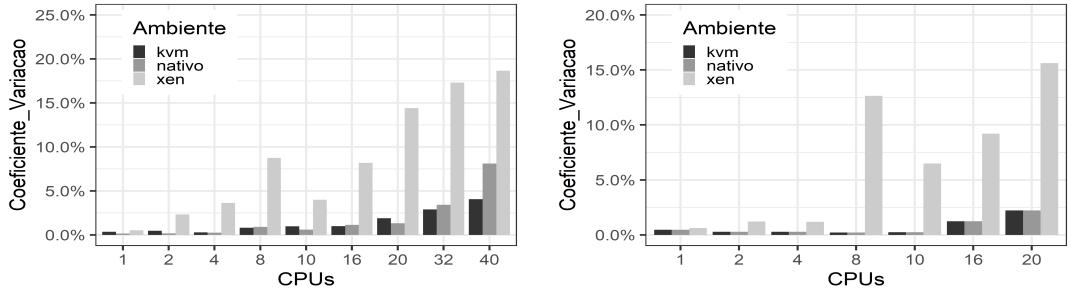
Ao analisar os gráficos de caixa na Figura 7.8, pode-se observar algumas anomalias no comportamento do Amber principalmente ao utilizar o ambiente com Xen. Essas anomalias foram identificadas como os primeiros cálculos executados.

Na Figura 7.9, a análise dos gráficos com o coeficiente de variação pode-se perceber que o ambiente com Xen gera uma variação significante mesmo sem o uso de hyperthreading. Nos resultados com 40 threads nota-se uma variação próxima a 20% enquanto que com 20 threads a variação se mantém em torno de 15% seja utilizando o hyper threading ou não. Por sua vez, o ambiente virtualizado com KVM se mantém estável e com resultados próximos ao do ambiente nativo.



(a) Boxplot com hyperthreading (b) Boxplot sem hyperthreading

Figura 7.8 – Boxplot com Amber



(a) Variação com hyperthreading (b) Variação sem hyperthreading

Figura 7.9 – Coeficiente de Variação do Amber

Finalmente, pode-se concluir que o ambiente virtualizado com KVM é mais performático que o ambiente Xen e que mantém o desempenho mais próximo ao ambiente nativo. Além disso, também percebe-se que o uso de hyperthreading não melhora o desempenho do cálculo, ou seja, o ambiente virtualizado com KVM sem hyperthreading é a melhor escolha para cálculos com o Amber.

7.4 Análise de Desempenho do Software Lammps

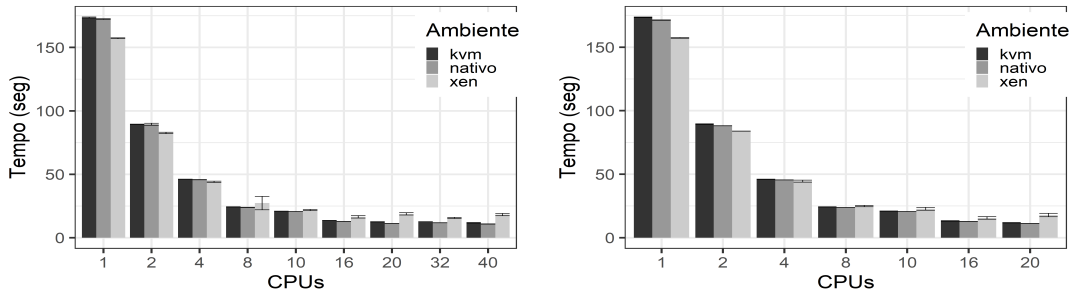
O software utilizado nesta sessão foi o Lammps. Software que começou a ser desenvolvido em meados de 1990 sob um acordo cooperativo de pesquisa e desenvolvimento (CRADA) entre dois laboratórios DOE (Sandia e LLNL) e 3 empresas (Cray, Bristol Myers Squibb e Dupont). O objetivo era desenvolver um código de Dinâmica Molecular clássico paralelo em grande escala; o esforço de codificação foi liderado por Steve Plimpton da Sandia, (PLIMPTON, 1995). Também trata-se de um software livre, ou seja, pode ser utilizado por qualquer pessoa. Assim como o NAMD e o Amber, o Lammps tem como característica o uso majoritário de CPU, pois ele não faz um grande alocação memória ou alto uso leitura e escrita em disco.

Para a utilização do Lammps nos benchmarks foi necessário fazer a compilação

do código, pois ele é disponibilizado como código fonte. A versão utilizada foi a de 03 de Março de 2020, para fazer a compilação do Lammps foi utilizado o GCC 4.8.5 e o OpenMPI 3.0.0, além disso, os pacotes do Lammps utilizados na compilação foram: USER-MISC, RIGID, MOLECULE, MANYBODY, KSPACE. Esses pacotes foram necessários para rodar o sistema de 20 mil átomos obtidos no (HECBIOSIM, 2020) . Por sua vez, a métrica para a avaliação de desempenho foi o tempo, em segundos, necessário para executar 500 passos das equações de movimento.

7.4.1 Resultados Obtidos com o Lammps

Na Figura 7.10 os gráficos apresentam os resultados de tempo entre os ambientes com hyperthreading e sem hyperthreading habilitados. Ao analisar os gráficos de barras é possível perceber uma maior variação por parte do Xen. Além disso, pode-se perceber que o hyperthreading não melhorou o desempenho, pois os resultados não melhoram ao adicionar mais recursos.



(a) Lammps com hyperthreading (b) Lammps sem hyperthreading

Figura 7.10 – Gráfico de Desempenho do Lammps

Ao observar o cálculo de eficiência com os dados da Tabela 7.7 e da Tabela 7.8 é possível perceber que o hyperthreading não gera vantagem em seu uso, pois o resultado alcançado pelo ambientes é bem similar. Portanto, o uso de hyperthreading não é vantajoso quando o cálculo pe feito com o programa Lammps.

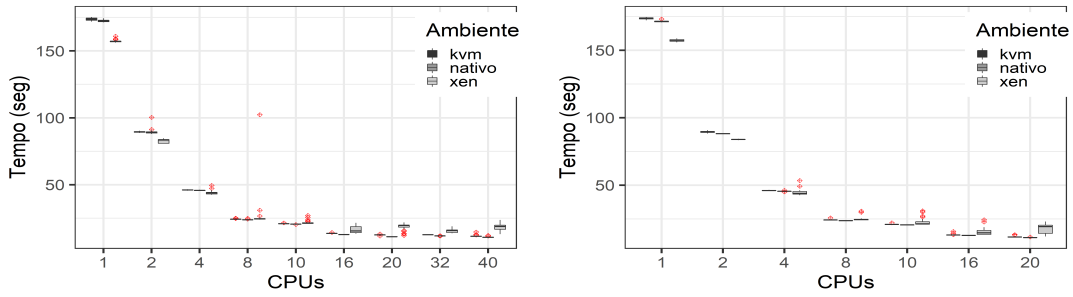
	1 Thread	40 Threads	Ganho
nativo	172.41	10.82	15.93 vezes
kvm-1	173.61	11.71	14.82 vezes
xen-1	157.22	18.28	8.60 vezes

Tabela 7.7 – Tempo Médio do Cálculo com Hyperthreading.

Por sua vez, ao analisar a Figura 7.11 com os gráficos de caixa, é possível notar novamente uma maior variação no ambiente Xen que apresenta várias anomalias nos resultados. Essas anomalias são as primeiras execuções dos cálculos.

	1 Thread	20 Threads	Ganho
nativo	171.33	11.23	15.25 vezes
kvm-1	173.54	11.72	14.80 vezes
xen-1	157.28	17.86	8.80 vezes

Tabela 7.8 – Tempo Médio do Cálculo sem Hyperthreading.

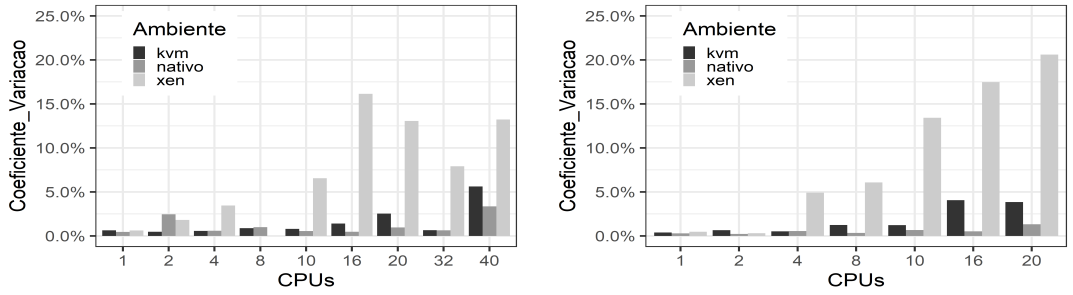


(a) Boxplot com hyperthreading

(b) Boxplot sem hyperthreading

Figura 7.11 – Boxplot do Lammps

Na Figura 7.12, através da análise dos gráficos de coeficiente de variação, pode-se perceber que o ambiente com Xen possui uma maior variação do que os demais ambientes mesmo ao utilizar o ambiente sem hyperthreading. Além disso, ao utilizar o ambiente com Xen, nota-se uma variação de até 20%, ao mesmo tempo que o ambiente KVM mantém sua variação próxima ao do ambiente nativo.



(a) Coeficiente de Variação com hyperthreading

(b) Coeficiente de Variação sem hyperthreading

Figura 7.12 – Coeficiente de Variação do Lammps

Por fim, pode-se destacar que o ambiente KVM é menos variável e mantém um desempenho muito semelhante ao ambiente nativo, tornando o KVM sem hyperthreading uma boa escolha para quem pretende utilizar um ambiente virtualizado para rodar cálculos em Lammps. Nota-se também que existe uma semelhança no comportamento do Lammps e do benchmark de uso de CPU comprovando que os testes de CPU podem servir como orientação sobre os softwares de dinâmica molecular que utilizam majoritariamente CPU.

7.5 Análise de Desempenho do Software Gaussian

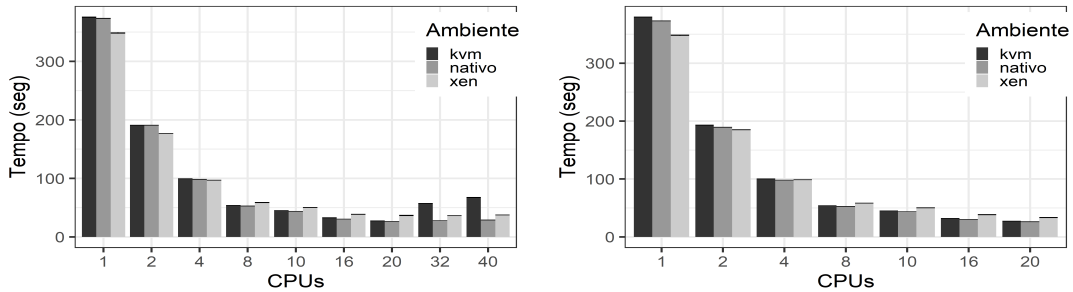
Nesta sessão o software utilizado para fazer o estudo de caso foi o Gaussian, trata-se de um pacote de software de química quântica computacional lançado inicialmente em 1970 por John Pople e seu grupo de pesquisa na Universidade Carnegie Mellon como Gaussian 70. (AL., 2016)

Diferente do Gamess, o Gaussian não é um software gratuito, portanto a utilização dele foi possível graças à licença institucional da Unicamp. Assim como o Gamess, o Gaussian tem como característica um bom uso de CPU, memória e disco, pois ele é otimizado para utilizar os recursos computacionais disponíveis.

Diferente do Gamess e Gromacs, não é necessário fazer a compilação do Gaussian, pois ele vem compilado para uso no Linux, sendo assim a instalação simples e igual em todos os ambientes. O sistema utilizado foi um cálculo de energia da molécula 5-amino tropolona usando funcional de densidade b3lyp e função de base coupled cluster triple zeta. O cálculo de energia de molécula foi escolhido, pois esse cálculo permite verificar o ganho de desempenho conforme é aumentado a quantidade de recursos computacionais. (PAZ JUAN J.; MORENO, 1998). A métrica para a avaliação de desempenho foi o tempo, em segundos, necessário para executar o cálculo de energia da molécula 5-amino tropolona.

7.5.1 Resultados Obtidos com o Gaussian

Como pode ser observado na Figura 7.13, os gráficos ilustram o ganho de desempenho nos ambientes com e sem hyperthreading. Nota-se que o hyperthreading não trás benefício no uso do Gaussian, principalmente no ambiente KVM. Ao mesmo tempo que no ambiente nativo e no ambiente Xen o tempo para conclusão do cálculo permanece similar ao utilizar 20 ou mais threads.



(a) Gaussian com hyperthreading (b) Gaussian sem hyperthreading

Figura 7.13 – Gráficos de Desempenho do Gaussian

Ao analisar o desempenho com a utilização de 8 threads ou mais no gráfico onde o hyperthreading não é utilizado, percebe-se que o ambiente KVM e nativo ficam

com tempos similares e o ambiente Xen mantém o menor desempenho. Obviamente, o melhor uso de múltiplos threads é mais importante para a computação paralela, logo as observações levam em consideração o melhor desempenho com mais threads possível no ambiente.

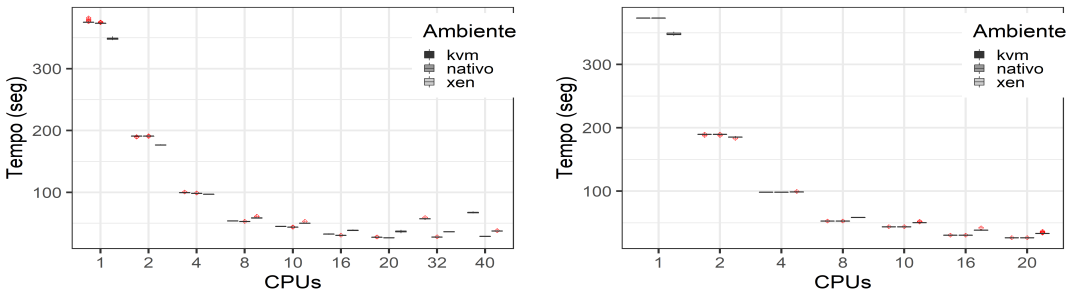
Ao analisar o cálculo de eficiência com os dados da Tabela 7.9, nota-se que o uso de hyperthreading prejudica mais o ambiente virtualizado com KVM, enquanto que o ambiente nativo e o Xen se mantém com resultados estáveis.

	1 Thread	40 Threads	Ganho
nativo	373.52	28.98	12.88 vezes
kvm-1	375.37	67.43	5,56 vezes
xen-1	348.62	37.62	9.26 vezes

Tabela 7.9 – Tempo Médio do Cálculo com Hyperthreading.

	1 Thread	20 Threads	Ganho
nativo	373.07	26.27	14.20 vezes
kvm-1	379.65	27.22	13.94 vezes
xen-1	348.07	33.44	10.40 vezes

Tabela 7.10 – Tempo Médio do Cálculo sem Hyperthreading.

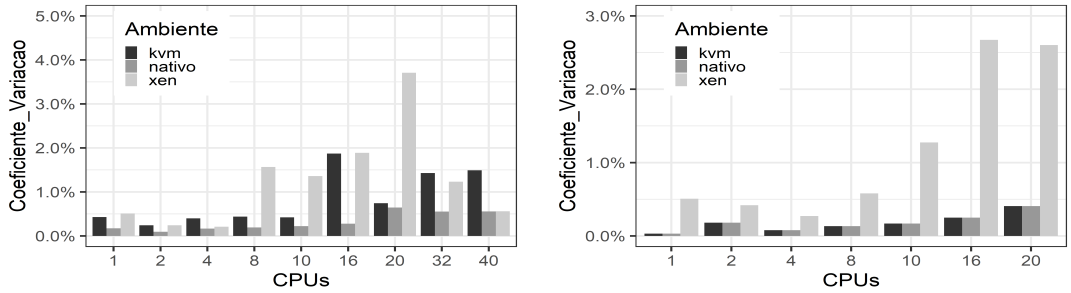


(a) Boxplot com hyperthreading (b) Boxplot sem hyperthreading

Figura 7.14 – Gráficos Boxplot do Gaussian

Por sua vez, ao analisar os gráficos de caixa na Figura 7.14 e os gráficos de coeficiente de variação na Figura 7.15, pode-se notar que em todos os ambientes a variação é baixa, não chegando a 4% no ambiente com maior variação, no caso o Xen com 20 threads. Essa variação baixa acontece tanto nos ambientes com uso do hyperthreading quanto nos que não usam.

Sendo assim, foi possível determinar que o ambiente com KVM sem hyperthreading é uma melhor escolha de virtualização, pois o KVM mostrou um desempenho similar ao ambiente nativo e o uso de hyperthreading não trouxe benefícios ao ambiente.



(a) Coeficiente de Variação com hyperthreading

(b) Coeficiente de Variação sem hyperthreading

Figura 7.15 – Gráficos de Coeficiente de Variação do Gaussian

7.6 Análise de Desempenho do Software Gamess

No segundo estudo de caso foi utilizado o software `Gamess (General Atomic and Molecular Electronic Structure System)`(SCHMIDT, 1993). Trata-se de um software de processamento paralelo utilizado para pesquisas de química quântica. No trabalho, foram feitos cálculos teóricos utilizando resultados não publicados e a versão GAMESS-2014R1. Além disso, ele foi compilado da mesma forma e com os mesmos parâmetros e utilizando o compilador gfortran 4.8.5 em todos os ambientes.

7.6.1 Resultados Obtidos com o Gamess

O comportamento do software se caracteriza pelo alto consumo de processamento e disco. A medida de análise de desempenho usada foi a velocidade, em segundos, que o cálculo demorou para realizar 1.000.000 de iterações. Durante a execução do cálculo percebeu-se que foi utilizado 1.38GB de armazenamento em disco.

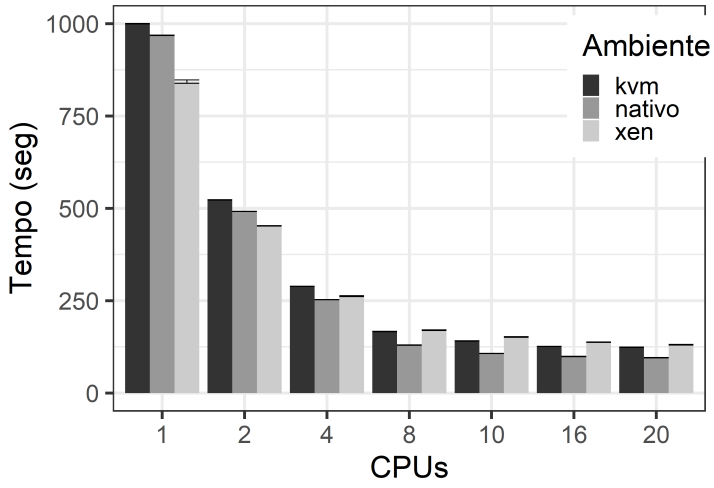


Figura 7.16 – Gráfico de desempenho do Gamess sem hyperthreanding.

Ao analisar o gráfico de desempenho na Figura 7.16, é possível verificar que

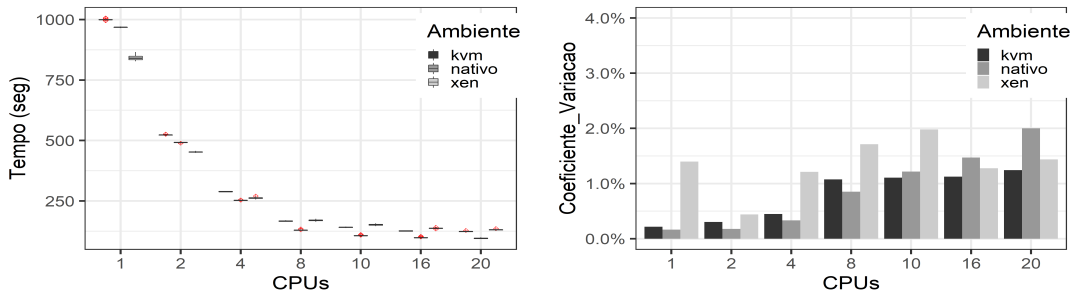
o ambiente nativo consegue realizar os cálculos mais rápidos com 20 threads, seguido do KVM e XEN, respectivamente. Ao mesmo tempo que é percepitivel o pouco ganho de desempenho ao aumentar os recursos após 10 threads. Neste caso, o XEN obteve um desempenho levemente pior que o KVM e que o ambiente nativo. É importante notificar que o Gamess não funcionou ao utilizar hyperthreading, mas ao utilizar somente os núcleos físicos não há qualquer problema.

Em seguida, com a análise da Tabela 7.11 é possível perceber a eficiência de escalabilidade do Gamess. Nela, pode-se verificar o tempo médio, em segundos, que o cálculo demorou para ser executado. Nesse caso, o melhor ganho é de 10 vezes no ambiente nativo seguidos de 8 vezes com KVM e 6 vezes com Xen.

	1 Thread	20 Threads	Ganho
nativo	968.13	95.90	10,09 vezes
kvm-1	999.51	123.63	8,08 vezes
xen-1	842.93	130.78	6,44 vezes

Tabela 7.11 – Tempo Médio do Cálculo sem Hyperthreading.

Ao analisar os gráficos boxplot e o coeficiente de variação na Figura 7.17, pode-se notar que a variação e todos os ambientes é bem pequena, chegando ao máximo de 2% no ambiente Xen com 10 threads e no ambiente nativo com 20 threads.



(a) Boxplot Gamess (b) Coeficiente de Variação

Figura 7.17 – Boxplot e Coeficiente de Variação do Gamess

Por fim, conclui-se que a melhor alternativa para a utilização do Gamess é o ambiente virtualizado com KVM sem hyperthreading, onde terá uma pequena perda de desempenho se comparado ao nativo, mas levemente melhor que o ambiente Xen.

7.7 Resultados de Cada Software

Nesta seção é apresentada a Tabela 7.12 com os resultados de cada software nos ambientes KVM e Xen. Esta tabela, apresenta apenas os resultados dos ambientes

virtualizados e o se o uso de hyperthreading é vantajoso no ganho de desempenho. Na tabela, os quatro primeiros softwares são de dinâmica molecular, ou seja, NAMD, Amber, Lammps, e Gromacs e os últimos dois softwares são de química quântica.

Software	Resultados com Melhor Desempenho	Xen	KVM
NAMD	Xen com hyperthreading	2.5290 ns	2.5016 ns
Gromacs	Xen com hyperthreading	83.47 s	83.76 s
Amber	KVM sem hyperthreading	7.60 s	4.78 s
Lammps	KVM sem hyperthreading	17.86 s	11.72 s
Gaussian	KVM sem hyperthreading	33.44 s	27.22 s
Gamess	KVM sem hyperthreading	130.78 s	123.63 s

Tabela 7.12 – Ambientes com Melhor Desempenho.

Conclusão

Este trabalho apresenta comparações de desempenho entre os hypervisors gratuitos que podem ser utilizados na montagem de ambientes computacionais virtualizados. Nosso objetivo foi determinar o hypervisor com melhor desempenho de acordo com a aplicação apresentada. Assim, através dos resultados obtidos, pode-se perceber o impacto dos diferentes hypervisors nas aplicações utilizadas e em seu resultado final.

Dessa maneira, é analisado o desempenho no uso de CPU, memória e disco, de forma exclusiva e compartilhada, em ambientes virtualizados com os hypervisors KVM e Xen. É importante salientar que os benchmarks são feitos em um ambiente de processamento paralelo fortemente acoplado, ou seja, um único sistema com grande capacidade computacional.

Os benchmarks de CPU, memória e disco mostram que ao utilizar CPU o Xen obtém melhor desempenho quanto ao uso de poucos CPUs, mas no momento em que o número de CPUs aumenta os dois hypervisors performam de forma semelhante, tanto no uso de forma exclusiva quanto no uso de forma compartilhada. Além disso, ambos têm um desempenho satisfatório se comparado ao ambiente sem virtualização, tornando ambos os hypervisors aptos para ambientes que demandam alto processamento de CPU. De forma semelhante, o benchmark de uso de memória apresenta uma vantagem para o KVM nos testes de leitura. E no uso de leitura e escrita em disco, o KVM apresenta um desempenho consideravelmente maior que o Xen em ambos os casos, exclusivo e compartilhado. Além disso, o KVM mantém a performance semelhante, tanto no exclusivo quanto no compartilhado, enquanto que o Xen foi fortemente impactado, tendo sua performance reduzida em torno de 50% no uso compartilhado de disco.

Sendo assim, é possível inferir que o Xen pode ser utilizado para aplicações que demandem alto processamento, enquanto que o KVM pode ser uma melhor opção para aplicações que demandem leitura e escrita em disco.

Por sua vez, os resultados dos estudos de caso que utilizam aplicações científicas apresentam uma diferença entre os hypervisors. Os estudos apresentam que tanto o KVM quanto o Xen podem ter um desempenho impactado dependendo da aplicação e da forma como ela utiliza os recursos computacionais fornecidos. Dessa forma, tanto o KVM quanto o Xen apresentam algumas vantagens e desvantagens. O KVM mostra menos variação e um bom desempenho quando os cálculos são feitos com Gamess, Amber, Lammps e Gaussian. Essas aplicações não apresentam um bom uso do hyperthreading, pois essas aplicações possuem códigos muito otimizados que não deixam tempo vago no

processador e essa característica faz o ambiente com KVM sem hyperthreading ter um melhor desempenho. Ao mesmo tempo que o Xen apresenta um melhor desempenho ao fazer os cálculos com NAMD e Gromacs, ambos se beneficiando do uso de hyperthreading. Isso resulta no Xen com hyperthreanding o ambiente com melhor desempenho.

Finalmente, o trabalho consegue mapear os comportamentos distintos dos hypervisors e das aplicações científicas apresentando uma solução orientada a dados para quem pensa em utilizar um ambiente virtualizado para computação paralela.

Perspectivas Futuras

Trabalhos futuros a serem realizados podem incluir benchmarks em ambientes de processamento paralelo fracamente acoplados, ou seja, um conjunto maior de memória e processamento distribuido em vários nós de computação. Além disso, pode-se criar benchmarks para testar a eficiência das GPUs que potencializam muito o desempenho de cálculos científicos. Finalmente, esses resultados podem ajudar a comunidade científica a planejar seu ambiente de computação de alto desempenho virtualizado.

Referências

AL., F. M. J. et. Gaussian, Inc. 2016. Disponível em: <https://gaussian.com/>.

BARHAM, P. e. a. Xen and the Art of Virtualization. *ACM Symposium on Operating Systems Principles*, 2003. Disponível em: <https://dl.acm.org/doi/10.1145/945445.945462>.

BERENDSEN, H. J. C. e. a. GROMACS: A message-passing parallel molecular dynamics implementation. *Computer Physics Communications*, 1994. Disponível em: <https://www.sciencedirect.com/science/article/pii/001046559500042E?via3Dihub>.

BESERRA, D. e. a. Performance Evaluation of Hypervisors for HPC Applications. *IEEE International Conference on Systems, Man, and Cybernetics*, n. 14, 2016. Disponível em: <https://ieeexplore.ieee.org/document/7379288>.

CACIATO, L. E. Virtualização e Consolidação dos Servidores do Datacenter. *Centro de Computação da Universidade Estadual de Campinas – São Paulo*, n. 23, 2009. Disponível em: <http://www.ccuec.unicamp.br/biti/download/Artigo_Virtualizacao_Datacenter.pdf>.

CARISSIMI, A. Virtualização: da teoria a soluções. *26º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, 2008. Disponível em: <https://www.gta.ufrj.br/ensino/CPE758/artigos-basicos/cap4-v2.pdf>.

CARISSIMI, A. Desmistificando a Computação em Nuvem. *erad-RS*, 2015. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/erad-rs/2015/002.pdf>.

CASE, D. A. e. a. Amber 2020. 2020. Disponível em: <https://ambermd.org/>.

CHAKTHRANONT, N. e. a. Exploring the Performance Impact of Virtualization on an HPC Cloud. *IEEE - 6th International Conference on Cloud Computing Technology and Science*, n. 15, 2014. Disponível em: <https://ieeexplore.ieee.org/document/7037698>.

CITRIX. Citrix XenServer 7.1 Administrator's Guide. 2019. Disponível em: <https://docs.citrix.com/en-us/xenserver/7-1/downloads/administrators-guide.pdf>.

COLVERO TAIS A.; DANTAS, M. C. D. P. Ambientes de Clusters e Grids Computacionais: Características, Facilidades e Desafios. *Congresso Sul Brasileiro de Computação*, 2005. ISSN 2359-2656. Disponível em: <http://periodicos.unesc.net/sulcomp/article/view/798>.

DUATO, J. e. a. rCUDA. 2020. Disponível em: <http://www.rcuda.net/index.php/what-s-rcuda.html>.

DUATO JOSé, e. a. An Efficient Implementation of GPU Virtualization in High Performance Clusters. *European Conference on Parallel Processing*, 2010. Disponível em: <https://link.springer.com/chapter/10.1007/978-3-642-14122-5_44>.

FELTER, W. e. a. An updated performance comparison of virtual machines and Linux containers. *IEEE - International Symposium on Performance Analysis of Systems and Software (ISPASS)*, n. 30, 2015. Disponível em: <https://ieeexplore.ieee.org/document/7095802>.

GUPTA ABHISHEK; MILOJICIC, D. Evaluation of HPC Applications on Cloud. *IEEE - Sixth Open Cirrus Summit*, n. 17, 2012. Disponível em: <https://ieeexplore.ieee.org/document/6200551>.

HECBIOSIM. The HECBioSim Benchmarks. 2020. Disponível em: <https://www.hecbiosim.ac.uk/benchmarks>.

HERRMANN, J. e. a. Virtualization Tuning and Optimization Guide. 2019. Disponível em: <https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html-single/virtualization_tuning_and_optimization_guide/index#sect-Virtualization_Tuning_Optimization_Guide-BlockIO-Caching>.

JIANG YUJUAN; XIANGYANG, L. B. A. Function Virtualization in High Performance Computing: Opportunities and Challenges. *Cluster Computing. The Journal of Networks, Software Tools and Applications*, 2018. ISSN 1877-0509. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1877050920315908>.

KERRISK, M. hdparm(8) — Linux manual page. 2018. Disponível em: <https://man7.org/linux/man-pages/man8/hdparm.8.html>.

KVM. Linux KVM. 2019. Disponível em: <https://www.linux-kvm.org/page/Main_Page>.

LORD, M. hdparm. 2020. Disponível em: <https://sourceforge.net/projects/hdparm/>.

ORACLE. MySQL Community Downloads. 2020. Disponível em: <https://dev.mysql.com/downloads/benchmarks.html>.

PAZ JUAN J.; MORENO, M. L. J. M. A theoretical study of the isotope effects on the fluorescence excitation spectrum of 5-aminotropolone. *T.I.S. São Carlos - Tecnologias, Infraestrutura e Software*, 1998. Disponível em: <https://aip.scitation.org/doi/abs/10.1063/1.476251>.

PHILLIPS, J. C. e. a. Scalable molecular dynamics with NAMD. *Journal of Computational Chemistry*, 2005. ISSN 26:1781-1802. Disponível em: <http://www.ks.uiuc.edu/Research/namd>.

PICKARTZ, S. e. a. *Virtualization in HPC - An Enabler for Adaptive Co-Scheduling?* 1. ed. Lansdale, USA: IOS Press, 2017. v. 28nd. (Advances in Parallel Computing, 3).

PLIMPTON, S. e. a. Fast Parallel Algorithms for Short-Range Molecular Dynamics. 1995. Disponível em: <https://lammps.sandia.gov/>.

PRADES JAVIER; REAñO, C. S. F. On the effect of using rCUDA to provide CUDA acceleration to Xen virtual machines. *Cluster Computing. The Journal of Networks, Software Tools and Applications*, 2018. Disponível em: <https://link.springer.com/article/10.1007/s10586-018-2845-0>.

RIGHI, R. d. R. Elasticidade em cloud computing: conceito, estado da arte e novos desafios. *Revista Brasileira de Computação Aplicada*, p. 02–17, 2013. ISSN 2176-6649. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/erad-rs/2015/002.pdf>.

RUBIN PAUL; MACKENZIE, D. K. S. Linux manual page. 2019. Disponível em: <http://man7.org/linux/man-pages/man1/dd.1.html>.

SADOOGHI, I. e. a. Understanding the Performance and Potential of Cloud Computing for Scientific Applications. *IEEE - Transactions on Cloud Computing*, n. 19, 2015. Disponível em: <https://ieeexplore.ieee.org/document/7045591>.

SANTOS, D. L. Migração de servidores físicos para virtuais – P2V usando ferramentas Open Source. *Curso de Especialização em Redes e Segurança de Sistemas*, 2009. Disponível em: <http://www.ppgia.pucpr.br/~jamhour/RSS/TCCRSS08A/Diego20Lima20Santos20-20Artigo.pdf>.

SCHMIDT, M. W. e. a. General Atomic and Molecular Electronic Structure System GAMESS . *Journal of Computational Chemistry*, April 1993. ISSN 14:1347-1363. Disponível em: <https://www.msg.chem.iastate.edu/index.html>.

VERMA, A. Cloud Platform Optimization for HPC. *Asian Conference on Supercomputing Frontiers*, 2020. Disponível em: <https://link.springer.com/chapter/10.1007/978-3-030-48842-0_4>.

VMWARE. Virtualização – Visão Geral. 2019. Disponível em: <http://www.vmware.com/br/virtualization/overview>.

XAVIER, M. G. e. a. Performance Evaluation of Container Based Virtualization for HPC Computing Environments. *IEEE - 2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, n. 27, 2013. ISSN 2377-5750. Disponível em: <https://ieeexplore.ieee.org/document/6498558>.

XENSERVER. Xenserver. 2019. Disponível em: <https://xenserver.org>.

YADAV, R. R. e. a. Performance Comparison Between Virtual Machines and Docker Containers. *IEEE - Latin America Transactions*, n. 09, 2018. Disponível em: <https://ieeexplore.ieee.org/document/8528247>.

YANG, C.-T. e. a. On implementation of GPU virtualization using PCI pass-through. *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings*, 2012. Disponível em: <https://ieeexplore.ieee.org/document/6427531>.

YOUNGE, A. J. e. a. Analysis of Virtualization Technologies for High Performance Computing Environments. *IEEE - 4th International Conference on Cloud Computing*, n. 01, 2011. Disponível em: <https://ieeexplore.ieee.org/document/6008687>.

YOUNGE ANDREW J.; FOX, G. C. Advanced Virtualization Techniques for High Performance Cloud Cyberinfrastructure. *14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, n. 29, 2014. Disponível em: <https://ieeexplore.ieee.org/document/6846506>.