UNIVERSIDADE FEDERAL DO PAMPA

Willian Soares da Silva

Avaliação Automatizada de Desempenho de Sistemas de Virtualização GNU/Linux e VMware para Computação de Alto Desempenho

Alegrete
2019

Willian Soares da Silva

Avaliação Automatizada de Desempenho de Sistemas de Virtualização GNU/Linux e VMware para Computação de Alto Desempenho

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Claudio Schepke

Coorientador: Me. Raul Dias Leiria

Alegrete
2019

Willian Soares da Silva

Avaliação Automatizada de Desempenho de Sistemas de Virtualização GNU/Linux e VMware para Computação de Alto Desempenho

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Trabalho de Conclusão de Curso defendido e aprovado em 24 de junho de 2019
Banca examinadora:

Prof. Dr. Claudio Schepke
Orientador
UNIPAMPA

Me. Raul Dias Leiria
Co-orientador
PUCRS

Prof. Dr. Rodrigo Mansilha
UNIPAMPA

Prof. Me. Adriano Vogel
PUCRS

Este trabalho é dedicado à Deus e à todas as pessoas de bem que,
como profissionais, contribuem para um mundo melhor.

AGRADECIMENTOS

Em primeiro, lugar agradeço à Deus por ter me guiado durante esta longa jornada. A minha família, em especial a minha mãe, por sempre acreditar na minha capacidade. A minha namorada Gislaine Petry pelo companheirismo. Aos amigos e colegas de curso que, foram fundamentais para que eu conseguisse chegar até aqui. Gostaria também de agradecer aos meus orientadores Claudio Schepke e Raul Leiria, que sempre me ajudaram quando eu precisei. Muito obrigado a todos vocês. Levarei comigo um pouco de cada um para o resto da minha vida.

A persistência é o caminho do êxito.
(Charles Chaplin)

RESUMO

Os *hypervisors* são tecnologias fundamentais para a virtualização. Eles são os responsáveis pela, abstração do *hardware* e fornecimento de recursos virtualizados para os *guests*. Esta atividade resulta em *overhead*. Dependendo do tipo de aplicação de que será executada nos ambientes virtualizados o desempenho pode variar significativamente, de acordo com a tecnologia escolhida. O presente trabalho tem como objetivo avaliar o desempenho de dois virtualizadores diferentes: *Kernel-based Virtual Machine* (KVM) (KVM, 2016) e *VMware ESXi* (VMWARE, 2018), verificar discrepâncias e comparar seus desempenhos com o desempenho do ambiente nativo. Para realizar os experimentos e contribuir com a reprodutibilidade deste trabalho, foi desenvolvido um *script*. Experimentos indicam que o KVM possui melhor desempenho.

Palavras-chave: *Hypervisors*. Linux KVM. VMware ESXi. *Benchmark*. Automatização. HPC.

ABSTRACT

Hypervisors are fundamental technologies for virtualization. They are responsible for the hardware abstraction and the provision of virtualized resources to the guests. This activity results in overhead. The type of application that will be performed in virtualized environments, may vary the performance significantly, according to the technology chosen. The present work aims to evaluate the performance of two different hypervisors: KVM and VMware ESXi, check discrepancies and compare their performances with the performance of the native environment. To perform the experiments and contribute with the reproducibility of this work, a script was developed. Experiments indicate that KVM has better performance.

Key-words: Hypervisors. Linux KVM. VMware ESXi. Benchmark. Automation. HPC.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

CPU *Central Processing Unit*

DHCP *Dynamic Host Configuration Protocol*

GB *Gigabyte*

HD *Hard Disk*

HPC *High Performance Computing*

IP *Internet Protocol*

KVM *Kernel-based Virtual Machine*

MPI *Message Passing Interface*

NPB *The NAS Parallel Benchmarks*

OMP *Open Multi-Processing*

RAM *Random Access Memory*

SO Sistema Operacional

SSH *Secure Shell*

USB *Universal Serial Bus*

vCPU *virtual CPU*

VM *Virtual Machine*

VMM *Virtual Machine Monitor*

SUMÁRIO

.

1 INTRODUÇÃO

A computação de alto desempenho (em inglês, *High Performance Computing* (HPC)) é uma área da computação que tem por objetivo reduzir o tempo de execução de aplicações que demandam alto poder de processamento (BUYYA; VECCHIOLA; SELVI, 2013). Essas aplicações, denominadas científicas, simulam eventos naturais tais como a previsão do tempo, dinâmica dos fluídos, genética ou até mesmo gerar imagens da subsuperfície da terra para localizar petróleo (STERLING; ANDERSON; BRODOWICZ, 2017). A HPC, também pode ser utilizadas com outras tecnologias, como a virtualização (RADISKHLEBOVA et al., 2019).

A virtualização permite que usuários executem diferentes tipos de sistemas operacionais, em diferentes *Virtual Machine* (VM)s, em um mesmo servidor físico. Uma das principais características da virtualização é fazer com que *workloads* de uma máquina física possam ser divididos em partes para serem executados em vários outros *guests* (VMs). O *hypervisor*, primeira camada mais perto do *hardware* que é responsável pela virtualização, acrescenta uma sobrecarga, pelo fato de ter que adicionar uma camada de abstração entre o ambiente virtualizado e os recursos físicos do computador que ela utiliza (MALISZEWSKI et al., 2018). Vários aspectos do software e sistema operacional podem afetar o desempenho dos virtualizadores e VMs por exemplo, a maneira que o hypervisor escalona os recursos como, CPU, memória, disco, e rede (HWANG et al., 2013).

Existem diferentes *hypervisors*, por exemplo, Oracle VM, Xen, Hyper-V, KVM, VMWare, etc. Cada um deles possui suas próprias características e cada tipo de tecnologia pode realizar o gerenciamento dos recursos dos computadores de maneira diferente. Essa variedade de *hypervisors* resulta em novos desafios para compreender qual *hypervisor* escolher, para o tipo de aplicação que se deseja executar no ambiente virtualizado (HWANG et al., 2013).

De acordo com a literatura, estudos foram desenvolvidos para compreender e avaliar o desempenho das tecnologias de virtualização para HPC, como os *hypervisors* KVM e ESXi. Contudo, nos trabalhos que avaliam esses *hypervisors*, a reprodutibilidade dos experimentos é desafiadora. Nos trabalhos que comparam esses dois *hypervisors*, alguns experimentos foram realizados em ambientes específicos, outros trabalhos não comentam sobre a reprodutibilidade dos experimentos, isso torna esses trabalhos difíceis de serem reproduzidos por outros pesquisadores.

De acordo com (BARKER, 2016), reprodutibilidade de experimentos é uma das principais formas de se verificar a eficácia/eficiência de contribuições científicas. Para contribuir com a literatura e desenvolver um trabalho que seja reprodutível, foi desenvolvido um *script* que realiza a avaliação dos *hypervisors* GNU/Linux KVM e VMware ESXi, de maneira automatizada no ambiente de HPC com o auxílio do *benchmark The NAS Parallel Benchmarks* (NPB).

1.1 Objetivos

Neste trabalho, é realizada uma análise comparativa entre dois *hypervisors*: o KVM (KVM, 2016) e VMware ESXi (VMWARE, 2018), no ambiente de HPC com o auxílio da suíte de *benchmarks* NPB. O objetivo, é identificar qual *hypervisor* pode ser considerado mais adequado para aplicações científicas. Os experimentos foram realizados em diferentes cenários definidos. Assim, foi desenvolvido um *script* para automatizar algumas etapas durante a fase de experimentos e tornar o trabalho reprodutível. Por fim, é realizada a discussão dos resultados obtidos.

1.2 Organização do Documento

O presente trabalho está organizado da seguinte maneira. No Capítulo 2, são apresentados conceitos básicos necessários para a compreensão deste trabalho. No Capítulo 3, são apresentado os trabalhos utilizados como referência para o desenvolvimento deste trabalho. No Capítulo 4, é apresentada a metodologia que foi utilizada na avaliação dos experimentos, bem como as partes automatizadas pelo *script* nos experimentos e *parsing* dos resultados. No Capítulo 5, é discorrido os detalhes sobre como foi realizada a avaliação das tecnologias de virtualização. No Capítulo 6, são discutidos os resultados obtidos neste trabalho. Por fim, no Capítulo 7 as considerações finais e as direções de trabalhos futuros.

2 CONTEXTUALIZAÇÃO

High Performance Computing (HPC) é uma área da computação associada à obtenção de uma maior capacidade de processamento computacional possível em um determinado tempo. Ela envolve uma classe de computadores chamados de supercomputadores para executar uma ampla gama de problemas de computação ou aplicações (alternativamente *workloads*) o mais rápido possível. A ação de executar um *workload* em um supercomputador é amplamente denominado supercomputação e é sinônimo de HPC (STERLING; ANDERSON; BRODOWICZ, 2017).

HPC, também pode ser aplicada em ambientes de virtualização (VOGEL et al., 2016) (MALISZEWSKI; GRIEBLER; SCHEPKE, 2018), porém nesse cenário ocorre perda de desempenho por parte do virtualizador, responsável pela virtualização. Dependendo da aplicação que deseja-se executar no ambiente virtualizado, o virtualizador pode gerar um *overhead* variável. Para avaliar o desempenho dos *hypervisors*, em ambientes virtualizados para HPC (e outras áreas), utiliza-se *benchmarks*, aplicações específicas para essa finalidade.

2.1 Computação de Alto Desempenho (HPC)

A Computação de Alto Desempenho, é o termo utilizado para descrever o uso de recursos computacionais (ou supercomputadores) juntamente com técnicas de processamento, para resolver problemas complexos. HPC também pode ser chamado de supercomputação (ARORA, 2016)

2.2 Paralelismo

O paralelismo é a capacidade que um computador tem de processar múltiplas tarefas de maneira simultânea. Uma aplicação paralela é composta por processos que são executados simultaneamente, para resolver um determinado problema (PITANGA, 2004). Uma tarefa é dividida em múltiplas subtarefas através de técnicas, como divisão e conquista por exemplo, e cada uma dessas subtarefas é processada em uma *Central Processing Unit* (CPU) diferente (BUYYA; VECCHIOLA; SELVI, 2013).

Com o paralelismo, é possível proporcionar soluções viáveis para problemas através do aumento do número de CPUs em um computador e adicionando um sistema de comunicação eficiente entre eles. Os *workloads* podem ser compartilhados entre diferentes processadores. Através dessa configuração é possível elevar o desempenho de um computador e alcançar um melhor desempenho (BUYYA; VECCHIOLA; SELVI, 2013). Um sistema computacional que é capaz de executar múltiplas instruções, em múltiplos processadores, é chamado de MIMD (*Multiple-instruction, multiple-data*). Cada processador possui instruções e fluxos de dados individuais. Computadores MIMD trabalham de maneira assíncrona (BUYYA; VECCHIOLA; SELVI, 2013). Esses computadores são

classificados como memória compartilhada e memória distribuída. A Figura 1 representa um sistema computacional MIMD.

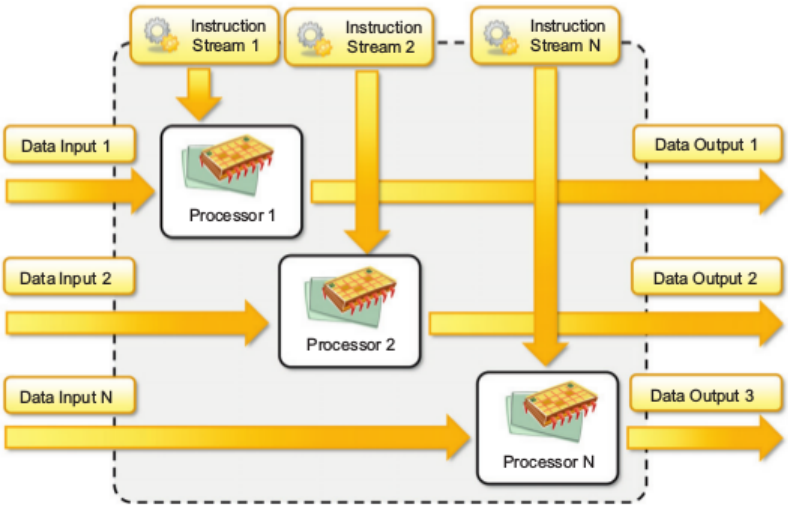

Figura 1 – Sistema Computacional MIMD (BUYYA; VECCHIOLA; SELVI, 2013).

No modelo memória compartilhada, os processadores estão conectados por uma única memória principal e todos eles tem acesso à essa memória. A comunicação entre os processadores, é realizada através da memória que é compartilhada entre eles, as modificações dos dados feitas na memória por um dos processadores, é visível para todos os outros processadores. No modelo memória distribuída, todos os processadores possuem suas próprias memórias individuais. A comunicação entre os processadores é realizada através da conexão à rede. É possível realizá-la através de trocas de mensagens entre as tarefas (BUYYA; VECCHIOLA; SELVI, 2013).

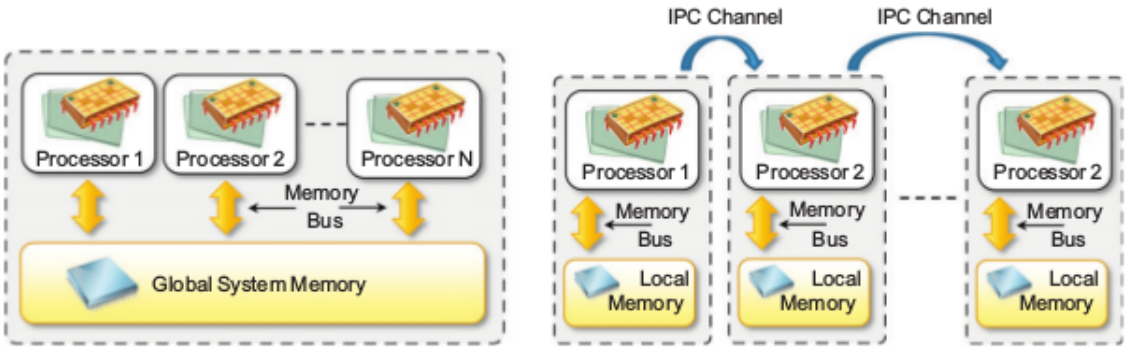

Figura 2 – Modelos de Memória (BUYYA; VECCHIOLA; SELVI, 2013).

Para executar aplicações em computadores de forma paralela é possível utilizar interfaces de programação paralela, como por exemplo o OpenMP, para memória compar-

tilhada e o MPI para memória distribuída. O OpemMP é uma interface de programação paralela que permite explorar o paralelismo através do uso da memória compartilhada. O MPI é uma interface de programação paralela onde é possível paralelizar aplicações através do uso da memória distribuída (PITANGA, 2004).

2.3 Virtualização

O elemento fundamental da virtualização é a VM, uma abstração de *software* que oferece suporte para executar diferentes sistemas operacionais e aplicações (POTHERI, 2018). O objetivo da virtualização é particionar um nó físico (ou sistema) em várias VMs independentes. Pode-se citar como exemplos, a utilização da virtualização na consolidação de servidores e criação ambientes de experimentos (NUSSBAUM et al., 2009).

Os *hypervisors*, também conhecidos como *Virtual Machine Monitor* (VMM), foram desenvolvidos para monitorar VMs. Múltiplas VMs podem ser executadas em um computador, cada uma contendo seu próprio Sistema Operacional (SO). Além de realizar o monitoramento das VMs, o *hypervisor* também é responsável pela alocação de recursos (por exemplo: CPU), do computador físico, para essas VMs (BLENK et al., 2015). Existem duas abordagens para a virtualização, o *hypervisor* tipo 1 e o *hypervisor* tipo 2 (TANENBAUM; BOS, 2015).

O *hypervisor* tipo 1 é o único *software* executando no modo privilegiado. Ele é responsável pelo suporte a múltiplas cópias do *hardware* real, máquinas virtuais, similares aos processos que um SO normal executa (TANENBAUM; BOS, 2015). O ESXI, da VMware, é um tipo de *hypervisor* tipo 1, com ele é possível criar *hardwares* virtuais diretamente sobre o *hardware* físico do computador que o *hypervisor* está instalado. Esse virtualizador não depende de um SO para se comunicar com os dispositivos de armazenamento e rede. O ESXi possui seu próprio *kernel* (VMkernel) para gerenciamento e comunicação com o dispositivo físico (CHAO, 2017). Outro exemplo de *hypervisor* tipo 1 é o KVM. Ele consiste em um módulo de *kernel* carregável, que permite ao SO Linux desempenhar a função de um *hypervisor* tipo 1 (*bare metal*) (DESAI et al., 2013). O KVM , ao contrário do ESXi, é um virtualizador *opensource*.

O *hypervisor* tipo 2, é um *software* que depende de outro SO para alocar e escalonar os recursos computacionais. Esse tipo de *hypervisor*, é semelhante a uma aplicação do usuário executando em um SO (TANENBAUM; BOS, 2015). Um exemplo de virtualizador tipo 2, é o *Virtual Box* da *Oracle* (ORACLE, 2019).

A Figura 3, ilustra os dois tipos de *hypervisors* mencionados anteriormente. À esquerda, a imagem demonstra a representação de um *hypervisor* tipo 1 e à direita representação de um *hypervisor* tipo 2.

Segundo (POTHERI, 2018) a virtualização para HPC também contribui para realizar experimentos que foram executados em outros ambientes, a figura Figura 4 ilustra algumas das contribuições da virtualização para HPC.

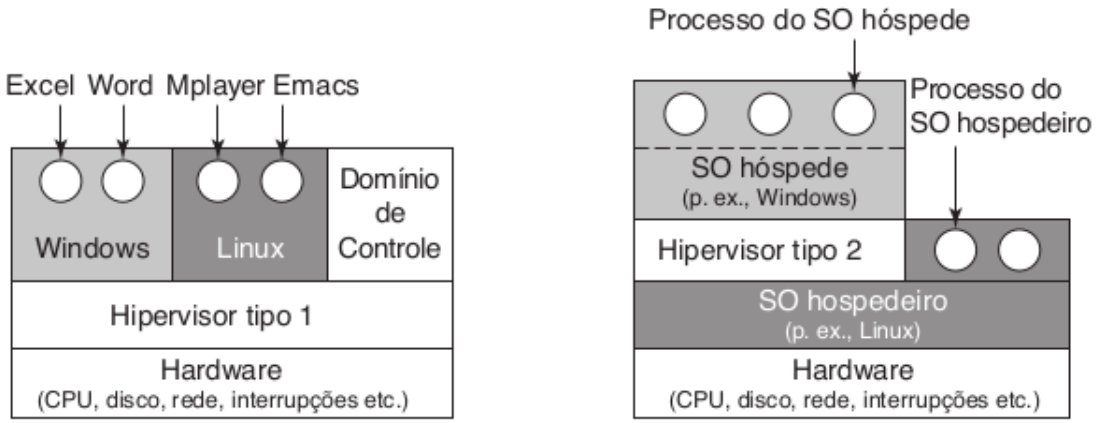

Figura 3 – *Hypervisors* tipo 1 e tipo 2 (TANENBAUM; BOS, 2015).

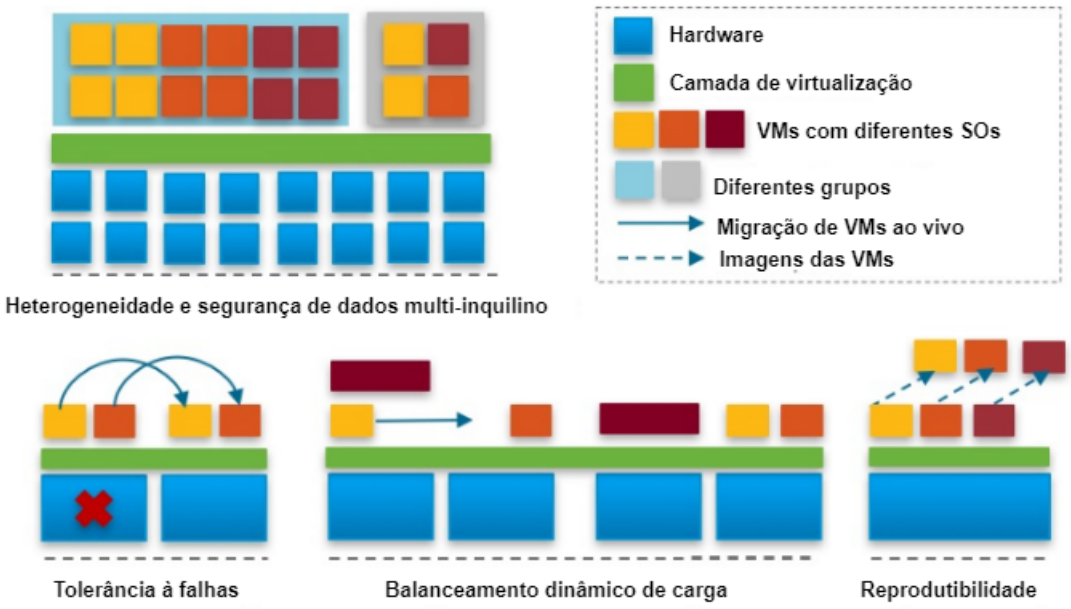

Figura 4 – Benefícios da Virtualização em HPC (DELL, 2017).

2.4 *Benchmarks*

O *benchmark* é utilizado como meio de comparação para avaliar sistemas, medindo o tempo necessário para realizar uma determinada tarefa (STERLING; ANDERSON; BRODOWICZ, 2017). Um exemplo de *benchmark* é o NPB, historicamente o NPB não foi o primeiro *benchmark* desenvolvido, antes dele existiam outros (BAILEY et al., 1991) como o *LINPACK Benchmark*.

O *benchmark Linpack* HPL (*High Parallel Linpack*), resolve uma série de equações lineares na forma matricial (DONGARRA, 1987). No *Livermore Loops*, os loops representam o tipo de núcleos de computação tipicamente encontrados em computação científica. Eles variam entre operações matemáticas comuns, como produto interno e multiplicação

de matrizes, e algoritmos de busca e armazenamento, como método de Monte Carlo (FEO, 1988).

O NPB é um conjunto de *benchmarks* desenvolvido pela *Numerical Aerodynamic Simulation* (NAS), para avaliar o desempenho de computadores esses *benchmarks* são derivados das aplicações dinâmicas dos fluídos computacionais (CFD) (BAILEY et al., 1995). Os resultados obtidos dos *benchmarks* derivados de CFD são considerados um indicador padrão para o desempenho de supercomputadores (BAILEY et al., 1995). Esses *kernels* exigem maior capacidade de processamento que os *benchmarks* anteriores como o *Livermore Loops* ou *Linpack*, portanto, são considerados mais apropriados para a avaliação de computadores paralelos (BAILEY et al., 1991). O NPB é composto por 5 *kernels* e 3 aplicações sintéticas, são eles:

- 5 *Kernels*
  - IS (*Integer Sort*), esse *kernel* ordena números inteiros utilizando o *Bucket Sort*, importante nos códigos de método de partículas (MALISZEWSKI; GRIEBLER; SCHEPKE, 2018);
  - EP (*Embarrassingly Parallel*), realiza a geração independente de valores gaussianos e variáveis aleatórias com o método *Polar Marsaglia*. O EP mede o desempenho de ponto flutuante em dados pseudo-aleatórios (MALISZEWSKI; GRIEBLER; SCHEPKE, 2018);
  - CG (*Conjugate Gradient*), gradiente conjugado, realiza o cálculo de equações matriciais (MALISZEWSKI; GRIEBLER; SCHEPKE, 2018);
  - MG (*Multigrid*), realiza a comunicação intensiva da memória. É um cálculo multigrid simplificado. Realiza testes de comunicação de dados de curta e longa distância (MALISZEWSKI; GRIEBLER; SCHEPKE, 2018);
  - FT (*Fourier Transform*), transformada de Fourier, utiliza comunicação *all-to-all*. É uma equação diferencial parcial em 3-D (MALISZEWSKI; GRIEBLER; SCHEPKE, 2018).
- 3 Aplicações sintéticas
  - BT (*Block Tridiagonal solver*);
  - SP (*Scalar Pentadiagonal solver*);
  - LU (*Lower-Upper Gauss-Seidel solver*) .

É possível escolher dentre 8 diferentes tipos de classes para executar o NPB, são elas:

- Classe S: pequena para fins de experimentos rápidos;

- Classe W: tamanho da estação de trabalho (uma estação de trabalho dos anos 90; agora provavelmente pequena demais);
- Classes A, B, C: problemas de avaliação padrão; Aumentando de tamanho de 4x indo de uma classe para a próxima;
- Classes D, E, F: grandes problemas de avaliação; Aumentando de tamanho de 16x de cada uma das classes anteriores;

2.5 Reprodutibilidade

A reprodutibilidade de experimentos é uma das principais formas de se verificar a eficácia/eficiência de contribuições científicas (BARKER, 2016). O interesse em desenvolver experimentos que sejam reprodutíveis, na área da computação, tem sido desenvolvidos, como nos trabalhos de (MANSILHA; BARCELLOS; BRASILEIRO, 2008) e (JUNIOR; CORDEIRO; GASPARY, 2018).

Figura 5 – Esquema do Processo da Iniciativa Brasileira de Reprodutibilidade (SERRA-PILHEIRA, 2019).

Segundo (NUSSBAUM, 2017), citado por (JUNIOR; CORDEIRO; GASPARY, 2018) existem alguns requisitos mínimos que são necessários para garantir que os resultados de avaliação de uma contribuição científica sejam reprodutíveis. Dentre esses requisitos, cita-se: o método, as métricas utilizadas, as condições de operação, os dados de entrada e de saída, entre outros.

Em (JUNIOR; CORDEIRO; GASPARY, 2018), a reprodutibilidade de experimentos é considerada desafiadora. A consequência negativa de experimentos não reprodutíveis são os resultados obtidos, pois não serão fiéis ao roteiro base. Avaliações executadas da mesma natureza geram resultados não passíveis de comparação. Os resultados obtidos

em (JUNIOR; CORDEIRO; GASPARY, 2018) revelam maior precisão (97,5%) em experimentos reprodutíveis.

De acordo com (BARCELLOS et al., 2006) e (URBAN; DÉFAGO; SCHIPER, 2001), citados por (MANSILHA; BARCELLOS; BRASILEIRO, 2008), a implementação e a plataforma onde os experimentos são executados, tendem a ser diferentes.

3 TRABALHOS RELACIONADOS

A proposta deste trabalho é torná-lo reprodutível, ao desenvolver um *script* para automatizar a avaliação dos *hypervisors* KVM e ESXi. Apesar da literatura contemplar trabalhos com esses *hypervisors* que mencionam a reprodutibilidade de experimentos (GUZEK et al., 2014) e (VARRETTE et al., 2013), eles foram executados em ambientes específicos. Outros, trabalhos que realizam a comparação desses *hypervisors* não mencionam a reprodutibilidade, (REDDY; RAJAMANI, 2014) (MANIK; ARORA, 2016).

Segundo (HWANG et al., 2013), o avanço do desempenho dos processadores e o uso de técnicas de virtualização tem ajudado a reduzir o custo computacional. Porém, ainda ocorre *overhead*, principalmente quando múltiplas VM estão competindo por recursos.

De acordo com (MALISZEWSKI; GRIEBLER; SCHEPKE, 2018), o *overhead* ocorre porque há adição de mais instruções que a CPU precisa gerenciar devido a virtualização completa (para VM), influenciando assim a perda de desempenho quando comparado com o ambiente nativo e com a tecnologia *lightweight* onde o *overhead* é menor (MALISZEWSKI; GRIEBLER; SCHEPKE, 2018).

Para obter o melhor desempenho das aplicações, nos ambientes virtualizados, é importante escolher qual *hypervisor* será utilizado para as determinadas aplicações, pois é ele que provê o ambiente de *hardware* virtualizado para possibilitar alocar múltiplas VMs a partir de um computador físico. Conforme (ELSAYED; ABDELBAKI, 2013) o *hypervisor* utilizado influência no desempenho do ambiente de virtualização.

Assim, foi realizada a revisão da literatura para entender como as diferentes tecnologias, *hypervisors* e contêineres, se comportam em diferentes cenários e experimentos. Os trabalhos relacionados foram organizados em 2 categorias: seção 3.1 trabalhos que realizam a avaliação do desempenho de virtualizadores, e a seção 3.2 trabalhos que comparam o desempenho entre *hypervisors* e a tecnologia *lightweight*.

3.1 Avaliação de *Hypervisors*

Nesta seção são abordados 4 trabalhos que realizam a avaliação do desempenho de *hypervisors*. Os trabalhos selecionados realizam comparações entre diferentes virtualizadores utilizando *benchmarks* e comparam os resultados com o ambiente nativo para um *baseline*.

3.1.1 SILVA, W. *et al.* 2019 - Avaliação dos Hipervisores VMware ESXi e GNU/Linux KVM para Computação de Alto Desempenho

Em (SILVA; LEIRIA; SCHEPKE, 2019), foi realizado um *survey* sobre o estado da arte, para compreender qual *hypervisor* (ESXi ou KVM) é mais adequado para aplicações em HPC. De acordo com a literatura, foram desenvolvidos trabalhos que comparam o desempenho de *hypervisors* com contêineres, outros, realizam a comparação de desempenho

entre *hypervisors*. Constatou-se que, a comparação entre o desempenho dos *hypervisors* VMware ESXI e o GNU/Linux KVM, carece de uma investigação assim como foi realizada nos trabalhos utilizados como referência neste trabalho.

Para identificar qual desses dois *hypervisors* pode ser considerado mais adequado para HPC e contribuir com a literatura, foi proposta a avaliação dessas tecnologias, com o auxílio da suíte de *benchmarks* NPB-*Message Passing Interface* (MPI). Nesse trabalho, também é proposto o desenvolvimento de um *script* para a realização dos experimentos. O objetivo do *script* é auxiliar avaliação dos *hypervisors* e tornar o trabalho reprodutível.

3.1.2 VOGEL, A. *et al.* 2016 - Medindo o Desempenho de Implantações de *Openstack, Cloudstack* e *Opennebula* em Aplicações Científicas.

Em (VOGEL et al., 2016), foi realizada a comparação entre três ambientes de nuvens computacionais (*Openstack, CloudStack e OpenNebula*), com aplicações científicas a fim de avaliar se haveriam discrepâncias. Nesse experimento, foi avaliado apenas o KVM e o ambiente nativo para a realização dos experimentos e comparação dos resultados.

No ambiente nativo e nos ambientes virtualizados, foi utilizado o SO *Ubuntu Server* 14.04 (*kernel* 3.19.0). Nos ambientes de virtualização completa, foi utilizado o KVM (versão 2.0.0). Para a avaliação de desempenho, foram realizadas as seguintes etapas:

- Executou-se o NPB-3.3 com MPI com até 16 processos (usando 2 máquinas) e *Open Multi-Processing* (OMP) com até 8 *threads*;
- Foi utilizada a classe B com os *benchmarks* BT, FT, IS, MG, CG, EP, LU e SP, os quais foram repetidos 40 vezes em cada ambiente.

Constatou-se em (VOGEL et al., 2016), que na maioria dos *benchmarks* as médias dos tempos foram semelhantes. Desta forma, comprova-se que os resultados em HPC não sofrem influência da ferramenta de gerenciamento de nuvem computacional. De acordo com o autor, não houve diferença significativa nos resultados obtidos. Pois, a tecnologia de virtualização utilizada nos ambientes dos experimentos foi a mesma, o KVM. A Figura 6 e a Figura 7 demonstram os resultados.

Na metodologia utilizada pelo autor, não consta detalhes sobre como foi realizada a execução do NPB-OMP e do NPB-MPI, para realizar os experimentos. Não há automatização dos experimentos.

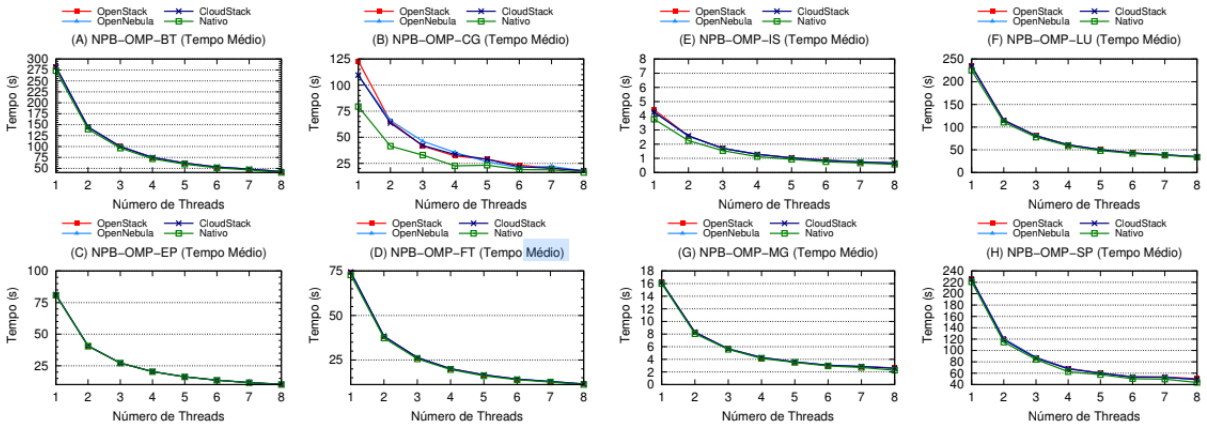

Figura 6 – NAS OMP Tempo de Execução (VOGEL et al., 2016).

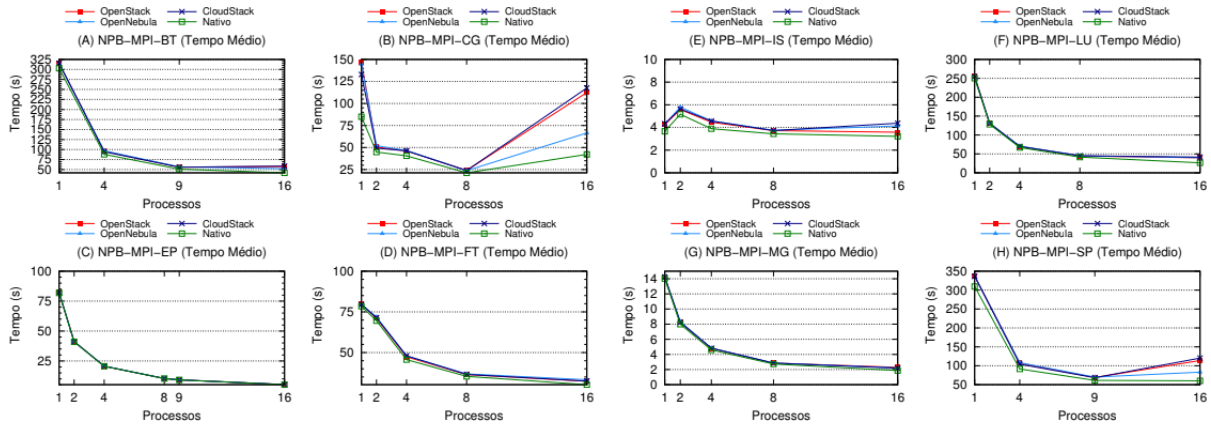

Figura 7 – NAS MPI Tempo de Execução (VOGEL et al., 2016).

3.1.3 GRANISZEWSKI, W. *et al.* 2016 - *Performance Analysis of Selected Hypervisors (Virtual Machine Monitors VMMs).*

No trabalho de (GRANISZEWSKI; ARCISZEWSKI, 2016), foi realizada a análise de desempenho dos *hypervisors*: Hyper-V; ESXi; OVM; *VirtualBox* e Xen (*software* livre). Foram utilizados diferentes *benchmarks*, para avaliar o desempenho dos seguintes componentes: CPU (*nbench*), *Random Access Memory* (RAM) (*ramspeed*), HDD (*Filebench*) e NIC (*netperf*). Os experimentos foram realizados em um servidor físico, contendo as seguintes configurações:

- Intel Core 2 Duo E8400 *dual-core* (*clock speed* de 3GHz);
- 4*Gigabyte* (GB) de DDR2 RAM;
- HD de 60 GB e 5400 rpm.

Cada *hypervisor* foi avaliado de maneira isolada, o *Hard Disk* (HD) foi formatado entre as instalações dos virtualizadores. Uma VM com o SO Ubuntu 12.04 LTS foi criada em cada *hypervisor*. O *Oracle VirtualBox*, foi o único *hypervisor* tipo 2 executado no *Windows Server* 2012. Os experimentos de desempenho foram realizados com mesmo SO, Ubuntu 12.04 LTS. Cada componente (CPU, memória RAM, HD e rede) foi avaliado separadamente. O estudo avaliou o desempenho da VM com uma e duas vCPUS. Cada VM avaliada, foi configurada com 2GB de memória RAM.

Depois de avaliar cada componente, foi realizada uma avaliação geral, para identificar o desempenho das VMs. O experimento, foi a compilação do *Kernel* Linux, que foi realizado duas vezes em cada VM e o foi obtido o tempo médio.

De acordo com os resultados obtidos nos experimentos, o ESXi foi *hypervisor* com o melhor desempenho. Os resultados nesse trabalho, também indicam que os *hypervisors* do tipo 1 geram menos *overhead*, em relação aos *hypervisors* tipo 2, devido ao acesso direto aos recursos do sistema (*hardware*). O autor do trabalho comenta que é preciso considerar, entre outros aspectos, os resultados dos *benchmarks* para escolher o tipo de *hypervisor* que deseja-se executar aplicações.

Nesse trabalho, foi avaliado o desempenho dos componentes computacionais em virtualizadores diferentes. Para os experimentos, foram utilizados *benchmarks* específicos para avaliar cada componente individualmente. No trabalho, não consta automatização dos experimentos.

3.1.4 HWANG, J. *et al.* 2013 - *A Component-Based Performance Comparison of Four Hypervisors*

Esse trabalho é utilizado como referência em (GRANISZEWSKI; ARCISZEWSKI, 2016). O trabalho de (HWANG et al., 2013) analisou 4 virtualizadores diferentes: Hyper-V, KVM, vSphere e Xen. Nos experimentos desse trabalho são alocadas 4 VMs: uma delas é um *web service* acessado por um cliente, e as outras três VMs foram criadas para gerar interferência. Esse experimento foi divido em quatro fases: *benchmark* da CPU, memória RAM, HD e rede. Durante cada fase, as três VMs executam o mesmo *benchmark* e é avaliado o impacto do desempenho na VM *web service*. O desempenho dos *hypervisors* foi avaliado da seguinte maneira:

- A CPU foi avaliada com o *bechmark Bytemark*;
- O *Ramspeed*, foi utilizado para avaliar o desempenho da cache e da largura de banda da memória;
- O *Bonnie++* é um *benchmark* de vazão de HD. E o *FileBench* gera *workloads*, que simula um servidor de emails, arquivos e servidor *web*;
- O *Netperf* é utilizado para avaliar o desempenho da rede;

- O *Application Workloads* foi avaliado com o auxílio do *sofware* (FREEBENCH, 2008), *Freebench* é um *benchmark* multi-plataforma, fornecendo codificação de áudio, compressão de dados, processamento científico, HTML, criptografia e descompactação;

- O *Multi-Tenant Interference*, foi utilizado para verificar o impacto de cada componente, CPU, RAM, HD e rede. Foi medida a resposta do HTTP enquanto cada um desses componentes foram estressados com diferentes *benchmarks*.

De acordo com resultados obtidos pelos *benchmarks*, em (HWANG et al., 2013), o *hypervisor* que teve melhor desempenho foi o vSphere. Contudo, o experimento relata que diferentes tipos de aplicações, podem necessitar de diferentes *hypervisors*. De acordo com esse trabalho, a escolha depende da aplicação que se deseja executar e quais os recursos disponíveis.

O trabalho compara diferentes *hypervisors*, através de execuções de *benchmarks*. Nesse trabalho são utilizadas aplicações sintéticas, não consta automatização dos experimentos.

3.1.5 ELSAYED, A. *et al.* 2013 - *Performance Evaluation and Comparison of the Top Market Virtualization Hypervisor*

No trabalho de (ELSAYED; ABDELBAKI, 2013), foi realizada a comparação quantitativa e qualitativa entre os *hypervisors*: VMware ESXi, Microsoft Hyper-V2008R2 e Citrix Xen Server 602. Os experimentos foram realizados utilizando o mesmo *hardware*, uma topologia de rede real, executando os mesmos métodos de avaliação nas mesmas VMs.

A comparação foi realizada por meio de experimentos, com *softwares open source* para avaliar o desempenho dos *hypervisors* sob diferentes *workloads*. Foram utilizados três servidores Intel, com a seguinte especificação: *Dual Processor* X5570 293 GHz, 24GB de memória DDR3, 5 X 146GB SAS HD, e 4 X 1Gbps portas de rede. Em cada servidor *host*, uma VM foi criada com as seguintes configurações: 4 CPU virtuais, 1 X 50GB HD, 20 GB RAM, Microsoft Windows Server 2008R2 enterprise X64, e SQL *server* 2008 R2.

O objetivo do estudo foi avaliar o desempenho dos virtualizadores, para isso foi utilizado diferentes *workloads*. A aplicação intensiva do HD é executada em uma única VM alocada em cada *hypervisor*, o *benchmark* foi realizado utilizando a aplicação sintéticas e o *software* de monitoramento de rede PRTG para a comparação dos três *hypervisors*: VMware ESXi5, Microsoft Hyper-V2008R2, e Citrix Xen Server 602. Os experimentos foram realizados sob a mesma configuração do sistema, cada cenário foi divido em dois experimentos.

- O primeiro avalia o desempenho do *hypervisor* em um banco de dados SQL;

- O segundo experimento utiliza um banco de Dados (SQL de 10GB) com *workload* intensivo. O foco é determinar o desempenho da utilização da CPU, memória consumida e utilização do HD.

Os 3 cenários nesse trabalho foram configurados da seguinte maneira:

- O primeiro cenário, com uma VM, executando o banco de dados SQL 10GB;
- O segundo cenário, com 4 VMs, cada uma executando o banco de dados SQL de 10GB em cada *hypervisor*;
- O terceiro cenário, compara o Hyper-V2008R2 e o Hyper-V2012RC.

O primeiro experimento demonstrou qual virtualizador é mais eficaz com o banco de dados SQL, através dos resultados foi constatado que o VMware ESXi5 teve melhor desempenho. No segundo experimento, o autor identificou qual virtualizador faz uso mais eficaz da CPU e memória do servidor *host* utilizando um *workload* intensivo. Constatou-se que o Citrix Xen Server 602 foi melhor, no primeiro cenário. No segundo cenário, o Hyper-V2008R2 teve o melhor desempenho. O terceiro cenário, demonstrou que o Hyper-V2012RC tem melhor desempenho no uso da memória e operações de HD, quando comparado com o Hyper-V 2008R2.

O autor do trabalho analisou 3 *hypervisors*: VMware ESXi5, Microsoft Hyper-V2008R2, e Citrix Xen Server 602 destes virtualizadores apenas o Citrix Xen Server é um *software* livre. Mais uma vez, os resultados demonstram que o desempenho dos *hypervisors*, variam de acordo com as aplicações executadas.

Os experimentos nesse trabalho foram realizados na camada de virtualização e não houve implementação de uma nuvem computacional. A avaliação de desempenho, foi realizada com o auxílio de *benchmarks*. Não consta automatização da avaliação.

3.2 Avaliação de *Hypervisors* x Contêineres

Esta subseção apresenta 3 trabalhos que realizam a comparação de desempenho entre *hypervisors* e a tecnologia de virtualização leve, contêineres.

3.2.1 RADISKHLEBOVA, A. A. *et al.* 2019 - *Study of the Possibilities of Using Virtualization Tools to Optimize the Cluster Resources Management*

No trabalho desenvolvido por (RADISKHLEBOVA et al., 2019), foi realizado um *survey* para avaliar a possibilidade de utilizar a virtualização para otimizar o gerenciamento de recursos em *clusters* para HPC. De acordo com o trabalho, o provisionamento de recursos em *clusters* para tarefas, pode causar perda de desempenho. Uma determinada

tarefa sendo executada em um *cluster*, pode utilizar o máximo de recursos disponíveis e assim, impedir a execução de outras aplicações.

Foi mencionada a utilização de contêineres, como o *Docker*. O VMware vSphere, também foi mencionado como uma possível solução, na utilização de *clusters*. O objetivo é utilizar essas tecnologias, para criar ambientes (VMs ou contêineres) onde uma determinada aplicação possar ser executada, utilizando apenas os recursos necessários.

Segundo os resultados do estudo, a utilização da virtualização pode auxiliar na configuração de recursos em *clusters*, de acordo com a necessidade da aplicação que se deseja executar. Cita-se também, o isolamento da aplicação. Se uma aplicação falha, ela não influência na performace de outras.

O foco dese trabalho, não é realizar a comparação de desempenho entre as tecnologias de virtualização. Porém, é possível verificar que, estudos estão sendo realizados para identificar qual tecnologia de virtualização utilizar em *clusters* para HPC.

3.2.2 MALISZEWSKI, A. M. *et al.* 2018 - *The Nas Benchmark Kernels for Single and Multi-Tenant Cloud Instances with LXC/KVM*

No trabalho de (MALISZEWSKI et al., 2018), foi realizada a avaliação comparativa de aplicações científicas, com apenas uma VM e múltiplas VMs, em nuvens computacionais utilizando as tecnologias de virtualização KVM e LXC, em uma nuvem privada na plataforma *Cloudstack*. Para os experimentos realizados no trabalho, utilizo-se o NPB-OMP para simular diferentes tipos de *workloads*. O *middleware* de nuvem utilizado no experimento foi o *CloudStack* 4.8. Três servidores idênticos executando o SO Ubuntu 14.04, KVM 2.0.0 e LXC 1.0.8. O *hardware* de cada servidor possui as seguintes especificações:

- Um processador Intel Xeon X5560 *quad-core* 280 GHz, com o *Hyperthreading* desabilitado intencionalmente;
- 24 GB RAM (DDR3 1333MHz);
- SATA II HD;
- Conexão de rede 1 GB.

O *software* e os *benchmarks* foram compilados com o GNU *Compiler Collection*, baseado no compilador Fortran na versão 485(*Red Hat* 485-11). A estrutura da nuvem computacional, consiste em um nó *front-end* para administração, com outros dois nós para executar os *workloads* e aplicações. O local de armazenamento, primário e secundário, são exportados do nó *front-end* via NFS e são utilizadas para armazenar as imagens das VMs, *templates* e imagens do SO. A utilização de memória (RAM), alocada nas VMs são 90% da capacidade total do *node*.

O NPB-OMP, foi executado variando o número de *threads*, para a avaliação do desempenho dos virtualizadores, que estão alocados em um ambiente de nuvem computacional. A subseção 3.2.2, demonstra a visão arquitetural do trabalho realizado pelo autor.

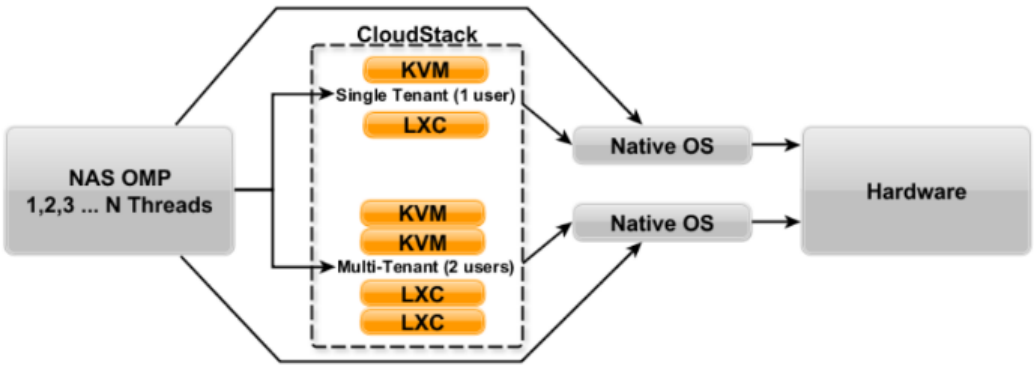

Figura 8 – Representação da Metodologia em (MALISZEWSKI et al., 2018).

O NPB-OMP, foi utilizado e compilado com a classe B, foram executados os *benchmarks* ep, ft, is, mg e cg. Dois cenários foram criados para os experimentos. No primeiro cenário, é utilizado uma instância, onde as nuvens computacionais, *CloudStack* baseadas em LXC e KVM, foram implantadas com todos os recursos do servior físico. O segundo cenário, é composto por duas instâncias de nuvens computacionais, *CloudStack* baseadas em LXC e duas nuvens computacionais *CloudStack* baseadas em KVM, o serviço oferece a metade do total de recursos do *host*.No cenário com apenas uma instância, foi executado o NPB-OMP com o número máximo de 8 *threads*, nos cenários com múltiplas instâncias (2), foi executado o NPB-OMP com o número máximo de 4 *threads*. Os experimentos, também foram executados no ambiente nativo, sem virtualização. Os cenários com uma ou múltiplas instâncias, e os ambientes são: a nuvem baseada em KVM, a nuvem baseada em LXC e o ambiente nativo.

Constatou-se que, o *overhead* no LXC é menor para esses tipos de aplicações. No primeiro cenário, ao comparar os tipos de instâncias, a tecnologia baseada em contêiner supera o KVM em 90% dos casos, sob o ambiente de nuvem privada com o *Cloudstack*. Houve apenas um caso onde o KVM foi melhor que o LXC, o experimento foi o FT definido com 8 *threads*, que de acordo com os resultados, foi uma diferença pequena. Além disso, a nuvem computacional baseada em LXC, supera o KVM em 57,5% nos resultados com múltiplas instâncias. Em 42,5% dos casos, não há diferenças significativas. É possível destacar que, com múltiplas instâncias, a nuvem baseada em KVM teve resultados similares à nuvem LXC. A utilização da memória e o acesso das aplicações, impactam con-

sideravelmente o desempenho das instâncias KVM, a CPU precisa tratar mais instruções e consequentemente mais buferizações, que causa a perda de desempenho se comparado com o ambiente nativo.

De acordo com os resultados, obtidos no trabalho do autor, a nuvem baseada em LXC é a melhor opção para aplicações científicas no cenário com apenas uma instância. Apesar da nuvem baseada em LXC, também ter apresentado bons resultados com múltiplas instâncias, a quantidade de resultados que não há diferença entre os ambientes de nuvens são de 45%. Cada nuvem computacional possui suas próprias características. A nuvem baseada em KVM, fornece o isolamento de recursos e usuários. A nuvem LXC, apresentou melhor desempenho de acordo com os resultados dos *benchmarks*. Porém, no isolamento de recursos, o nuvem *kvm* supera a LXC. Nesse trabalho, não consta a automatização dos experimentos.

3.2.3 MORABITO, R. *et al.* 2015 - *Hypervisors vs Lightweight Virtualization: A Performance Comparison*

Em (MORABITO; KJÄLLMAN; KOMU, 2015), citado por (GRANISZEWSKI; ARCISZEWSKI, 2016), é realizada uma análise comparativa entre *hypervisors* e a tecnologias *lightweight*. O *benchmark* foi realizado nos ambientes KVM, LXC, Docker, e OSv. Foi utilizado o desempenho do ambiente nativo como *baseline*, para comparar com o desempenho das demais tecnologias, a fim de avaliar o *overhead* causado por elas. Nesse trabalho, avaliou-se o desempenho dos componentes computacionais, como CPU, RAM, HD e rede. Os detalhes, sobre como foram realizados os experimentos são descritos a seguir:

- O desempenho da CPU, foi avaliado utilizando o *Y-cruncher*. A aplicação foi executada 30 vezes. Também foi utilizada a ferramenta *NBENCH*, para avaliar o desempenho da CPU. A diferença dessa ferramenta para a *Y-cruncher*, é que ela é executada em uma única *thread*. O *benchmark Linpack*, avalia o desempenho do sistema utilizando um sistema de álgebra linear simples, nesse algoritmo em foi utilizada uma matriz de tamanho N=1000, executada 15 vezes;

- Para avaliar o desempenho do HD, foi utilizado o *benchmark Bonnie++*. Foi utilizado um arquivo de 25 GB, que executa leituras e escritas em sequência;

- O *benchmark STREAM*, foi utilizado para avaliar o desempenho da memória, os resultados foram obtidos a partir de quatro operações: *Copy*, *Scale*, *Add*, e *Triad*;

- Foi utilizado o *benchmark Netperf*, para medir o desempenho da rede entre dois *hosts*. Os experimentos foram configurados para executar, de forma unidirecional, a requisição/resposta de transferência de dados com os protocolos *TCP* e *UDP* .

De acordo com os resultados obtidos, as tecnologias LXC e Docker, tiveram melhor desempenho quando comparadas com o KVM. Nesse trabalho, consta que alguns *benchmarks* foram executados repetidas vezes. Porém, não é trivial se os experimentos foram realizados manualmente ou não.

3.3 Conclusão do Capítulo

Neste capítulo, foi apresentado um total de 8 trabalhos que avaliam diferentes tecnologias de virtualização. Os autores desses trabalhos, comentam que diferentes aplicações podem necessitar de diferentes *hypervisors*.

No trabalho previamente desenvolvido, propomos a avaliação de desempenho dos *hypervisors* KVM e ESXi, a fim de verificar o *overhead* causado por essas tecnologias, no ambiente de HPC. Para isso, foi proposto o desenvolvimento de um *script* para automatizar os experimentos.

A Tabela 1 descreve resumidamente, os experimentos realizados nos trabalhos utilizados como referência neste trabalho.

Autores	**Plataformas**	**Workloads**	**Automatização**
SILVA, W. *et al.* 2019	Linux KVM VMWare ESXi	NPB-MPI	BASH Python
RADISKHLEBOVA, A. A. *et al.* 2019	LXC Docker vSphere	NC	NC
MALISZEWSKI, A. M. *et al.* 2018	KVM LXC	NPB-OMP	NC
VOGEL, A. *et al.* 2016	KVM	NPB-33, MPI	NC
GRANISZEWSKI, W. *et al.* 2016	Hyper-V ESXi OVM VirtualBox Xen	NBENCH Netperf Filebench Ramspeed	NC
MORABITO, R. *et al.* 2015	KVM LXC Docker OSv	Aplicação real Y-cruncher NBENCH Linpack Bonnie++ Netperf	NC
HWANG, J. *et al.* 2013	Hyper-V KVM vSphere Xen	Aplicação real Bytemark Ramspeed Bonnie++ FileBench Netperf	NC
ELSAYED, A. *et at.* 2013	VMware ESXi5 Citrix Xen Server 602 Hyper-V2008R2	Aplicação real Aplicações sintéticas PRTG *network monitor tools*	NC

Tabela 1 – Comparativo dos Trabalhos Relacionados.

4 METODOLOGIA

Neste capítulo é introduzida a metodologia utilizada, para realizar a avaliação das tecnologias de virtualização. O foco deste trabalho é comparar o desempenho dos virtualizadores VMware ESXi e GNU/Linux KVM, juntamente ao ambiente nativo, com o auxílio da execução de *benchmarks*. Na seção 4.1 são discorridos detalhes sobre a avaliação e a automatização. Na seção 4.3 é apresentado o ambiente computacional escolhido para a realização dos experimentos. Na seção 4.2 são demonstradas as métricas utilizadas nos experimentos.

4.1 Avaliação de Desempenho

Durante o desenvolvimento deste trabalho, foram utilizados diferentes tipos de tecnologias, como: linguagens de programação, *benchmarks* e SOs. Assim como foi realizada a leitura de diferentes trabalhos. Com uma quantidade considerável de informação disponível para trabalhar, foi criado um repositório textual na plataforma GitHub, para que fosse possível desenvolver este trabalho de maneira organizada. No repositório encontram-se informações sobre como este trabalho foi desenvolvido.

O foco deste trabalho é avaliar o desempenho dos virtualizadores GNU/Linux KVM e VMware ESXi, e compará-los com o desempenho de um ambiente não virtualizado (nativo) para um *baseline*, focando na reprodutibilidade de experimentos. De acordo com as pesquisas realizadas, nos trabalhos utilizados como referência neste trabalho não consta experimentos comparando esses dois *hypervisors* com foco na reprodutibilidade de experimentos. A identificação, do *hypervisor* mais adequado para aplicações científicas em HPC, contribuirá com a literatura. Outra motivação da escolhas dessas tecnologias, é que o *hypervisor* GNU/Linux KVM é *opensource*, ou seja, de código aberto. O *hypervisor* da VMware (ESXi) não é *opensource*.

Para a avaliação de desempenho, foi escolhida a suíte de *benchmarks* NPB3.3.1. O motivo desta escolha, é que os autores dos trabalhos relacionados ((VOGEL et al., 2016), (MALISZEWSKI; GRIEBLER; SCHEPKE, 2018)), cujos objetivos são semelhantes a este, também utilizam o NPB em seus experimentos. Além de ser considerado um conjunto de experimentos consolidados para HPC (VOGEL et al., 2016). Um *script*, de automatização, foi desenvolvido para auxiliar na fase de realização dos experimentos e também, para contribuir com a reprodutibilidade do trabalho. Por fim, gerar os gráficos a partir dos resultados obtidos das execuções do *script*.

4.2 Métricas Avaliadas

A análise de desempenho foi realizada a partir de um conjunto de métricas, também utilizadas em (ROSSO, 2015), para entender as diferenças dos resultados e fazer uma

análise comparativa dos virtualizadores. As métricas de desempenho que foram utilizadas para avaliar os dois *hypervisors* e o ambiente nativo neste trabalho são:

Média do Tempo de Execução: o tempo total de execução, que foi obtido através da média de 20 execuções, de cada *benchmark* da suíte de *benchmarks* NPB.

Desvio Padrão: para avaliar a qualidade das soluções, será considerado o desvio padrão obtido para os tempos de execução.

Speed-Up: essa métrica é utilizada para expressar quantas vezes a execução do *benchmark* paralelo ficou mais rápida que a versão sequencial. O cálculo do *speed-up* é feito pela razão entre o tempo de execução sequencial e a versão paralela. A Equação 4.1 ilustra essa razão, onde T(1) indica o melhor tempo de processamento da versão sequencial, e T(p) o tempo de processamento da versão paralela. Caso S(p) $>$ 1 a versão paralela reduziu o tempo de execução, caso S(p) $<$ 1 a versão paralela ficou mais lenta que a sequencial.

$$S(p) = \frac{T(1)}{T(p)} \tag{4.1}$$

Cada aplicação tem um número de unidades de processamento ideais para a obtenção do melhor desempenho. Ou seja, nem sempre a adição de unidades de processamento aumentará o desempenho. Para alguns *benchmarks* a melhor configuração é quando o número de unidades de processamento utilizados são múltiplos de potência de 2 (1, 2, 4, 8, ...), enquanto outros para um número quadrado de elementos de processamento (1, 4, 9, 25, ...).

Eficiência: a eficiência é uma medida que demonstra a taxa de utilização média das unidades de processamento utilizadas. O cálculo da eficiência é feito pela razão entre o *speed-up* e as unidades de processamento utilizadas. A Equação 4.2 ilustra essa razão, onde S(p) é o *speed-up* e p é o número de unidades de processamento.

$$E(p) = \frac{S(p)}{p} \tag{4.2}$$

É considerada eficiência ideal quando cada unidade de processamento fica ativa 100% durante todo o tempo de execução No entanto, geralmente essas unidades aguardam por resultados de unidades vizinhas. Isso faz com que se reduza a taxa de utilização e consequentemente a eficiência. Outro motivo é que existem partes do código que não podem ser paralelizadas, como alguma etapa de leitura de dados ou processamento de saída.

4.3 Ambiente de Execução

O ambiente de execução utilizado neste trabalho é um servidor físico localizado na UNIPAMPA em Alegrete, composto por um processador Intel Xeon E5. Para ter acesso às características do ambiente de execução, foi utilizado o comando *lscpu* no SO Ubuntu. As informações obtidas, a partir da execução deste comando, estão representadas na Tabela 2.

Componente	**Intel Xeon E5**
Modo operacional da CPU	32-bit, 64-bit
Byte Order	Little Endian
CPU(s)	6
Thread(s) per núcleo	1
Núcleo(s) por soquete	6
Soquete(s)	1
Nó(s) de NUMA	1
ID de fornecedor	GenuineIntel
Família da CPU	6
CPU MHz	1202,005
CPU max MHz	1900,0000
CPU min MHz	1200,0000
Virtualização	VT-x
Cache de L1d	32K
Cache de L1i	32K
Cache de L2	256K
Cache de L3	15360K

Tabela 2 – Ambiente de Execução.

O ambiente de execução possui um processador. O processador Intel Xeon E5-2650 possui 6 cores operando em 1.9 GHz de frequência com suporte a *Hyper-Threading*, o qual foi desabilitado intencionalmente para a realização dos experimentos. O sistema de memória do processador é formado por três níveis de cache e a memória principal:

- Cache L1 privada de 32 KB para dados e 32 KB para instruções;
- Cache L2 privada de 256 KB unificada para dados e instruções;
- Cache L3 compartilhada de 15360K unificada para dados e instruções;
- Memória principal de 16 GB.

Para obter as informações do HD, utilizado neste trabalho, foi executado o comando *smartctl -i /dev/sda*, no SO Ubuntu Server 18.04.2. Foi utilizado um HD SAMSUMG *SpinPoint* F3, com capacidade máxima de armazenamento de 500 GB e 7200 RPM.

Neste capítulo foi abordada a metodologia utilizada neste trabalho. Inicialmente foi discutido, em mais detalhes, sobre como foi realizada a avaliação do desempenho dos virtualizadores e do ambiente nativo, assim como a motivação para a automatização dos experimentos. Após, foram apresentadas as configurações do ambiente nativo onde serão realizados os experimentos, seguido pelas métricas de avaliação.

5 AVALIAÇÃO DO DESEMPENHO

Para realizar a avaliação do desempenho, dos *hypervisors* GNU/Linux e VMware ESXi em HPC, foi desenvolvido um *script* para automatizar algumas etapas durante a fase dos experimentos. O objetivo, do desenvolvimento desse *script*, foi tornar o presente trabalho automatizado e reprodutível, para que fosse possível verificar a veracidade dos resultados obtidos a partir da execução deste *script*. Nas seções a seguir, é descrita em detalhes como foi realizada as avaliações de cada ambiente.

O *script* desenvolvido teve como objetivo auxiliar a avaliação de desempenho do ambiente nativo e das tecnologias de virtualização KVM e ESXi. Para cada cenário dos experimentos, as tecnologias foram avaliadas na seguinte ordem: ambiente nativo, KVM e ESXi. Os etapas realizadas para a avaliação de desempenho são descritas a seguir.

5.1 Instalação do SO e *Hypervisors*

Primeiramente, foi necessário configurar o servidor físico. Foi realizada a instalação do SO e dos virtualizadores. O SO (Ubuntu Server) e o *hypervisor* ESXi, foram instalados em HDs diferentes, que possuem as mesmas configurações de *hardware*.

No caso do VMware ESXi, por se tratar de um produto que não é *opensource*, foi preciso realizar a inscrição no site oficial da VMware para ter acesso à uma chave de avaliação gratuita, o *VMware vSphere - Free Trial*. Durante a fase de avaliação gratuita, há algumas limitações impostas pela VMware, não há suporte técnico da empresa, é possível utilizar um número máximo de 128 *virtual CPU* (vCPU) e há um limite total de 60 dias (a partir da data de inscrição) para a sua utilização. Após a data limite, é possível renovar gratuitamente a chave por mais 60 dias.

Foram realizadas algumas etapas para as instalações do SO no ambiente nativo e do *hypervisor* ESXi. Essas etapas foram executadas em 2 momentos. No primeiro momento, foi realizada a instalação do Ubuntu Server em um dos HDs. No segundo momento, foi realizada a instalação do VMware ESXi em um outro HD. As etapas realizadas, são descritas a seguir:

- *Download* dos arquivos *.iso*, a partir do site oficial de seus respectivos desenvolvedores, (KVM, 2016) e (VMWARE, 2018), foi realizado o *download*, no formato *.iso*;

- Gravação dos Arquivos *.iso*, foi realizada com o auxílio de um dispositivo *flash drive* com 4 GB de capacidade máxima de armazenamento, para torná-lo bootável foi utilizado o *software* Rufus (RUFUS, 2019);

- Instalação do SO, após, o dispositivo bootável, o foi inserido em uma porta *Universal Serial Bus* (USB) 3.0, do servidor físico para instalação das tecnologias.

A Figura 9 ilustra os procedimentos realizados para as instalações de cada uma das tecnologias utilizadas neste trabalho.

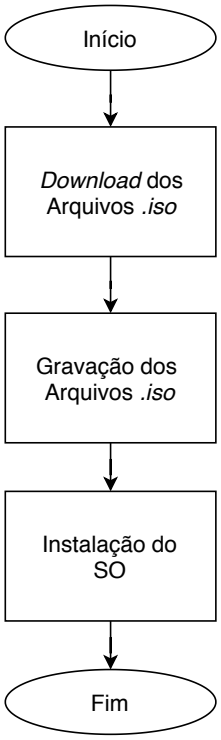

Figura 9 – Instalação dos Ambientes.

Após a instalação do SO e do *hypervisor*, iniciou-se o processo de instalação do *hypervisor* KVM. O virtualizador KVM foi instalado no ambiente nativo, diferente do ESXi, o KVM necessita de um SO para realizar a sua instalação. Foi necessário executar, alguns comandos no terminal do ambiente nativo para realizar a sua instalação. Os comandos executados são descritos a seguir:

```
1 # apt-get install qemu-kvm qemu virt-manager virt-viewer libvirt-
    bin
```

Na seguinte etapa, foi realizado o *boot* no HD contendo o Ubuntu Server. Foi instalada a interface gráfica GNOME no Ubuntu Server. Para avaliar o ambiente nativo utilizando o NPB-SER e o NPB-MPI, foi necessário realizar as seguintes configurações no ambiente: instalar o *gfortran* compilador GNU de Fortran (GFORTRAN, 2019); o (MPICH, 2019) e o comando *Make*. Para isso foram executados os seguintes comandos manualmente no terminal do Ubuntu Server (nativo).

```
1 # apt install gfortran mpich make
```

5.2 Cenários dos Experimentos

O desempenho dos virtualizadores, KVM e VMware ESXi, foi avaliada em diferentes cenários. Cada cenário utilizado durante os experimentos possui sua própria configuração, como por exemplo: quantidade de CPUs, tamanho da memória RAM, tamanho do HD e quantidade de nós (VMs). Os cenários foram definidos de acordo com as limitações do servidor físico.

Cenário 01 - *Baseline* - Execução no ambiente nativo. Neste cenário, foi avaliado o desempenho do ambiente nativo. Não foi estipulada a quantidade de memória RAM no servidor físico para cada experimento. Contudo, a quantidade de CPUs físicas foi configurada para cada experimento. Utilizou-se a capacidade total de CPU disponíveis, mas não foi ultrapassada a quantidade máxima de CPU físicas no servidor.

Cenário 02 - *Hardware* parcialmente dedicado para as VMs. Neste cenário, é avaliado o desempenho dos *hypervisors* utilizando 1 nó virtualizado. Semelhante ao cenário 1, nesta fase dos experimentos, o número de CPUs físicas também foi configurado para cada experimento. Os experimentos realizados, utilizaram um número inferior de CPUs disponíveis no servidor físico, também foi utilizado o número máximo de CPUs disponíveis. Neste cenário, foi possível variar a quantidade de memória RAM utilizada. O *hardware* deste cenário é considerado parcialmente dedicado, pois haverá compartilhamento de recursos (CPU) no experimento utilizando um nó com 6 cores (capacidade máxima do servidor físico). Pelo menos um core é compartilhado entre o nó e o servidor.

A Tabela 3, demonstra as configurações utilizadas nos cenários 1 e 2. Ressalta-se que, nos experimentos realizados no ambiente nativo, não foi estipulada a quantidade de memória RAM utilizada, apenas no Cenário 2.

Experimento	Nós	Cores	RAM	HD
#1	1	1	4 GB	32 GB
#2	1	1	16 GB	32 GB
#3	1	2	4 GB	32 GB
#4	1	2	16 GB	32 GB
#5	1	3	4 GB	32 GB
#6	1	3	16 GB	32 GB
#7	1	4	4 GB	32 GB
#8	1	4	16 GB	32 GB
#9	1	5	4 GB	32 GB
#10	1	5	16 GB	32 GB
#11	1	6	4 GB	32 GB
#12	1	6	16 GB	32 GB

Tabela 3 – Cenários 1 e 2.

Cenário 03 - *Hardware* dedicado para as VMs. Neste cenário, foram utilizadas 2 VMs para os experimentos. A quantidade de CPUs, configurada para as VMs nos

experimentos, não ultrapassou a capacidade máxima do servidor físico. Por isto, neste cenário, é dito que o *hardware* é dedicado para as VMs. A Tabela 4, demonstra as variações das configurações utilizadas durantes os experimentos no Cenário 3.

Experimento	Nós	Cores	RAM	HD
#13	A	1	4 GB	32 GB
	B	1	4 GB	32 GB
#14	A	2	4 GB	32 GB
	B	2	4 GB	32 GB

Tabela 4 – Cenário 3 - *Hardware* dedicado para as VMs Variando o Número de Cores.

Cenário 04 - *Hardware* compartilhado entre as VMs e/ou o SO hospedeiro. Neste cenário, foram utilizadas 2 VMs para os experimentos. A quantidade de CPUs, configurada para as VMs nos experimentos, intencionalmente ultrapassou a capacidade máxima do servidor. Por isto, neste cenário, é dito que o *hardware* compartilhado entre as VMs, pelo menos 1 *core* foi compartilhada entre mais de uma VM. A Tabela 5, demonstra as variações das configurações utilizadas durantes os experimentos no cenário 4.

Experimento	Nós	Cores	RAM	HD
#15	A	4	4 GB	32 GB
	B	4	4 GB	32 GB

Tabela 5 – Cenário 4 - *Hardware* Compartilhado.

5.3 Execução Automatizada do *NAS Parallel Benchmark*

A fim de automatizar o processo de avaliação, foi desenvolvido um *script* com as linguagens de programação Bash e Python, para auxilar a execução do NPB. O código do *script* foi armazenado em um repositório do GitHub (github.com/RDLeiria/WillianSoares), para ter o controle das versões e realizar o *clone* em outros ambientes para os experimentos.

O *script* contém um arquivo de entrada que é utilizado para configurar o experimento a ser realizado. A partir deste arquivo de entrada, configurado manualmente, outro arquivo *script* recebe todos os parâmetros necessários para sua execução. Em seguida, o *script* realiza as seguintes etapas:

- Cria Diretórios para os Resultados. Esta função do código cria um diretório para cada experimento. Dentro desse diretório, é criado um outro diretório para armazenar o resultado do ambiente avaliado. Por exemplo: /Experimento1/NativoResultado;
- *Download* do NPB. Esta etapa do *script* realiza o *download* do arquivo compactado do NPB e em seguida realiza a sua descompactação;

- Seleciona os *Benchmarks*, no código, esta é uma que lista contém todos os 8 *benchmarks* do NPB. A partir desta lista, é executada a seguinte função do *script*;

- Compila os *Benchmarks*. Todos os *benchmarks* são compilados de acordo com a quantidade de nós, CPUs e classe, definidos préviamente no arquivo de entrada;

- Executa os *Benchmarks*. Cada um dos *benchmarks* compilados é executado *n* vezes (de acordo com a configuração do arquivo de entrada). Para cada nova execução, de um *benchmark*, a saída (*raw*) é armazenada no final em um arquivo *.txt*. São gerados arquivos individuais para cada *benchmark* executado.

A Figura 10 ilustra as partes automatizadas pelo *script* para auxiliar a avaliação de desempenho do ambiente nativo e dos virtualizadores.

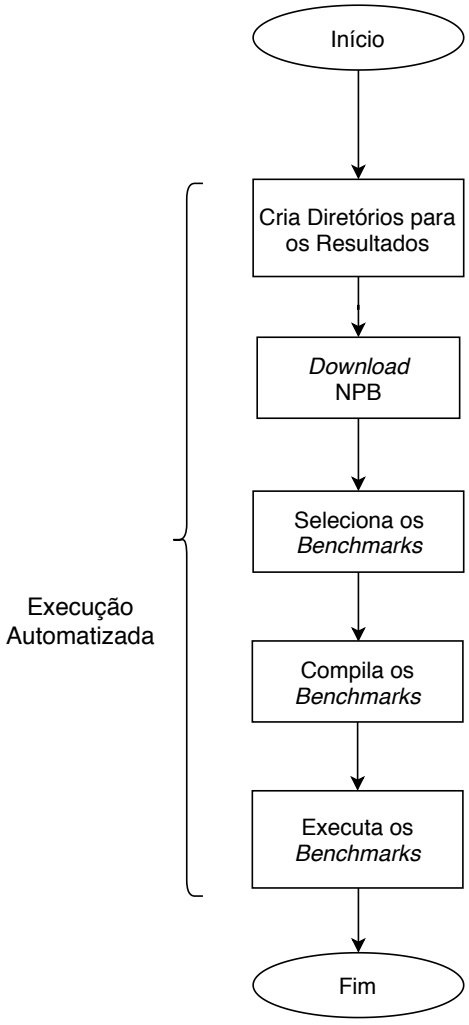

Figura 10 – *Script* para Execução Automatizada do NPB.

5.4 Avaliação de Desempenho do Ambiente Nativo

Para realizar a avaliação do desempenho do ambiente nativo utilizando o *script* que executa de maneira automatizada o *benchmark* NPB, foi necessário realizar algumas configurações, que estão detalhadas nesta seção. O ambiente nativo foi avaliado utilizando duas versões do NPB. A primeira versão é a *serial* e a segunda versão utiliza o MPI. Os experimentos realizados com essas versões dos *benchmarks*, são comentadas a seguir.

5.4.1 *Baseline*

Para realizar a avaliação do desempenho do ambiente nativo, foi desenvolvido um *script* específico para executar o NPB a versão serial. Esse *script*, também está no repositório do GitHub, e contém as mesmas funções do *script* automatizado. Contudo, não é necessário configurar um arquivo de entrada para executar o NPB-SER. É necessário editar o código fonte, para informar a quantidade de repetições para os *benchmarks* e a classe que deseja-se compilar e executar os *benchmarks*.

Durante a fase de experimentos no ambiente nativo, utilizando o NPB-SER, não foi definida a quantidade de CPUs utilizadas, a quantidade de memória RAM, e o tamanho do HD. Utilizou-se as configurações do servidor físico sem realizar nenhuma alteração.

5.4.2 MPI

Para avaliar o desempenho do ambiente nativo cutilizando o NPB-MPI, foi criada uma VM. Nesta fase dos experimentos, não foi adicionada nenhuma outra VM. Porém, houve variações quanto ao número de CPUs utilizadas para executar o NPB. A seguir, é discutido como foi realizada a avaliação de desempenho desse experimento.

- *Boot* no Ambiente, em primeiro lugar, o servidor físico é inicializado no ambiente nativo realizar a avaliação do desempenho;
- Configuração do Ambiente, são instalados programas que são necessários para utilizar a versão NPB-MPI;
- *Download* do *Script*, com o ambiente para avaliação devidamente configurado, nesta etapa é realizada o *download* do código fonte do *script*. Com essa finalidade, foi utilizado o comando:

```
1     $ git clone https://github.com/RDLeiria/WillianSoares
```

- Configuração do Arquivo de Entrada, para o *script* executar de maneira automatizada o NPB, é necessário configurar o arquivo de entrada que envia para o *script* todos os parâmetros que ele necessita para executar. Os parâmetros são: classe dos

benchmarks, quantidade de CPUs por VM, total de CPUs utilizadas, quantidade de VMs, quantidade de repetições para os *benchmarks*, nome do ambiente de execução e identificação do experimento;

- Execução do *Script*, nesta fase do experimento, as etapas são as mesmas ilustradas na Figura 10;
- *Commit* dos Resultados, após a avaliação do ambiente, em um determinado cenário (número de VMs, quantidade de CPUs utilizadas), os resultados são gravados manualmente no repositório do GitHub;
- Avaliar Outro Cenário, para avaliar outro cenário, foi necessário configurar manualmente o arquivo de entrada novamente e executá-lo. Após avaliar todos os cenários definidos para este ambiente, os experimentos neste ambiente foram finalizados.

A Figura 11, ilustra os passos descritos anteriormente para a avaliação do desempenho do ambiente nativo utilizando o NPB-MPI.

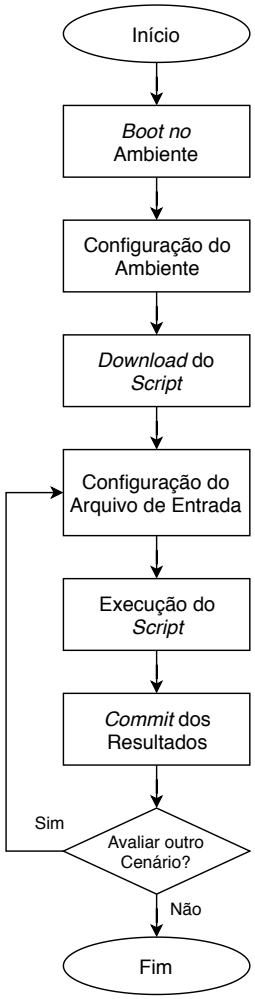

Figura 11 – Etapas da Avaliação com o NPB-MPI.

Parsing dos Resultados, para gerar os gráficos é necessário configurar um outro arquivo de entrada manualmente, a partir deste arquivo é calculada a média dos tempos de execuções de cada *benchmark* para cada ambiente (Nativo, KVM e ESXi) e por fim, são gerados os gráficos.

5.5 Avaliação de Desempenho dos *Hypervisors*

As etapas, para a avaliação de desempenho de ambas tecnologias de virtualização, são semelhantes. Cada uma das tecnologias foi avaliada isoladamente. Apenas quando todos os cenários definidos foram executados com sucesso, iniciou-se o processo de avaliação do outro *hypervisor*. A seguir, são descritas as etapas realizadas, para avaliar o desempenho destes *hypervisors*.

Algumas etapas são similares, às etapas realizadas na avaliação de desempenho do ambiente nativo NPB-MPI.

Todas estas etapas mencionadas, também estão presentes na avaliação de desempenho os *hypervisors* KVM e ESXi. Os detalhes de cada etapa são exatamente os mesmos mencionados na subseção 5.4.2. As diferenças dos experimentos realizados nesses ambientes estão na criação das VMs.

- Configuração da(s) VM(s), para cada experimento, é configurada manualmente as VMs. Essa configuração das VMs, inclui: definir o número de CPUs, definir a quantidade de memória RAM e definir o tamanho do HD virtualizado;

- Criação da(s) VM(s), nesta fase do experimento, é realizada a instalação do Ubuntu Server 18.04.02 nas VMs a partir de um arquivo no formato *.iso*.

Posteriormente à criação da(s) VM(s), foram executadas as seguintes etapas: *download* do *script*, configuração do arquivo de entrada, execução do *script* e *commit* dos resultados. Quando todos os *benchmarks* foram executados para o *hypervisor*, em um determinado cenário, o experimento era concluído. No caso de avaliação de outro cenário, utilizando o mesmo *hypervisor*, foi necessário reconfigurar as VMs manualmente.

Nos casos em que, diferentes cenários utilizavam a mesma quantidade de VMs, não foi necessário realizar novamente a instalação do SO. Contudo, foi realizada a instalação manual do SO nas VMs, nos cenários em que foi necessário adicionar mais de uma VM. A Figura 12, ilustra as etapas que foram realizadas, para avaliar o desempenho dos *hypervisors*.

Para realizar a comunicação entre as VMs, foram criadas e configuradas manualmente, chaves públicas entre as VMs dos experimentos. Antes de executar o *script*, foi necessário criar um arquivo *host.txt*, para armazenar os *Internet Protocol* (IP)s das VMs nos experimentos.

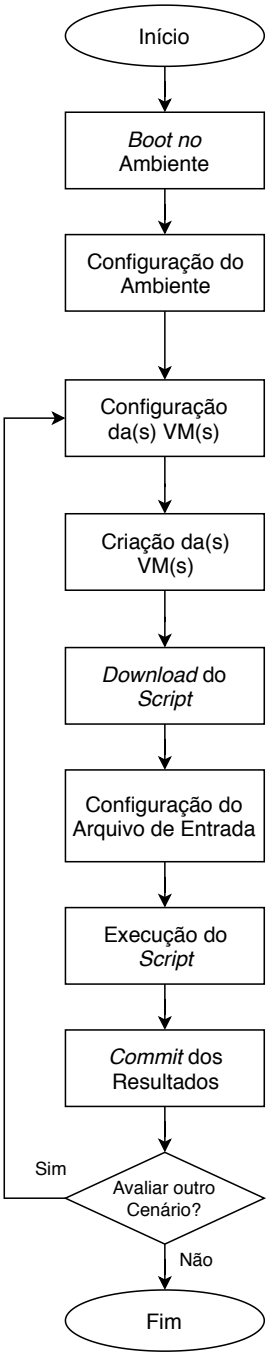

Figura 12 – Avaliação dos *Hypervisors*.

5.5.1 GNU/Linux KVM

As VMs neste ambiente foram criadas com o auxílio da interface gráfica *Virt Manager*, utilizando o *Ubuntu Desktop* 18.04.2. Inicialmente, tentou-se realizar a instalação de n VMs via linha de comando para os experimentos, utilizando o *Ubuntu Server* 18.04.2. Foi necessário dedicar tempo para aprender como realizar essa tarefa e surgiram alguns impasses durante as tentativas de execução desta tarefa. Por esse motivo, foi instalada a versão gráfica no ambiente nativo e o *software Virt Manager*, para que fosse possível criar as VMs .

Em todas as VMs, utilizadas nos experimentos, foi instalado o SO *Ubuntu Server* 18.04.2. Neste caso, não foi instalada a interface gráfica para a realização dos experimentos. Todos os comandos, foram executados pelo terminal na própria VM ou através do *Secure Shell* (SSH).

Para os experimentos nesse ambiente, foi utilizado o protocolo de rede *Dynamic Host Configuration Protocol* (DHCP), nesse tipo de protocolo o servidor distribui IPs para os *hosts* (VMs). Para acessar via SSH as VMs nesse ambiente, foi necessário realizar o acesso SSH no ambiente nativo primeiro.

5.5.2 VMware ESXi

Para avaliar o desempenho, do *hypervisor* ESXi, foram criadas VMs com o auxílio da interface gráfica *Web Client* ESXi. Esta interface, foi acessada a partir de um *browser* através o endereço IP, configurado para o *hypervisor* durante a sua fase de instalação. No *browser*, foi necessário informar o usuário e senha para ter acesso ao VMware ESXi.

Nas VMs foi instalado o SO Ubuntu Server 18.04.2, sem interface gráfica. Semelhante ao que foi realizado, nos experimentos utilizando o KVM, todos os comandos foram executados em uma VM a partir do terminal ou através do SSH.

A rede foi configurada, com o protocolo de internet versão 4 (IPV4). Para cada VM criada, durante a fase de instalação do SO, a VM era configurada e recebia um endereço IPV4 definido manualmente. Com essa configuração, foi possível acessar as VMs através do SSH.

5.6 Geração dos Gráficos

Os resultados foram obtidos, a partir das execuções do *script* em cada cenário mencionado anteriormente. Após a avaliação destes cenários, iniciou-se a fase de criação dos gráficos. Para esta etapa, também foi desenvolvido um *script* que auxilia a criação dos gráficos, a partir dos resultados dos experimentos. Semelhante ao *script* para a execução do NPB, para gerar os gráficos também é necessário configurar um arquivo de entrada, este arquivo é responsável pelo envio dos parâmetros para o *script* principal. As etapas para o *parsing* dos resultados, são descritas em detalhes a seguir.

- **Configuração do Arquivo de Entrada**: é um arquivo desenvolvido na linguagem de programação BASh, nele deve ser informado a classe do experimento, a quantidade de CPUs por VM e a identificação (número) do experimento. É necessário executar o arquivo via terminal;

- **Execução do *Script***: o *script* em Python, recebe os parâmetros do *script* em BASh, realiza a leitura dos arquivos de saída *raw .txt*, dos resultados das execuções

do NPB e calcula a média do tempo de execução de cada *benchmark*, para cada ambiente, em um determinado cenário e grava em um arquivo *.csv*;

- **Criação dos Graficos**: a partir da leitura, dos arquivos *.csv*, o *script* em Python cria os gráficos e salva no formato *.pdf*.

A Figura 13, ilustra como foi realizada a etapa de geração dos gráficos, a partir dos resultados obtidos, com o auxílio de *script* .

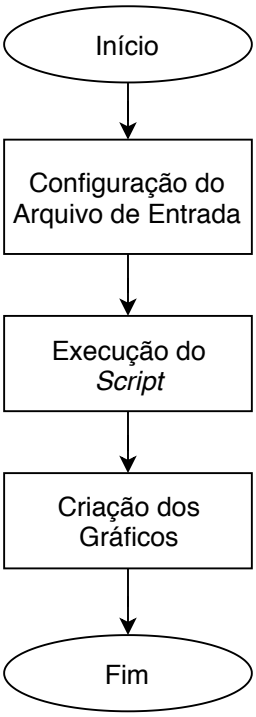

Figura 13 – Geração dos Gráficos.

5.7 Disponibilização do Código

Com o intuito de contribuir com a comunidade e continuar o desenvolvimento deste trabalho, o código fonte do projeto, foi disponibilizado em um repositório público na plataforma GitHub. Todos os *scripts* desenvolvidos e utilizados durante o desenvolvimento deste trabalho, podem ser acessados no repositório do GitHub. O acesso ao repositório pode ser realizado através deste endereço https://github.com/RDLeiria/WillianSoares

6 ANÁLISE DOS RESULTADOS

Com o objetivo de identificar qual *hypervisor* pode ser considerado mais adequado para ambientes de HPC. Foram gerados gráficos que comparam os resultados obtidos a partir da execução do *benchmark* NPB-MPI. Os gráficos apresentam o tempo médio de execução dos ambientes (Nativo, KVM e ESXi) com diferentes cenários de execuções. Para identificar os ambientes avaliados, variou-se a quantidade de nós, número de CPUs e memória RAM. Nos resultados dos gráficos para o ambiente nativo foi utilizada a cor cinza, a cor vermelha para o KVM e a cor azul para o ESXi.

Os Experimentos 1 e 2 foram realizados utilizando apenas 1 nó, nos virtualizadores e 1 CPU. A memória RAM foi configurada para 4 GB no Experimento 1 e 16 GB para o Experimento 2. Os gráficos do Experimento 1, representados pela Figura 14, demonstram o tempo médio das execuções dos *benchmarks*.

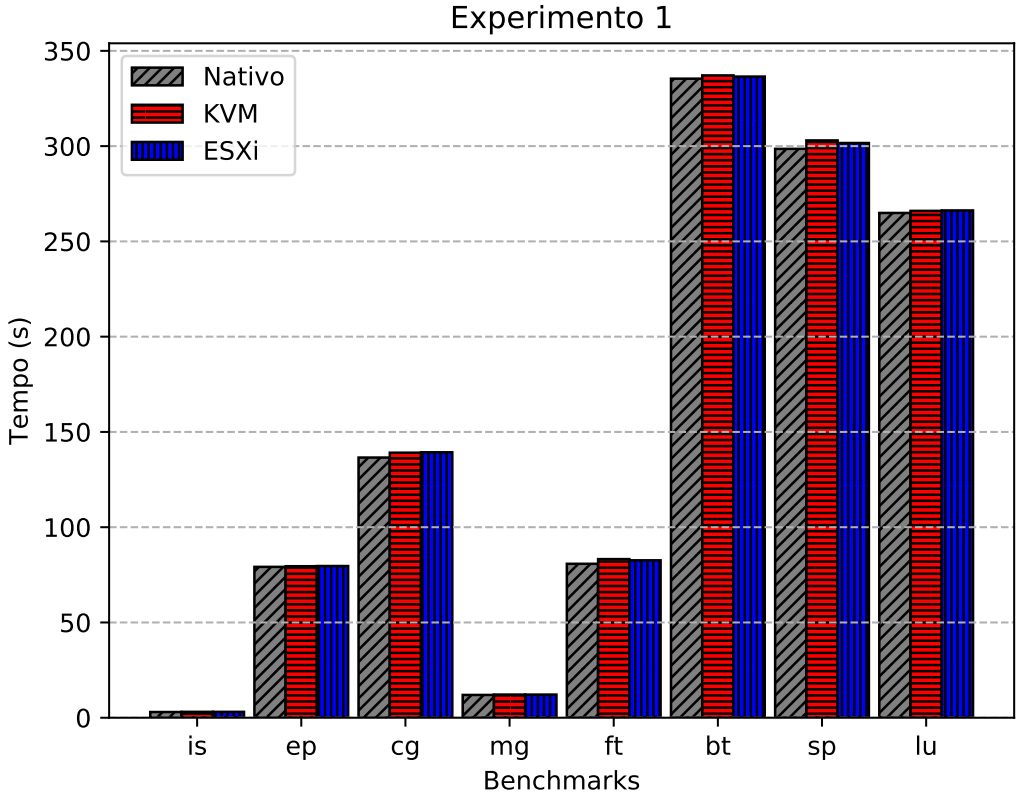

Figura 14 – Experimento 1 - 1 Nó, 1 CPU, 4 GB RAM.

Os resultados obtidos no Experimento 1, representados pela Tabela 6, indicam que o *hypervisor* KVM teve melhor desempenho nas execuções dos *benchmarks* EP, CG e LU, eles utilizam menos memória ao executar comparado aos demais *benchmarks*. O *hypervisor* ESXi, teve melhor desempenho que o KVM ao executar os *benchmakrs* IS, FT, BT e SP, esses *benchmarks* necessitam utilizar maior quantidade de memória para comunicação. O cenário 1 demonstra que o ESXi tem melhor desempenho.

	IS	EP	CG	MG	FT	BT	SP	LU
Nativo	3,00	79,18	136,55	11,96	80,79	335,39	264,92	298,56
KVM	3,09	79,52	139,10	12,14	83,24	337,10	266,24	302,99
ESXi	3,08	79,61	139,31	12,14	82,61	336,49	266,01	301,62

Tabela 6 – Tempo Médio -Experimento 1 - 1 Nó, 1 CPU, 4 GB RAM.

	IS	EP	CG	MG	FT	BT	SP	LU
Nativo	0,01	0,12	1,85	0,02	0,35	1,21	1,58	1,30
KVM	0,02	0,13	2,40	0,00	0,09	1,28	1,42	0,94
ESXi	0,02	0,56	1,91	0,03	0,41	0,61	1,34	0,78

Tabela 7 – Desvio Padrão - Experimento 1 - 1 Nó, 1 CPU, 4 GB RAM.

Os gráficos gerados, a partir das execuções dos *benchmarks* nesse cenário, são representados pela Figura 15.

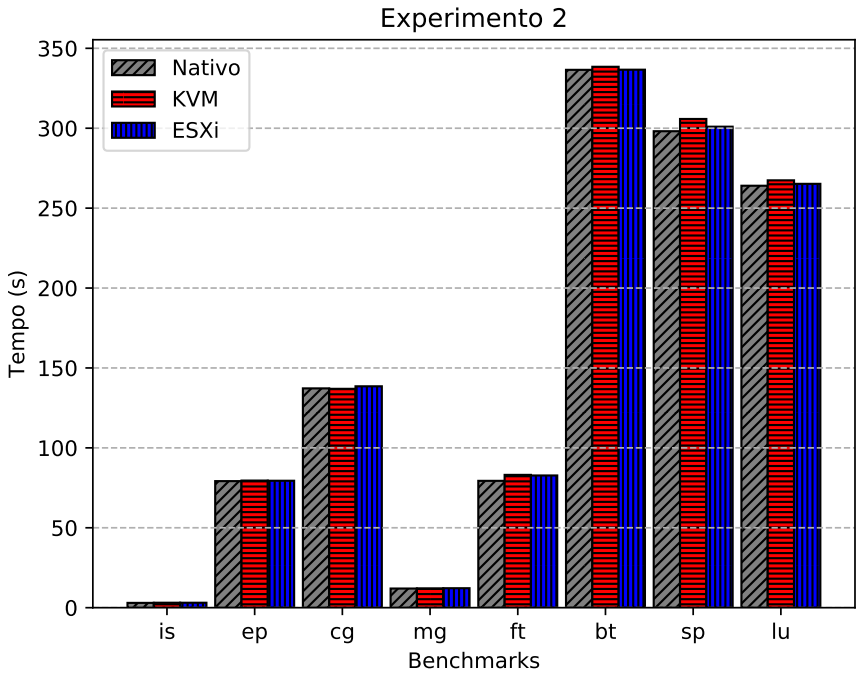

Figura 15 – Experimento 2 - 1 Nó, 1 CPU, 16 GB RAM.

De acordo com os resultados obtidos no Experimento 2, representados pela Tabela 8. Foi possível identificar, uma anomalia ao avaliar o *hypervisor* KVM. O tempo médio de execução do *benchmark* CG foi menor que tempo obtido no ambiente nativo, isso pode ter ocorrido porque esse *benchmark* necessita utiliza uma quantidade baixa de memória (SUBHLOK; VENKATARAMAIAH; SINGH, 2001). O KVM teve melhor desempenho na execução do *benchmark* MG. Nos demais casos, o ESXi teve desempenho superior ao KVM, executando os *benchamrks* IS, EP, FT, BT, SP e LU. Nesse cenário o nó com o ESXi também foi melhor na utilização de CPU e memória.

	IS	EP	CG	MG	FT	BT	SP	LU
Nativo	3,00	79,18	137,21	11,98	79,36	336,53	298,12	264,01
KVM	3,10	79,53	136,94	12,09	83,09	338,44	305,78	267,39
ESXi	3,09	79,42	138,51	12,17	82,72	336,61	300,95	265,20

Tabela 8 – Tempo Médio - Experimento 2 - 1 Nó, 1 CPU, 16 GB RAM.

	IS	EP	CG	MG	FT	BT	SP	LU
Nativo	0,01	0,18	2,61	0,02	0,26	1,05	1,48	1,06
KVM	0,07	0,16	1,25	0,00	0,46	1,08	9,44	1,36
ESXi	0,01	0,12	2,18	0,02	0,35	0,81	0,94	0,69

Tabela 9 – Desvio Padrão - Experimento 2 - 1 Nó, 1 CPU, 16 GB RAM.

Os Experimentos 3 e 4, foram executados utilizando 1 nó, com 2 CPUs. O tamanho da memória RAM, foi configurado para 4 GB no Experimento 3 e 16 GB no Experimento 4. Alguns *benchmarks* da suíte NPB não executam com uma determinada quantidade de CPUs, por exemplo, os *benchmarks* SP e BT, que necessitam um número quadrado de CPUs para executar (BAILEY et al., 1995). Por esse motivo, o tempo médio de execução desses *benchmarks* não estão presentes nos gráficos dos experimentos 3 e 4, que utilizam a mesma quantidade de CPUs. A Figura 16, representa o tempo médio das execuções de cada *benchmark* neste cenário.

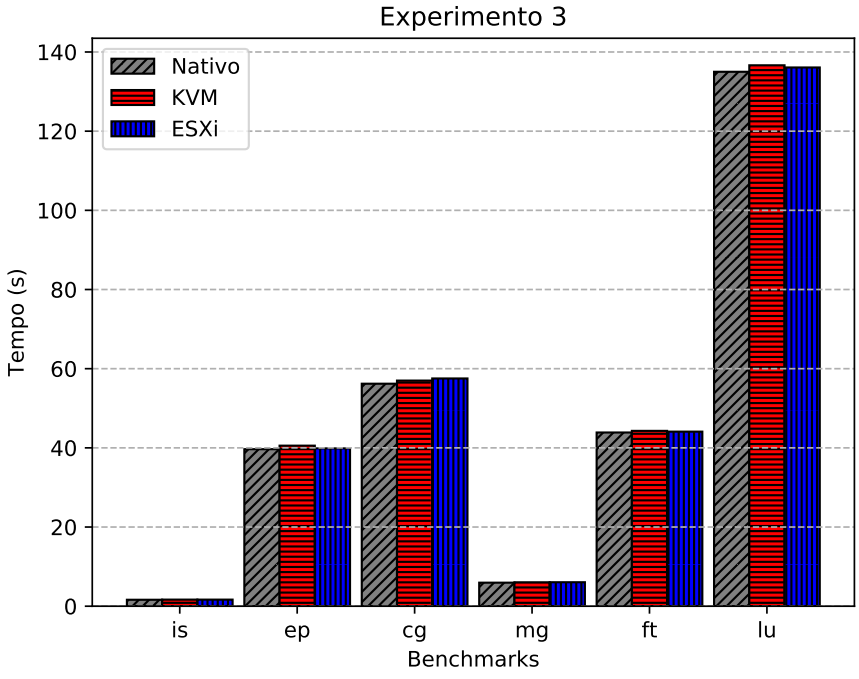

Figura 16 – Experimento 3 - 1 Nó, 2 CPUs, 4 GB RAM.

Os resultados obtidos no Experimento 3, representados pela Tabela 10, indicam que o KVM teve melhor desempenho ao executar os *benchmarks* MG, FT e LU, que realizam a comunicação *ring pattern*. O *hypervisor* ESXi, teve melhor desempenho ao executar os *benchmarks* IS , EP e CG, cada *benchmark* se comunica de maneira diferente, o IS realiza a comunicação *all-all*, o EP a comunicação é considerada insignificante e no CG a comunicação é unidirecional (SUBHLOK; VENKATARAMAIAH; SINGH, 2001).

	IS	**EP**	**CG**	**MG**	**FT**	**LU**
Nativo	1,62	39,66	56,20	5,94	43,87	135,00
KVM	1,66	40,55	56,99	6,03	44,29	136,63
ESXi	1,66	39,86	57,52	6,05	44,09	136,08

Tabela 10 – Tempo Médio - Experimento 3 - 1 Nó, 2 CPUs, 4 GB RAM.

	IS	**EP**	**CG**	**MG**	**FT**	**LU**
Nativo	0	0,12	0,43	0	0,12	0,78
KVM	0	3,19	0,08	0	0,11	0,50
ESXi	0	0,19	0,18	0	0,14	0,34

Tabela 11 – Desvio Padrão - Experimento 3 - 1 Nó, 2 CPUs, 4 GB RAM.

A Figura 17, representa a média dos tempos de execuções dos *benchmarks* neste cenário.

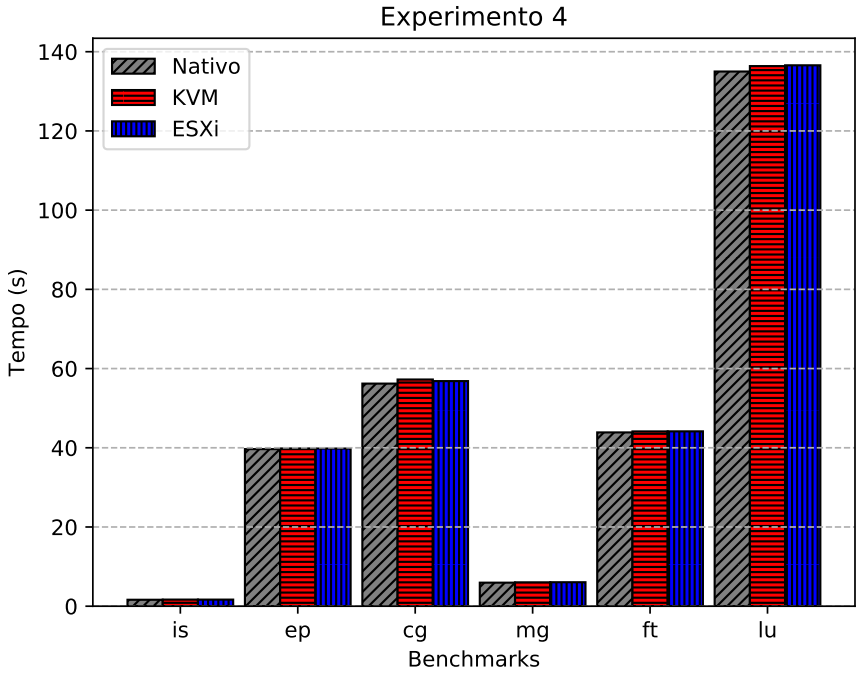

Figura 17 – Experimento 4 - 1 Nó, 2 CPUs, 16 GB RAM.

Nos resultados obtidos no experimento 4, representados pela Tabela 12, o KVM teve melhor desempenho nas execuções dos *benchamrks* MG, FT, e LU. Nas execuções dos outros *benchmarks* IS, EP e CG. O *hypervisor* ESXi teve melhor desempenho. Os Experimentos 3 e 4 não sofrem influência do tamanho da memória principal, pois o KVM teve melhor desempenho nos *benchmarks* que realizam a comunicação *ring pattern* e o ESXi nos demais casos.

	IS	**EP**	**CG**	**MG**	**FT**	**LU**
Nativo	1,62	39,66	56,20	5,94	43,87	135,00
KVM	1,66	39,82	57,23	6,02	44,13	136,36
ESXi	1,66	39,80	56,83	6,05	44,14	136,57

Tabela 12 – Tempo Médio - Experimento 4 - 1 Nó, 2 CPUs, 16 GB RAM.

	IS	**EP**	**CG**	**MG**	**FT**	**LU**
Nativo	0	0,12	0,43	0	0,12	0,78
KVM	0	0,08	0,43	0	0,14	0,57
ESXi	0	0,21	0,42	0	0,14	0,57

Tabela 13 – Desvio Padrão - Experimento 4 - 1 Nó, 2 CPUs, 16 GB RAM.

O Experimento 5 representado pela Figura 18 e o experimento 6 representado pela Figura 19, possuem uma particularidade. Nestes experimentos, a VM foi configurada com um número ímpar (3) de CPUs. O único *benchmark* que pode ser executado utilizando um número ímpar de CPUs é o EP, por esse motivo, ele é o único *benchmark* que consta nos gráficos destes experimentos. O tempo médio de execuções, obtido nos dois *hypervisors*, é quase semelhante ao resultado do ambiente nativo. Contudo, é possível notar que, os *hypervisors* causam *overhead*.

De acordo com os resultados obtidos, nos Experimento 5 e 6, o *hypervisor* KVM teve melhor resultado que o ESXi. O *benchmark* EP apresenta uma comunicação irrelevante nos experimentos, porém mantém a taxa de utilização da CPU alta (SUBHLOK; VENKATARAMAIAH; SINGH, 2001).

	EP
Nativo	26,40
KVM	26,59
ESXi	26,64

Tabela 14 – Tempo Médio - Experimento 5 - 1 Nó, 3 CPUs, 4 GB RAM.

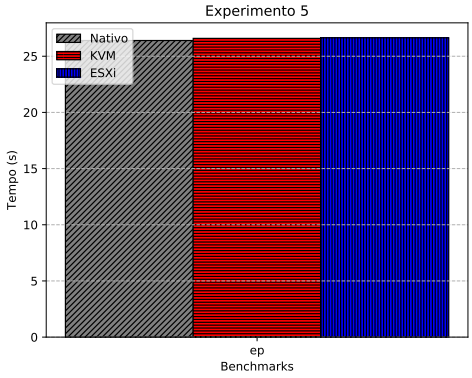

Figura 18 – Experimento 5 - 1 Nó, 3 CPUs, 4 GB RAM.

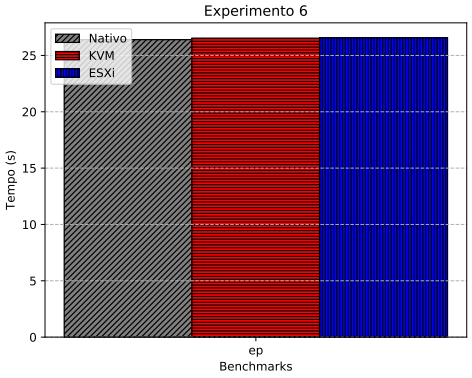

Figura 19 – Experimento 6 - 1 Nó, 3 CPUs, 16 GB RAM.

	EP
Nativo	0,03
KVM	0,15
ESXi	0,12

Tabela 15 – Desvio Padrão - Experimento 5 - 1 Nó, 3 CPUs, 4 GB RAM.

	EP
Nativo	26,40
KVM	26,53
ESXi	26,56

Tabela 16 – Tempo Médio - Experimento 6 - 1 Nó, 3 CPUs, 16 GB RAM.

	EP
Nativo	0,03
KVM	0,05
ESXi	0,06

Tabela 17 – Desvio Padrão - Experimento 6 - 1 Nó, 3 CPUs, 16 GB RAM.

O nó utilizado nos experimentos 7 e 8, foi configurado com 4 CPUS. Nos resultados obtidos no experimento 7, representados pela Tabela 18, o *hypervisor* KVM teve melhor desempenho nas execuções dos *benchmarks* IS, EP, CG, SP e LU, todos esses *benchmarks* mantém a execução da CPU perto do 100% e utilizam uma quantidade menor de memória que os *benchmarks* MG e BT (SUBHLOK; VENKATARAMAIAH; SINGH, 2001). O ESXi teve melhor desempenho ao executar os *benchmarks* MG, FT e BT. O KVM tem melhor desempenho quando é necessário utilizar a CPU e o ESXi quando é necessário utilizar a memória, neste experimento.

	IS	EP	CG	MG	FT	BT	SP	LU
Nativo	0,90	19,83	30,93	3,40	24,46	91,28	100,96	72,73
KVM	0,98	19,90	31,55	3,50	25,13	91,82	102,38	73,47
ESXi	1,00	20,15	31,65	3,49	25,07	91,67	102,41	73,66

Tabela 18 – Tempo Médio - Experimento 7 - 1 Nó, 4 CPUs, 4 GB RAM.

	IS	EP	CG	MG	FT	BT	SP	LU
Nativo	0,00	0,04	0,12	0,01	0,05	0,11	0,27	0,25
KVM	0,00	0,03	0,12	0,00	0,05	0,06	0,24	0,17
ESXi	0,05	0,62	0,09	0,00	0,04	0,08	0,16	0,30

Tabela 19 – Desvio Padrão - Experimento 7 - 1 Nó, 4 CPUs, 4 GB RAM.

Nos resultados obtidos no Experimento 8, representados pela Tabela 20, ocorreu um fato inédito. Foi a primeira vez que um dos *hypervisors*, o ESXi, teve melhor desempenho ao executar todos os *benchmarks* da suíte NPB-MPI. A adição de memória RAM teve impacto no desempenho do *hypervisor* ESXi, melhorando o seu desempenho em todos os *benchmarks*.

	IS	EP	CG	MG	FT	BT	SP	LU
Nativo	0,90	19,83	30,93	3,40	24,46	91,28	100,96	72,73
KVM	0,99	19,96	31,62	3,50	25,09	91,90	102,62	73,65
ESXi	0,98	19,92	31,57	3,49	25,06	91,76	102,28	73,52

Tabela 20 – Tempo Médio - Experimento 8 - 1 Nó, 4 CPUs, 16 GB RAM.

	IS	EP	CG	MG	FT	BT	SP	LU
Nativo	0,00	0,04	0,12	0,01	0,05	0,11	0,27	0,25
KVM	0,00	0,05	0,06	0,00	0,04	0,13	0,29	0,17
ESXi	0,00	0,04	0,09	0,00	0,04	0,10	0,15	0,26

Tabela 21 – Desvio Padrão - Experimento 8 - 1 Nó, 4 CPUs, 16 GB RAM.

A Figura 20, representa a média dos tempos de execuções dos *benchmarks* obtidos neste cenário. O gráfico gerado, a partir da média dos tempos de execuções dos *benchmarks* no Experimento 8, é representado pela Figura 21.

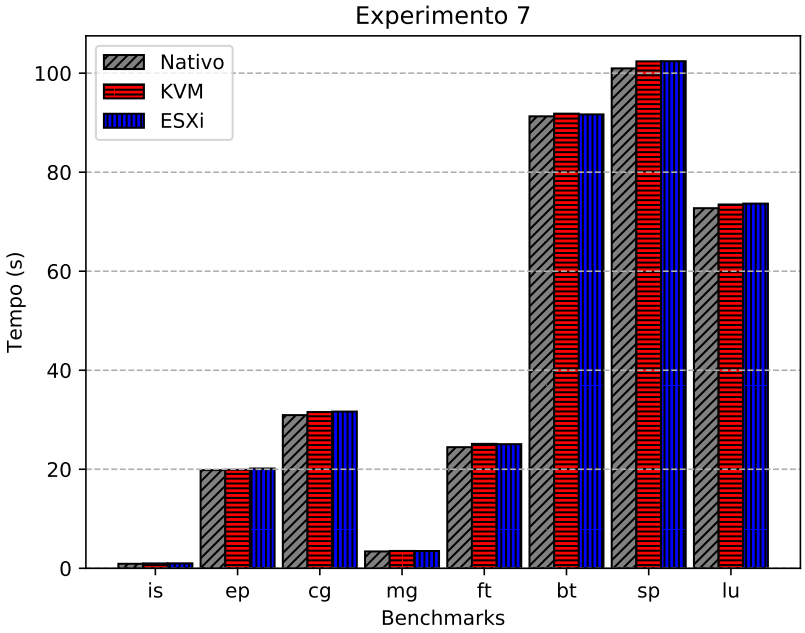

Figura 20 – Experimento 7 - 1 Nó, 4 CPUs, 4 GB RAM.

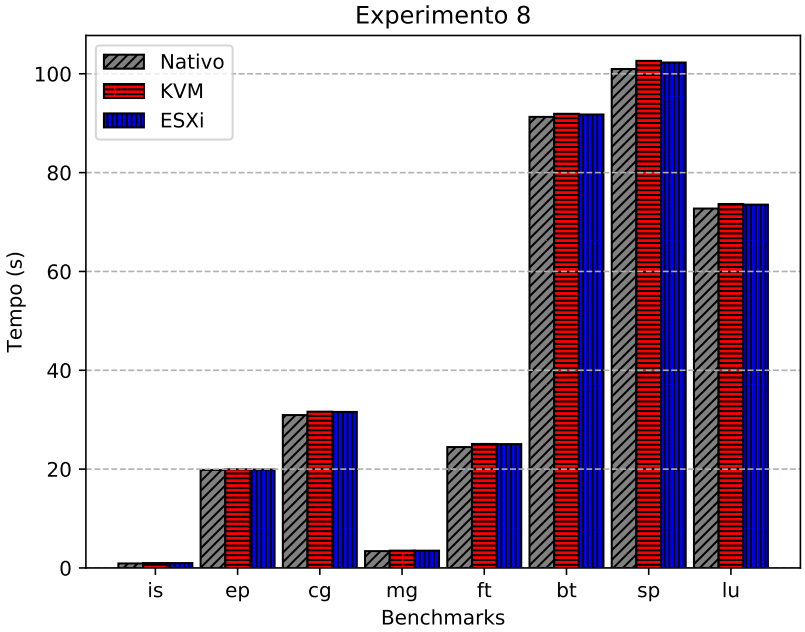

Figura 21 – Experimento 8 - 1 Nó, 4 CPUs, 16 GB RAM.

No Experimento 9, representado pela Figura 22 e no Experimento 10, representado pela Figura 23. A VM foi criada com 5 CPUs. Assim como nos Experimentos 5 e 6, onde a VM foi configurada com 3 CPUs, neste experimento o único *benchmark* que foi executado foi o EP. O *hypervisor* KVM tende a ter melhores resultados, para aplicações onde a comunicação é considerada irrelevante.

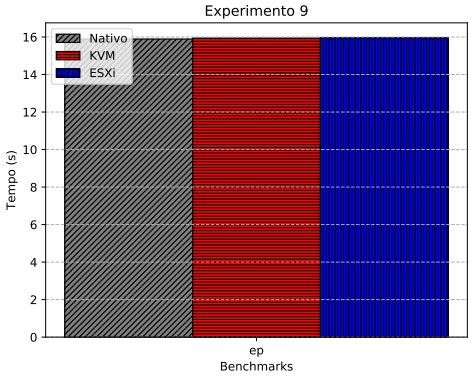

Figura 22 – Experimento 9 - 1 Nó, 5 CPUs, 4 GB RAM.

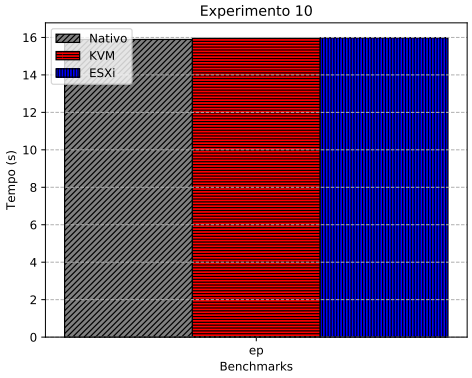

Figura 23 – Experimento 10 - 1 Nó, 5 CPUs, 16 GB RAM.

	EP
Nativo	15,89
KVM	15,94
ESXi	15,95

Tabela 22 – Tempo Médio - Experimento 9 - 1 Nó, 5 CPUs, 4 GB RAM.

	EP
Nativo	0,06
KVM	0,03
ESXi	0,05

Tabela 23 – Desvio Padrão - Experimento 9 - 1 Nó, 5 CPUs, 4 GB RAM.

	EP
Nativo	15,89
KVM	15,95
ESXi	15,98

Tabela 24 – Tempo Médio - Experimento 10 - 1 Nó, 5 CPUs, 16 GB RAM.

	EP
Nativo	0,06
KVM	0,07
ESXi	0,07

Tabela 25 – Desvio Padrão - Experimento 10 - 1 Nó, 5 CPUs, 16 GB RAM.

No experimento 11, representado pela Figura 24, a VM foi configurada com 6 CPUs. Neste experimento, os únicos *benchmarks* que podem ser executados utilizando esta quantidade de CPUs é o EP e o LU. O tempo médio de execução dos *benchmarks* nos *hypervisors*, está representado pela Tabela 26. No Experimento 11 o KVM obteve melhor desempenho que o ESXi, ao executar os dois *benchmarks*, EP e LU, que fazem

uso intensivo da CPU e baixa utilização de memória.

	EP	LU
Nativo	13,24	53,63
KVM	13,32	54,44
ESXi	13,37	54,60

Tabela 26 – Tempo Médio - Experimento 11 - 1 Nó, 6 CPUs, 4 GB RAM.

	EP	LU
Nativo	0,03	0,10
KVM	0,05	0,20
ESXi	0,04	0,18

Tabela 27 – Desvio Padrão - Experimento 11 - 1 Nó, 6 CPUs, 4 GB RAM.

A Figura 24, representa a média do tempo de execução dos *benchmarks* neste experimento.

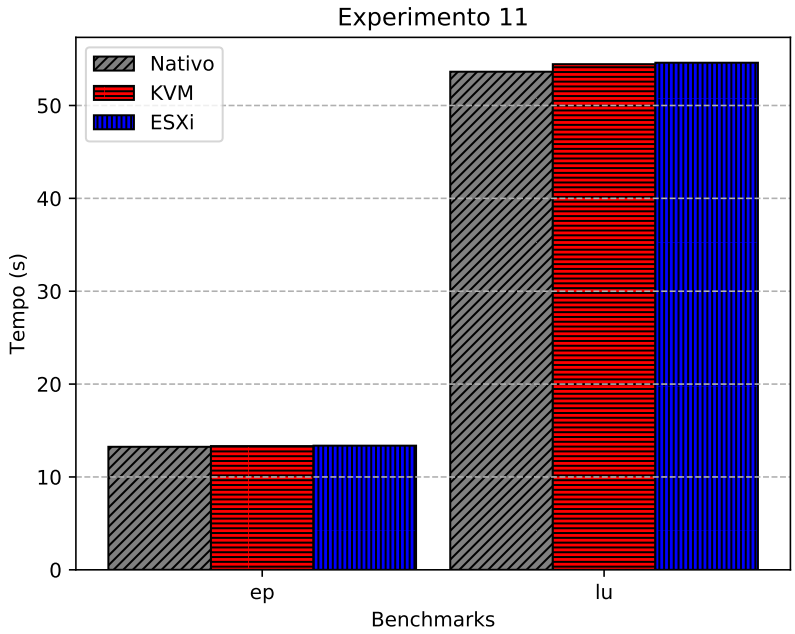

Figura 24 – Experimento 11 - 1 Nó, 6 CPUs, 4 GB RAM.

No experimento 12, o tempo médio obtido nos *hypervisors*, está representado pela Tabela 28. Foram executados apenas os *benchmarks* EP e LU para realizar a avaliação do desempenho dos *hypervisors*. Semelhante ao Experimento 11, no Experimento 12, o KVM teve melhor desempenho ao executar os dois *benchmarks*, EP e LU. A Figura 25, representa o tempo médio de execuções dos *benchmarks* no Experimento 12.

De acordo com os resultados, obtidos nos experimentos realizados com apenas um nó, foi possível identificar que o desempenho dos *hypervisors* variava de acordo com

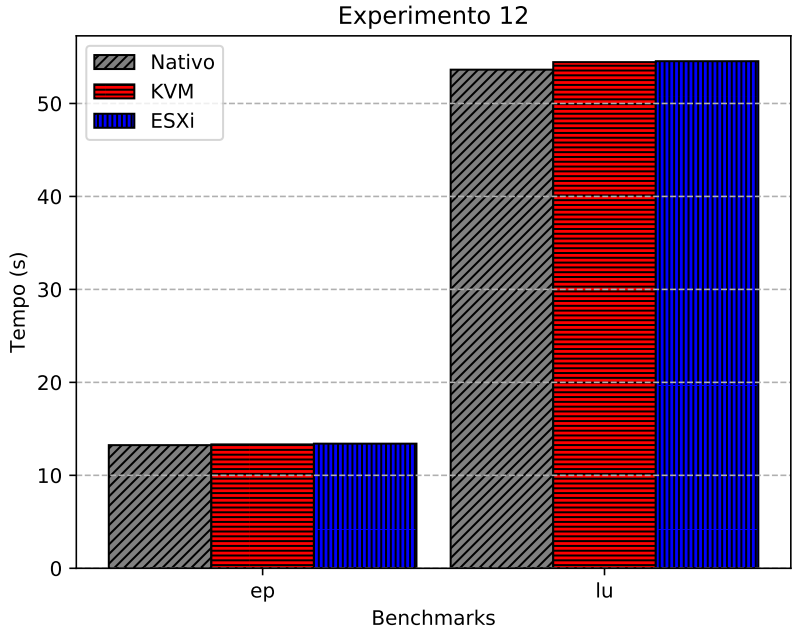

Figura 25 – Experimento 12 - 1 Nó, 6 CPUs, 16 GB RAM.

	EP	**LU**
Nativo	13,24	53,63
KVM	13,32	54,44
ESXi	13,40	54,54

Tabela 28 – Tempo Médio - Experimento 12 - 1 Nó, 6 CPUs, 16 GB RAM.

	EP	**LU**
Nativo	0,03	0,10
KVM	0,05	0,20
ESXi	0,04	0,18

Tabela 29 – Desvio Padrão - Experimento 12 - 1 Nó, 6 CPUs, 16 GB RAM.

o experimento. Contudo, algumas diferenças foram observadas. No Experimento 8, o *hypervisor* ESXi teve desempenho superior que o KVM, ao executar todos os 8 *benchmarks* do NPB. Foi possível identificar também, que o KVM teve melhor desempenho nos *benchmarks* que fazem baixa utilização da memória, enquanto o *hypervisor* ESXi tem desempenho superior ao executar os *benchmarks* que necessitam utilizar mais a memória do sistema.

Nos Experimentos 13, 14 e 15, é utilizado o ambiente nativo com um nó físico, como *baseline*. No experimento 13, representado pela Figura 26, foram criados dois nós. Esses nós, possuem configurações de *hardware* iguais: 1 CPU e 4 GB de RAM. O *hypervisor* com o melhor desempenho foi o ESXi, comparado com o ambiente nativo, o *overhead* também foi menor nas execuções dos *benchmarks* BT e SP, comparados ao KVM. O KVM, teve

melhor desempenho que o ESXi, ao executar o *benchmark* FT. O experimento revelou que o ESXi tem melhor desempenho com múltiplos nós, fazendo a utilização eficiente dos recursos.

	EP	**MG**	**FT**	**BT**	**SP**	**LU**
KVM	61,11	6,20	51,1	337,68	305,27	150,82
ESXi	39,73	6,16	63,58	336,53	301,07	148,61

Tabela 30 – Tempo Médio - Experimento 13 - 2 Nós, 1 CPU (cada nó), 4 GB RAM.

	EP	**MG**	**FT**	**BT**	**SP**	**LU**
KVM	13,38	0,01	0,52	1,54	3,47	0,70
ESXi	0,08	0,01	5,81	0,77	1,35	0,43

Tabela 31 – Desvio Padrão - Experimento 13 - 2 Nós, 1 CPU (cada nó), 4 GB RAM.

No experimento 14, representado pela Figura 27, o *hypervisor* KVM teve melhor desempenho ao executar os *benchmarks* MG, BT, SP e LU. Neste experimento, o KVM superou os resultados obtidos pelo ESXi, que teve desempenho superior ao executar os *benchmarks* EP e FT. O aumento no número de cores nos nós melhora o desempenho do *hypervisor* KVM.

	EP	MG	FT	LU
KVM	19,90	3,59	28,93	75,82
ESXi	19,94	3,62	35,38	76,09

Tabela 32 – Tempo Médio - Experimento 14 - 2 Nós, 2 CPUs (cada nó), 4 GB RAM.

	EP	MG	FT	LU
KVM	0,03	0,01	0,08	0,30
ESXi	0,04	0,00	2,78	0,22

Tabela 33 – Desvio Padrão - Experimento 14 - 2 Nós, 2 CPUs (cada nó), 4 GB RAM.

No experimento 15, representado pela Figura 28, os dois *hypervisors* apresentaram resultados distintos. A execução dos *benchmarks* EP e FT, demonstraram que o *hypervisor*

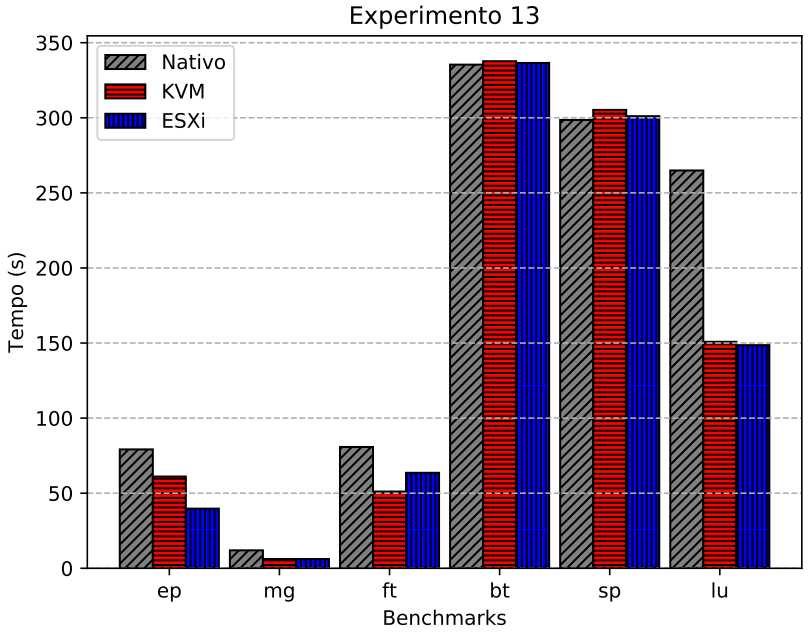

Figura 26 – Experimento 13 - 2 Nós, 1 CPU (cada nó), 4 GB RAM.

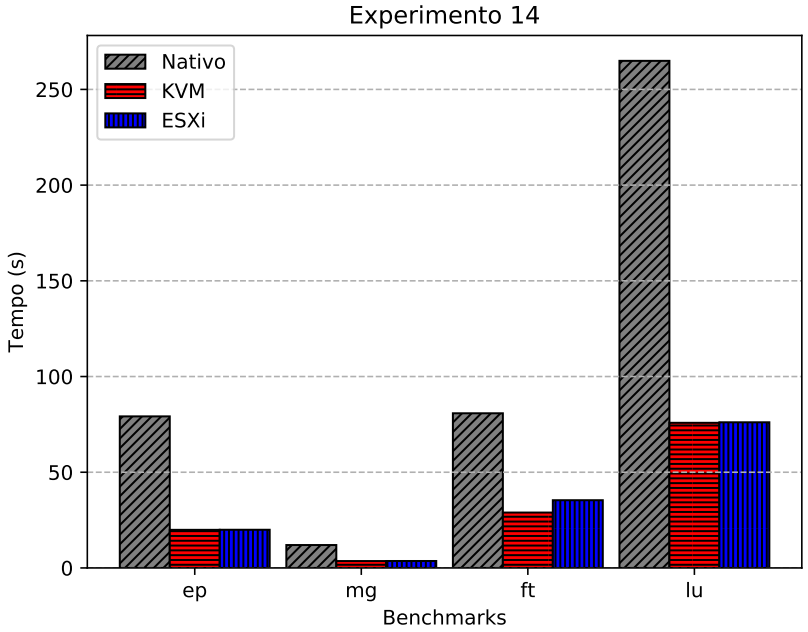

Figura 27 – Experimento 14 - 2 Nós, 2 CPUs (cada nó), 4 GB RAM.

ESXi causa *overhead* menor que o KVM. Contudo, nos demais *benchmarks* (MG, BT, SP e LU), o KVM tem desempenho superior ao ESXI.

Neste experimento, onde foi alocado mais CPUs para os nós, que a capacidade do servidor físico. O KVM supera o ESXi, na maioria dos casos. Quando há mais nós que recursos disponíveis (CPU), o KVM tem desempenho superior quando comparado ao

ESXi.

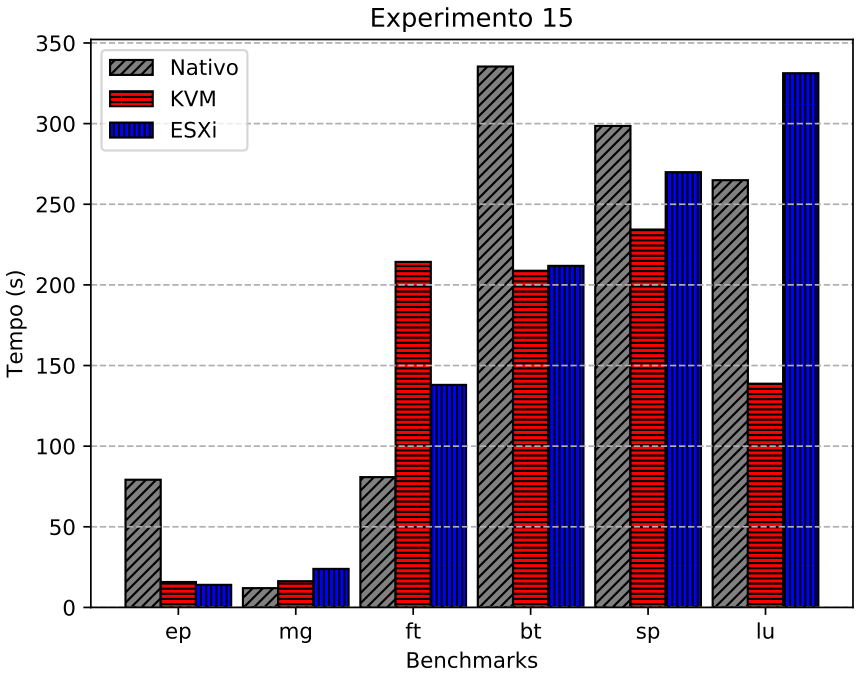

Figura 28 – Experimento 15 - 2 Nós, 4 CPUs (cada nó), 4 GB RAM.

	EP	**MG**	**FT**	**BT**	**SP**	**LU**
KVM	15,77	16,27	214,25	208,76	234,32	138,73
ESXi	14,00	23,89	137,99	211,79	269,89	331,24

Tabela 34 – Tempo Médio - Experimento 15 - 2 Nós, 4 CPUs (cada nó), 4 GB RAM.

	EP	**MG**	**FT**	**BT**	**SP**	**LU**
KVM	0,89	1,55	24,09	3,51	6,54	1,59
ESXi	0,13	1,09	6,91	5,28	6,39	7,52

Tabela 35 – Desvio Padrão - Experimento 15 - 2 Nós, 4 CPUs (cada nó), 4 GB RAM.

7 CONCLUSÃO

O presente trabalho, teve como objetivo, realizar a comparação dos *hypervisors* GNU/Linux KVM e VMware ESXi. Apesar da literatura, abordar trabalhos que realizam a comparação de desempenho de diferentes tecnologias de virtualização, não foram encontrados trabalhos que comparam o desempenho entre esses dois *hypervisors* e que sejam reprodutíveis.

Os *hypervisors* KVM e ESXi foram avaliados no ambiente de HPC, para identificar qual tecnologia pode ser considerada mais adequada na execução de aplicações científicas. A avaliação foi realizada com o auxílio de um *script*, desenvolvido para automatizar a fase de realização dos experimentos e contribuir com a reprodutibilidade do trabalho.

Algumas etapas dos experimentos presentes neste trabalho não ficaram totalmente reprodutíveis. Alguns aspectos como a criação e configuração de cenários (n VMs) e criação automatizada dos gráficos precisam ser revisadas em trabalhos futuros. Por outro lado, como um todo foram descritos detalhadamente os processos durante o desenvolvimento deste trabalho para que seja possível compreender as metodologias utilizadas e a maneira que os resultados foram obtidos.

Conclui-se que a partir dos experimentos realizados nos cenários, ambas tecnologias podem ser consideradas para utilização para virtualização em HPC. As limitações do *hardware* do servidor físico tiveram influência nas escolhas dos cenários. Algumas diferenças de tempos de execuções obtidas nos experimentos podem parecer irrelevantes, porém se considerados sua utilização em larga escala, pode haver uma diferença significativa na escolha do *hypervisor*. Para os testes realizados neste trabalho, os experimentos demonstraram que KVM foi o virtualizador que teve melhor desempenho.

REFERÊNCIAS

ARORA, R. **Conquering Big Data with High Performance Computing**. [S.l.]: Springer, 2016. Citado na página 27.

BAILEY, D. et al. **The NAS parallel benchmarks 2.0**. [S.l.], 1995. Citado 2 vezes nas páginas 31 e 65.

BAILEY, D. H. et al. The nas parallel benchmarks. **The International Journal of Supercomputing Applications**, Sage Publications Sage CA: Thousand Oaks, CA, v. 5, n. 3, p. 63–73, 1991. Citado 2 vezes nas páginas 30 e 31.

BARCELLOS, M. P. et al. Bridging the gap between simulation and experimental evaluation in computer networks. In: IEEE. **39th Annual Simulation Symposium (ANSS'06)**. [S.l.], 2006. p. 8–pp. Citado na página 33.

BARKER, M. **1,500 scientists lift the lid on reproducibility**. 2016. [Online; accessed 15-May-2019]. Disponível em: <https://www.nature.com/news/1-500-scientists-lift-the-lid-on-reproducibility-1.19970>. Citado 2 vezes nas páginas 25 e 32.

BLENK, A. et al. Survey on network virtualization hypervisors for software defined networking. **IEEE Communications Surveys & Tutorials**, IEEE, v. 18, n. 1, p. 655–685, 2015. Citado na página 29.

BUYYA, R.; VECCHIOLA, C.; SELVI, S. T. **Mastering cloud computing: foundations and applications programming**. [S.l.]: Newnes, 2013. Citado 4 vezes nas páginas 15, 25, 27 e 28.

CHAO, L. **Virtualization and Private Cloud with VMware Cloud Suite**. [S.l.]: CRC Press, 2017. Citado na página 29.

DELL. **Virtualized HPC Performance with VMware vSphere 6.5 on a Dell PowerEdge C6320 Cluster**. 2017. [Online; accessed 5-June-2018]. Disponível em: <https://www.dell.com/support/article/br/pt/brbsdt1/sln316645/virtualized-hpc-performance-with-vmware-vsphere-6-5-on-a-dell-poweredge-c6320-cluster?lang=en>. Citado 2 vezes nas páginas 15 e 30.

DESAI, A. et al. Hypervisor: A survey on concepts and taxonomy. **International Journal of Innovative Technology and Exploring Engineering**, Citeseer, v. 2, n. 3, p. 222–225, 2013. Citado na página 29.

DONGARRA, J. J. The linpack benchmark: An explanation. In: SPRINGER. **International Conference on Supercomputing**. [S.l.], 1987. p. 456–474. Citado na página 30.

ELSAYED, A.; ABDELBAKI, N. Performance evaluation and comparison of the top market virtualization hypervisors. In: IEEE. **Computer Engineering & Systems (ICCES), 2013 8th International Conference on**. [S.l.], 2013. p. 45–50. Citado 2 vezes nas páginas 35 e 39.

FEO, J. T. An analysis of the computational and parallel complexity of the livermore loops. **Parallel Computing**, Elsevier, v. 7, n. 2, p. 163–185, 1988. Citado na página 31.

FREEBENCH. **Google Code Archive**. 2008. [Online; acessado 2 de setembro de 2018]. Disponível em: <https://code.google.com/archive/p/freebench/>. Citado na página 39.

GFORTRAN. **Welcome to GCC Wiki**. 2019. [Online; accessed 1-may-2019]. Disponível em: <https://gcc.gnu.org/>. Citado na página 52.

GRANISZEWSKI, W.; ARCISZEWSKI, A. Performance analysis of selected hypervisors (virtual machine monitors-vmms). **International Journal of Electronics and Telecommunications**, De Gruyter Open, v. 62, n. 3, p. 231–236, 2016. Citado 3 vezes nas páginas 37, 38 e 43.

GUZEK, M. et al. A holistic model of the performance and the energy efficiency of hypervisors in a high-performance computing environment. **Concurrency and Computation: Practice and Experience**, Wiley Online Library, v. 26, n. 15, p. 2569–2590, 2014. Citado na página 35.

HWANG, J. et al. A component-based performance comparison of four hypervisors. In: IEEE. **Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on**. [S.l.], 2013. p. 269–276. Citado 4 vezes nas páginas 25, 35, 38 e 39.

JUNIOR, N. A. A.; CORDEIRO, W. L. da C.; GASPARY, L. P. Permitindo maior reprodutibilidade de experimentos em ambientes distribuídos com nodos de baixa confiabilidade. In: SBC. **Anais do XXXVI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos**. [S.l.], 2018. Citado 2 vezes nas páginas 32 e 33.

KVM. **Main Page — KVM,**. 2016. [Online; accessed 5-June-2018]. Disponível em: <https://www.linux-kvm.org/index.php?title=Main_Page&oldid=173792>. Citado 3 vezes nas páginas 11, 26 e 51.

MALISZEWSKI, A. M.; GRIEBLER, D.; SCHEPKE, C. Desempenho em instâncias lxc e kvm de nuvem privada usando aplicações científicas. 2018. Citado 4 vezes nas páginas 27, 31, 35 e 47.

MALISZEWSKI, A. M. et al. The nas benchmark kernels for single and multi-tenant cloud instances with lxc/kvm. In: **International Conference on High Performance Computing & Simulation (HPCS)**. [S.l.]: IEEE, 2018. Citado 4 vezes nas páginas 15, 25, 41 e 42.

MANIK, V. K.; ARORA, D. Performance comparison of commercial vmm: Esxi, xen, hyper-v & kvm. In: IEEE. **2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)**. [S.l.], 2016. p. 1771–1775. Citado na página 35.

MANSILHA, R. B.; BARCELLOS, M. P.; BRASILEIRO, F. V. Torrentlab: Um ambiente para avaliaçao do protocolo bittorrent. **XXVI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos (SBRC 2008)**, p. 1–14, 2008. Citado 2 vezes nas páginas 32 e 33.

MORABITO, R.; KJÄLLMAN, J.; KOMU, M. Hypervisors vs. lightweight virtualization: a performance comparison. In: IEEE. **Cloud Engineering (IC2E), 2015 IEEE International Conference on**. [S.l.], 2015. p. 386–393. Citado na página 43.

MPICH. **MPICH | High-Performance Portable MPI**. 2019. [Online; accessed 1-may-2019]. Disponível em: <https://www.mpich.org/>. Citado na página 52.

NUSSBAUM, L. Testbeds support for reproducible research. In: **Proceedings of the Reproducibility Workshop**. New York, NY, USA: ACM, 2017. (Reproducibility '17), p. 24–26. ISBN 978-1-4503-5060-0. Disponível em: <http://doi.acm.org/10.1145/3097766.3097773>. Citado na página 32.

NUSSBAUM, L. et al. Linux-based virtualization for hpc clusters. In: **Montreal Linux Symposium**. [S.l.: s.n.], 2009. Citado na página 29.

ORACLE. **Welcome to VirtualBox.org!** 2019. [Online; accessed 19-april-2019]. Disponível em: <https://www.virtualbox.org/>. Citado na página 29.

PITANGA, M. **Construindo supercomputadores com Linux**. [S.l.]: Brasport, 2004. Citado 2 vezes nas páginas 27 e 29.

POTHERI, M. **Virtualized High Performance Computing (HPC) Reference Architecture (Part 1 of 2)**. 2018. [Online; accessed 1-may-2019]. Disponível em: <https://blogs.vmware.com/apps/2018/09/vhpc-ra-part1.html>. Citado na página 29.

RADISKHLEBOVA, A. A. et al. Study of the possibilities of using virtualization tools to optimize the cluster resources management. In: IEEE. **2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)**. [S.l.], 2019. p. 310–314. Citado 2 vezes nas páginas 25 e 40.

REDDY, P. V. V.; RAJAMANI, L. Performance evaluation of hypervisors in the private cloud based on system information using sigar framework and for system workloads using passmark. **International Journal of Advanced Science and Technology**, v. 70, p. 17–32, 2014. Citado na página 35.

ROSSO, J. P. Análise de desempenho de aplicações científicas em ambiente de nuvem privada. Universidade Federal do Pampa, 2015. Citado na página 47.

RUFUS. **Create bootable USB drives the easy way**. 2019. [Online; accessed 1-may-2019]. Disponível em: <https://rufus.ie/>. Citado na página 51.

SERRAPILHEIRA. **Projeto vai estimar a reprodutibilidade da ciência brasileira**. 2019. [Online; accessed 20-may-2018]. Disponível em: <https://serrapilheira.org/projeto-vai-estimar-a-reprodutibilidade-da-ciencia-brasileira/>. Citado 2 vezes nas páginas 15 e 32.

SILVA, W. da; LEIRIA, R.; SCHEPKE, C. Avaliação dos hipervisores vmware esxi e gnu/linux kvm para computação de alto desempenho. ERAD/RS 2019 - Fórum de Iniciação Científica, 2019. Citado na página 35.

STERLING, T.; ANDERSON, M.; BRODOWICZ, M. **High Performance Computing: Modern Systems and Practices**. [S.l.]: Morgan Kaufmann, 2017. Citado 3 vezes nas páginas 25, 27 e 30.

SUBHLOK, J.; VENKATARAMAIAH, S.; SINGH, A. Characterizing nas benchmark performance on shared heterogeneous networks. In: IEEE. **Proceedings 16th International Parallel and Distributed Processing Symposium**. [S.l.], 2001. p. 9–pp. Citado 4 vezes nas páginas 64, 66, 67 e 68.

TANENBAUM, A.; BOS, H. **Modern Operating Systems**. Pearson, 2015. ISBN 9780133591620. Disponível em: <https://books.google.com.br/books?id=9gqnngEACAAJ>. Citado 3 vezes nas páginas 15, 29 e 30.

URBAN, P.; DÉFAGO, X.; SCHIPER, A. Neko: A single environment to simulate and prototype distributed algorithms. In: IEEE. **Proceedings 15th International Conference on Information Networking**. [S.l.], 2001. p. 503–511. Citado na página 33.

VARRETTE, S. et al. Hpc performance and energy-efficiency of xen, kvm and vmware hypervisors. In: IEEE. **2013 25th International Symposium on Computer Architecture and High Performance Computing**. [S.l.], 2013. p. 89–96. Citado na página 35.

VMWARE. **VMware vSphere Hypervisor gratuito, Virtualização gratuita (ESXi)**. 2018. [Online; accessed 5-June-2018]. Disponível em: <https://www.vmware.com/br/products/vsphere-hypervisor.html>. Citado 3 vezes nas páginas 11, 26 e 51.

VOGEL, A. et al. Medindo o desempenho de implantaç oes de openstack, cloudstack e opennebula em aplicações cientıficas. **16th Escola Regional de Alto Desempenho do Estado do Rio Grande do Sul (ERAD/RS)**, p. 279–282, 2016. Citado 5 vezes nas páginas 15, 27, 36, 37 e 47.