

**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 (HPC)**

Alegrete  
2018



**Willian Soares da Silva**

**Avaliação Automatizada de Desempenho de Sistemas  
de Virtualização GNU/Linux e VMware para  
Computação de Alto Desempenho (HPC)**

Trabalho de Conclusão de Curso apresentado  
ao Curso de Graduação em Ciência da Com-  
putação da Universidade Federal do Pampa  
como requisito parcial para a obtenção do tí-  
tulo de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Claudio Schepke

Coorientador: Ms. Raul Dias Leiria

Alegrete  
2018



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 essa longa jornada. A minha família, e 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 os meus orientadores Claudio Schepke e Raul Leiria, que sempre me ajudaram quando eu precisei. Muito obrigado à 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

Virtualizadores e contêineres são tecnologias fundamentais para a virtualização. Eles são os responsáveis pela abstração do hardware e fornecimento de recursos virtualizados para os *guests*. Dependendo do tipo de aplicação de que será executada nos ambientes virtualizados a performance pode variar significativamente de acordo com a tecnologia escolhida. Na verdade o desempenho da tecnologia de virtualização depende do tipo de aplicação que será executado. 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 entender qual deles teve o melhor desempenho nos experimentos a serem realizados. Para isso, vamos propor o desenvolvimento de um *script* que automatize as seguintes funções: execução dos *benchmarks* para avaliar os virtualizadores, coleta de dados e geração de gráficos.

**Palavras-chave:** Virtualizadores. Contêineres. Linux KVM. VMware ESXi. Benchmark. Automatização.



## ABSTRACT

Hypervisors and containers are key technologies for virtualization. They are responsible for abstracting the hardware and providing virtualized resources for the guests. ~~Depending on the type of application that will run in virtualized environments, performance may vary significantly depending on the technology chosen. In fact the performance of virtualization technology depends on the type of application that will run.~~ The present work aims to evaluate the performance of two different virtualizers: KVM (KVM, 2016) and VMware ESXi (VMWARE, 2018), to verify discrepancies and to understand which one has the best performance in the experiments to be performed. Thus, we will propose the development of a script that automates the following functions: running benchmarks to evaluate virtualizers, data collection and generation of graphs.

**Key-words:** Hypervisors. Container. Linux KVM. VMware ESX. Benchmark. Automation.



## LISTA DE FIGURAS

Figura 1 – Representação de um sistema computacional MIMD (BUYYA; VECCHIOLA; SELVI, 2013).	26
Figura 2 – Sistema computacional com memória compartilhada (esquerda) e memória distribuída (direita) (BUYYA; VECCHIOLA; SELVI, 2013).	27
Figura 3 – Benefícios da virtualização em <i>High Performance Computing</i> (HPC) (DELL, 2017).	29
Figura 4 – <i>Hypervisor</i> tipo 1 (esquerda) e tipo 2 (direita) (TANENBAUM; BOS, 2015).	30
Figura 5 – Virtualização leve (XAVIER et al., 2013).	31
Figura 6 – Arquitetura do Docker (PREETH et al., 2015).	32
Figura 7 – Representação em alto nível da metodologia utilizada em (MALISZEWSKI et al., 2018).	39
Figura 8 – Configurações dos experimentos.	47
Figura 9 – Visão arquitetural do trabalho proposto.	51
Figura 10 – Exemplo de saída do gráfico (MALISZEWSKI et al., 2018).	53



## LISTA DE TABELAS

Tabela 1 – Desempenho computacional em <i>Floating Point Operations Per Second</i> (FLOPs).	23
Tabela 2 – A tabela apresenta o primeiro colocado de cada ano na lista dos supercomputadores com maior capacidade de processamento do mundo (TOP500, 2018).	24
Tabela 3 – A tabela mostra o primeiro colocado de cada ano na lista dos supercomputadores mais potentes do mundo em eficiência energética (TOP500, 2018).	25
Tabela 4 – Tabela comparativa dos trabalhos relacionados.	42
Tabela 5 – Informação do servidor onde os virtualizadores serão instalados para avaliação.	48
Tabela 6 – Cronograma de atividades.	55



## LISTA DE SIGLAS

**CPU** *Central Processing Unit*

**PE** *Processing Element*

**FLOP** *Floating Point Operation Per Second*

**GB** *Gigabyte*

**HD** *Hard Disk*

**HPC** *High Performance Computing*

**KVM** *Kernel-based Virtual Machine*

**LXC** *Linux Containers*

**MIMD** *Multiple Instruction Multiple Data*

**MPI** *Message Passing Interface*

**NPB** *The NAS Parallel Benchmarks*

**OMP** *Open Multi-Processing*

**PC** *Personal Computer*

**RAM** *Random Access Memory*

**SO** Sistema Operacional

**vCPU** *virtual CPU*

**VM** Virtual Machine

**VMM** *Virtual Machine Monitor*



## SUMÁRIO

1	INTRODUÇÃO . . . . .	21
1.1	Motivação e objetivos . . . . .	22
1.2	Organização do documento . . . . .	22
2	CONTEXTUALIZAÇÃO . . . . .	23
2.1	Computação de Alto Desempenho (HPC) . . . . .	23
2.2	Paralelismo . . . . .	25
2.3	Benchmark . . . . .	27
2.4	Virtualização para HPC . . . . .	28
2.5	Hypervisors . . . . .	29
2.6	Contêineres . . . . .	30
2.7	Conclusão do capítulo . . . . .	31
3	TRABALHOS RELACIONADOS . . . . .	33
3.1	Avaliação de <i>Hypervisors</i> . . . . .	33
3.1.1	VOGEL, A. et al. 2016 - Medindo o Desempenho de Implementações de <i>Openstack</i> , <i>Cloudstack</i> e <i>Opennebula</i> em Aplicações Científicas . . . . .	33
3.1.2	GRANISZEWSKI et al. 2016 - <i>Performance Analysis of Selected Hypervisors (Virtual Machine Monitors (VMMs))</i> . .	34
3.1.3	HWANG, J. et al. 2013 - <i>A Component-Based Performance Comparison of Four Hypervisors</i> . . . . .	35
3.1.4	ELSAYED et al. 2013 - <i>Performance Evaluation and Comparison of the Top Market Virtualization Hypervisor</i> . . . . .	36
3.2	Avaliação <i>Hypervisors</i> x Contêineres . . . . .	38
3.2.1	MALISZEWSKI, A. M. et al. 2018 - <i>The Nas Benchmark Kernels for Single and Multi-Tenant Cloud Instances with LXC/KVM</i> . . . . .	38
3.2.2	MALISZEWSKI et al. 2018 - Desempenho em Instâncias LXC e KVM de Nuvem Privada Usando Aplicações Científicas. . . . .	40
3.2.3	MORABITO et al. 2015 - <i>Hypervisors vs. Lightweight Virtualization: A Performance Comparison</i> . . . . .	41
3.3	Conclusão do capítulo . . . . .	42
4	METODOLOGIA . . . . .	45
4.1	Avaliação de desempenho . . . . .	45
4.2	Ambiente dos experimentos . . . . .	48
4.3	Métricas de desempenho . . . . .	48

4.4	Conclusão do Capítulo . . . . .	49
5	AVALIAÇÃO DE DESEMPENHO . . . . .	51
5.1	Proposta do trabalho . . . . .	51
5.2	Conclusão do capítulo . . . . .	54
6	CONSIDERAÇÕES FINAIS . . . . .	55
6.1	Cronograma de atividades . . . . .	55
	REFERÊNCIAS . . . . .	57
	Índice . . . . .	61

## 1 INTRODUÇÃO

O uso da virtualização para Computação de Alto Desempenho - *High Performance Computing (HPC)* - vem recebendo uma crescente atenção da literatura. Seja o seu uso em nuvens computacionais ou em supercomputadores, a virtualização realiza o gerenciamento de Virtual Machine (VMs), acelera a implantação dessas máquinas e melhora a eficiência do uso dos recursos disponíveis. Além disso, essa tecnologia também permite que usuários executem diferentes tipos de sistemas operacionais **e em diferentes VM** em um mesmo servidor físico. Apesar de ser um tópico bastante discutido atualmente, essa ideia já havia chamado a atenção de pesquisadores a alguns anos atrás. Na década de 1960 a empresa *IBM (International Business Machines)* já realizava experiências com não apenas uma, mas duas tecnologias de virtualização desenvolvidas de forma independente: **SIMMON** e **CP-40** (TANENBAUM; BOS, 2015)

Uma das principais características **dessa** tecnologia é fazer com que *workloads* de uma máquina física possam ser divididas em partes para serem executadas em vários outros **usuários VMs**. Dessa forma, é possível fazer o uso eficiente do computador físico ao atribuir tarefas **à um** conjunto de nós. Além disso economiza-se energia, pois ao utilizar um ambiente virtualizado em uma mesma máquina física, evitando o uso de servidores físicos que consomem uma quantidade significativa de energia e contribuem com a emissão de CO<sub>2</sub> no meio ambiente, **estudos** já estão sendo realizados para monitorar o consumo energético de servidores (LEIRIA, 2016).

**Apesar** das vantagens obtidas com o uso **dessa** tecnologia, **existe um custo**. 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 VM, por exemplo, a maneira que o hypervisor escalona os recursos como, CPU, memória, disco, e rede (HWANG et al., 2013). A combinação de novas arquiteturas de *Central Processing Unit* (CPU) com suporte **a** virtualização e os avanços no desenvolvimento de *hypervisors* **tem** atenuado esse problema. Porém, os *hypervisors* ainda assim apresentam diferentes **tipos de desempenho** por causa dessa sobrecarga.

Existem diferentes *hypervisors*, **por exemplo**, Oracle VM, Xen, Hyper-V, KVM, VMWare, etc. Cada um deles ~~possui suas próprias características~~, devida ~~essa circunstância~~ ~~cada tipo de tecnologia~~ realiza o gerenciamento dos recursos dos computadores de maneira diferente. Essa variedade de *hypervisors* resulta em ~~novos~~ desafios para entender qual a melhor escolha a ser feita dependendo do tipo de aplicação que se deseja executar no ambiente virtualizado.

## 1.1 Motivação e objetivos

Neste trabalho vamos propor uma análise comparativa entre dois *hypervisors*: o KVM (KVM, 2016) e VMware ESXi (VMWARE, 2018), os testes serão realizados em diferentes cenários que serão definidos, focando no paralelismo das aplicações. Vamos propor o desenvolvimento de um script terá as seguintes funções: automatização do processo das execuções dos *benchmarks* da suíte *The NAS Parallel Benchmarks (NPB)-Message Passing Interface (MPI)*; coletas dos resultados das execuções dos *benchmarks* e geração dos gráficos a partir dos resultados obtidos. Por fim vamos comparar os resultados e verificar qual dos virtualizadores teve melhor desempenho de acordo com os experimentos.

## 1.2 Organização do documento

O trabalho está organizado da seguinte maneira. No Capítulo 2 são apresentados conceitos básicos sobre virtualizadores, computação de alto desempenho e *benchmarks*. No Capítulo 3 estão os artigos utilizados como referência para o desenvolvimento deste trabalho. No Capítulo 4 apresentaremos a metodologia que será utilizada na avaliação dos experimentos, bem como a proposta de automatização dos experimentos e *parsing* dos resultados. No Capítulo 5 apresentamos um esboço teórico do trabalho proposto que será utilizado como base para o desenvolvimento do script. E por fim, no Capítulo 6 as considerações finais e as direções de trabalhos futuros.

## 2 CONTEXTUALIZAÇÃO

### 2.1 Computação de Alto Desempenho (HPC)

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

Uma métrica utilizada para avaliar o desempenho dos supercomputadores em HPC, é operações de ponto flutuante por segundo ou **FLOPs**. O FLOP é a adição ou multiplicação de dois números reais (ou ponto flutuantes) representado na forma que o supercomputador consegue processar e manipular. Os primeiros supercomputadores tinham tecnologia para alcançar apenas cerca de 1 KiloFLOPs. Atualmente, um **Personal Computer** (PC) desempenha alguns GigaFLOPs. Já um supercomputador, tem a capacidade de processamento que é muito maior que os PCs de acordo com essa métrica, alcançando a casa dos TeraFLOPs. A Tabela 1 mostra em ordem crescente a capacidade de processamento de **supercomputadores** em FLOPs.

Ordem de grandeza	Abreviatura	Exponencial	Decimal
FLOPs	FLOPs	$10^0$	1
KiloFLOPs	KFLOPs	$10^3$	1.000
MegaFLOPs	MFLOPs	$10^6$	1.000.000
GigaFLOPs	GFLOPs	$10^9$	1.000.000.000
TeraFLOPs	TFLOPs	$10^{12}$	1.000.000.000.000

Tabela 1 – Desempenho computacional em FLOPs.

Esse métrica de avaliação (FLOPs) é o mesmo utilizado para medir a capacidade de processamento dos supercomputadores com melhor capacidade de processamento do mundo **na lista do Top500**. A Tabela 2 mostra as máquinas com maior poder computacional de cada ano, utilizando os dados do mês de novembro, a partir do ano 2000 até o ano de 2018. A Tabela 2 está organizada da seguinte maneira: a coluna **Ano** indica qual o supercomputador maior capacidade de processamento durante esse período, sempre levando em consideração a atualização do mês de novembro (exceto o ano de 2018 - o supercomputador é do mês de junho). A segunda coluna é o **Nome** dado ao supercomputador pelos seus criadores. A terceira coluna **Cores** mostra o número de núcleos físicos que cada máquina possui. A quarta coluna **Rmax (TFLOPs/s)** mostra o desempenho máximo alcançado pelo supercomputador quando é executada uma aplicação sintética para realizar a sua avaliação. A última coluna **Rpeak (TFlop/s)** é o máximo de desempenho que o supercomputador teoricamente teria que alcançar quando executada a aplicação sintética para avaliação.

Podemos observar que algumas dessas máquinas permanecem em primeiro lugar na lista do Top500 por mais de um ano e quando há um novo vencedor, a sua capacidade

Ano	Nome do supercomputador	Cores	Rmax (TFLOPs/s)	Rpeak (TFLOPs/s)
2018	Summit - IBM Power System AC922	2.282.544	122.300,0	187.659,3
2017	Sunway TaihuLight - Sunway MPP	10.649.600	93.014,6	125.435,9
2016	Sunway TaihuLight - Sunway MPP	10.649.600	93.014,6	125.435,9
2015	Tianhe-2A - TH-IVB-FEP Cluster	3.120.000	33.862,7	54.902,4
2014	Tianhe-2A - TH-IVB-FEP Cluster	3.120.000	33.862,7	54.902,4
2013	Tianhe-2A - TH-IVB-FEP Cluster	3.120.000	33.862,7	54.902,4
2012	Titan - Cray XK7, Opteron 6274 16C 2.200GHz	560.640	17.590,0	27.112,5
2011	K computer, SPARC64 VIIIfx 2.0GHz	705.024	10.510	11.280,4
2010	Tianhe-1A - NUDT TH MPP, X5670 2.93Ghz 6C	186.368	2.566	4.701,0
2009	Jaguar - Cray XT5-HE Opteron 6-core 2.6 GHz	224.162	1.759	2.331,0
2008	Roadrunner - BladeCenter QS22/LS21 Cluster	129.600	1.105	1.456,7
2007	BlueGene/L - eServer Blue Gene Solution	212.992	478,2	596,4
2006	BlueGene/L - eServer Blue Gene Solution	131.072	280,6	367,0
2005	BlueGene/L - eServer Blue Gene Solution	131.072	280,6	367,0
2004	BlueGene/L beta-System - BlueGene/L	32.768	70,72	91,75
2003	Earth-Simulator, NEC.	5.120	35,86	40,96
2002	Earth-Simulator, NEC.	5.120	35,86	40,96
2001	ASCI White, SP Power3 375 MHz	8.192	7.226	12.288
2000	ASCI White, SP Power3 375 MHz	8.192	4.938	12.288

Tabela 2 – A tabela apresenta o primeiro colocado de cada ano na lista dos supercomputadores com maior capacidade de processamento do mundo (TOP500, 2018).

de processamento (**Rmax FLOPs/s**) é por vezes maior que o dobro em comparação ao primeiro lugar do ano anterior. O aumento no número de cores dessas máquinas elevaram sua capacidade de processamento, porém é interessante notar que de acordo com a Tabela 2, nem sempre a capacidade de processamento está ligada à quantidade de núcleos que o supercomputador possui, por exemplo, o supercomputador que está em primeiro lugar na lista deste ano (2018) possui uma quantidade inferior de *cores* se comparado ao computador que estava na liderança no fim do ano passado (2017). Isso ocorre por causa do uso de GPUs nos supercomputadores, elas ajudam a aumentar a capacidade de processamento destas máquinas e assim não é necessário um grande numero de *cores* para melhorar o desempenho.

Além de atualizar o rank dos 500 supercomputadores como maior capacidade de processamento do mundo nos meses de junho e novembro anualmente, a crescente preocupação com o meio ambiente e o consumo energético desses servidores chamou atenção da equipe do top500.org, que em 15 de novembro de 2007 criou a primeira lista Green500 (TOP500, 2018).

A lista Green500 classifica os 500 supercomputadores do mundo em eficiência energética. As operações computacionais de desempenho a qualquer custo levou ao surgimento de supercomputadores que consomem grandes quantidades de energia elétrica, e produzem tanto calor a ponto de grandes instalações de resfriamento precisarem ser construídas para garantir a temperatura adequada. Para lidar com essa tendência, a lista Green500 valoriza o desempenho energeticamente eficiente para a supercomputação sustentável. A Tabela 3 mostra as máquinas com maior eficiência energética de cada ano, utilizando sempre os dados do mês de novembro, a partir do ano 2013 até o ano de 2018 (exceto o

Ano	Nome do Supercomputador	GFLOPs/W	Total Power(kW)
2018	Shoubu system B - ZettaScaler-2.2	18.404	47
2017	Shoubu system B - ZettaScaler-2.2	17.009	50
2016	NVIDIA DGX-1	9,4621	349,5
2015	ExaScaler-1.4 80Brick	7,0314	50,3
2014	ASUS ESC4000 FDR/G2S	5,2721	57,2
2013	LX 1U-4GPU/104Re-1G Cluster	4,5032	27,8

Tabela 3 – A tabela mostra o primeiro colocado de cada ano na lista dos supercomputadores mais potentes do mundo em eficiência energética (TOP500, 2018).

ano de 2018 - o supercomputador é do mês de junho).

A Tabela 3 é um pouco diferente da Tabela 2, visto que ela leva em consideração a capacidade de processamento por *Watts*. Na coluna **Ano** temos os supercomputadores que ficaram em primeiro lugar em cada ano, seguido de seu **Nome** na segunda coluna. A terceira coluna mostra a quantidade de processamento que o supercomputador realiza por *Watt* (**GFLOPs/W**), quanto maior for a quantidade de FLOPs computados por *Watt* melhor é a classificação do supercomputador. Por fim, a última coluna mostra a quantidade de energia elétrica consumida pelo super computador.

É interessante notar que, durante todos esses anos até a presente data, no qual foram feitas essas duas listas pelo site top500.org classificando os supercomputadores, o supercomputador que está em primeiro lugar na lista Top500 ainda não foi o mesmo que ocupou a primeira posição na lista Green500 no mesmo ano.

Outra métrica também levada em consideração é o tempo que o computador leva para resolver um determinado problema (STERLING; ANDERSON; BRODOWICZ, 2017). Há aplicações computacionais que são utilizadas especificamente para esse tipo de avaliação, como os *benchmarks*.

## 2.2 Paralelismo

O paralelismo é a capacidade que um sistema computacional tem de processar múltiplas tarefas de maneira simultânea (BUYYA; VECCHIOLA; SELVI, 2013). Uma aplicação paralela é composta por várias tarefas (*tasks*) que são executadas simultaneamente para resolver um determinado problema. Uma tarefa (*task*) é dividida em múltiplas subtarefas (*subtasks*) através de técnicas como divisão e conquista por exemplo, e cada subtarefa é processada em uma CPU diferente.

Há aplicações que necessitam de um maior poder computacional. Com o paralelismo, é possível proporcionar soluções viáveis para esse problema, através do aumento do número de CPUs em um sistema computacional 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 sistema

computacional e alcançar melhor performance (BUYYA; VECCHIOLA; SELVI, 2013).

Um sistema computacional capaz de executar múltiplas instruções em múltiplos processadores é chamado de *Multiple Instruction Multiple Data* (MIMD). Cada elemento de processamento MIMD possui instruções e fluxos de dados separados. Computadores MIMD trabalham de maneira assíncrona (BUYYA; VECCHIOLA; SELVI, 2013). A Figura 1 representa um sistema computacional MIMD.

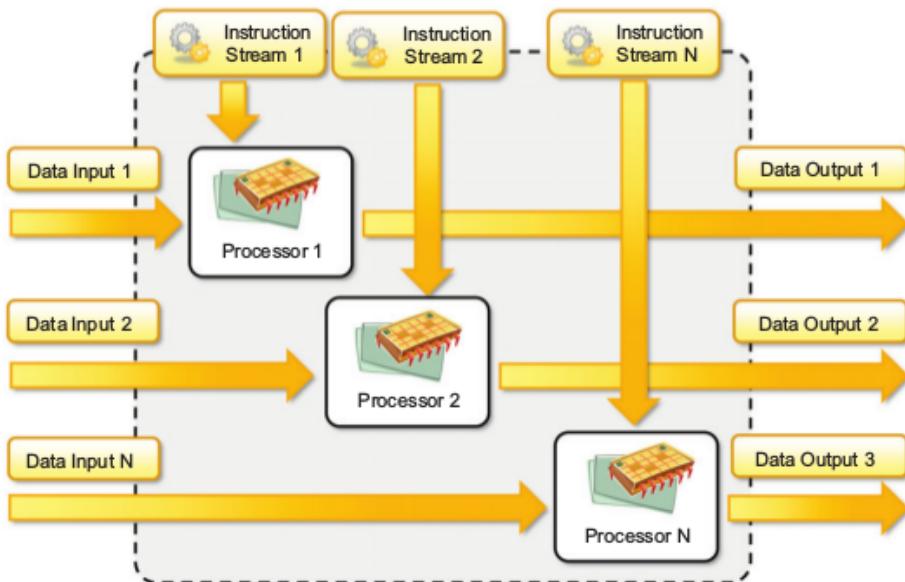


Figura 1 – Representação de um sistema computacional MIMD (BUYYA; VECCHIOLA; SELVI, 2013).

As MIMD são classificadas como memória compartilhada e memória distribuída, pela maneira que os *Processing Elements* (PEs) são alocados na memória principal.

No modelo *memória compartilhada* todos PEs estão conectados em uma única memória principal e todos eles têm acesso à ela, como mostra a Figura 2. A comunicação entre os PEs é feita através da memória que é compartilhada entre eles, as modificações dos dados feitas na memória por um dos PEs é visível para todos os outros PEs.

Na memória distribuída, todos os PEs possuem memória local. A comunicação entre os PEs nesse modelo é feita através da conexão de rede. Cada PEs opera de forma assíncrona, e se a comunicação/sincronização entre as tarefas é necessária, elas podem fazer isso através de trocas de mensagens entre elas (BUYYA; VECCHIOLA; SELVI, 2013). A Figura 2 ilustra os dois tipos de memórias mencionadas.

Para executar aplicações em computadores de forma paralela é possível utilizar interfaces de programação paralela, como por exemplo o *Open Multi-Processing* (OMP), para memória compartilhada e o MPI para memória distribuída.

O OpenMP (OPENMP, 2018) é uma interface de programação paralela que permite explorar o paralelismo através do uso da memória compartilhada.

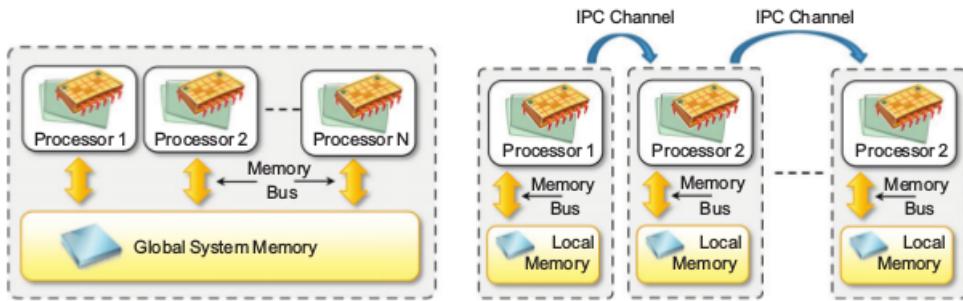


Figura 2 – Sistema computacional com memória compartilhada (esquerda) e memória distribuída (direita) (BUYYA; VECCHIOLA; SELVI, 2013).

O MPI (MPI, 2018) é uma interface de programação paralela onde é possível parallelizar aplicações através do uso da memória distribuída.

### 2.3 Benchmark

Para avaliar o desempenho de componentes de um sistema computacional como: *Hard Disk (HD)*, *Random Access Memory (RAM)*, rede, entrada/saída e virtualizadores, podemos utilizar os *benchmarks*. Um exemplo de *benchmark* é o NPB. Historicamente o NPB não foi o primeiro *benchmark* desenvolvido (BAILEY et al., 1991), antes dele existiam outros como o *LINPACK Benchmark*, que é utilizado para avaliar o desempenho dos computadores nas Tabelas 2 e 3, e o outro *benchmark* criado antes do NPB foi o *Livermore loops*.

O *benchmark Linpack HPL (High Parallel Linpack)*, resolve uma série de equações lineares na forma matricial (STERLING; ANDERSON; BRODOWICZ, 2017). Um *benchmark* é utilizado como meio de comparação para avaliar dois sistemas independentes medindo o tempo necessário para realizar a mesma tarefa (STERLING; ANDERSON; BRODOWICZ, 2017). Enquanto no *Livermore Loops*, os *loops* representam o tipo de núcleos de computação tipicamente encontrados em computação científica de larga escala (FEO, 1988). Eles variam de operações matemáticas comuns, como produto interno e multiplicação de matrizes, e algoritmos de busca e armazenamento, como método de Monte Carlo.

O NPB é um conjunto de *benchmarks* que tem como objetivo realizar a avaliação de desempenho de sistemas computacionais. Eles são desenvolvidos e mantidos pela NASA Advanced Supercomputing (NAS) (NAS, 2018) (antigo Programa de Simulação Aerodinâmica Numérica da NASA) com base no Centro de Pesquisas Ames da NASA. Antes da criação do NPB alguns *benchmarks* que avaliavam o desempenho dos computadores não eram adequados para avaliar máquinas paralelas de alto desempenho (BAILEY et al., 1991). Havia problemas como restrições de ajuste impeditivas de paralelismo, tamanhos insuficientes de problemas que os tornava inadequados para sistemas altamente

paralelos, etc. Por esse motivo o NPB foi desenvolvido, para resolver a falta de referências aplicáveis à máquinas altamente paralelas. Os *benchmarks* da suíte NPB são derivados das aplicações dinâmicas dos fluídos computacionais (CFD) e é composto por 5 *kernels* e 3 pseudo-aplicações (NAS, 2018), ~~são elas:~~

- 5 *Kernels*
  - IS (Ordenação de Inteiros) - Ordenação de inteiros, acesso randômico à memória;
  - EP - *Embarassingly Parallel*;
  - CG - Gradiente Conjugado, acesso a memória irregular e comunicação;
  - MG - *Multi-Grid* em uma sequência de malhas, comunicação de longa e curta distância, memória intensiva;
  - FT - Transformada rápida de Fourier 3D discreta, comunicação todos-para-todos.
- 3 Pseudo-aplicações
  - BT - Solucionador tri-diagonal de blocos;
  - SP - Solucionador pentádico escalar;
  - LU - Solucionador Gauss-Seidel.

É possível escolher dentre 8 diferentes tipos de classes para executar o NPB (NAS, 2018), ~~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.~~

## 2.4 Virtualização para HPC

Por mais que os supercomputadores tenham alta capacidade de processamento atualmente, apenas algumas aplicações de HPC conseguem explorar a verdadeira capacidade computacional (TRINITIS; WEIDENDORFER, 2017). Por isso, considera-se a

virtualização uma opção para obter o máximo de desempenho dos supercomputadores (TRINITIS; WEIDENDORFER, 2017).

A virtualização é a capacidade de criar computadores virtuais dentro de servidores físicos (máquinas *host*). Essas máquinas provêem a plataforma de *hardware* para a VMs. Múltiplas máquinas virtuais podem ser alocadas em uma máquina *host*. Essa tecnologia possui benefícios como tolerância à falhas e segurança (DELL, 2017). A Figura 3 representa algumas das características da virtualização.

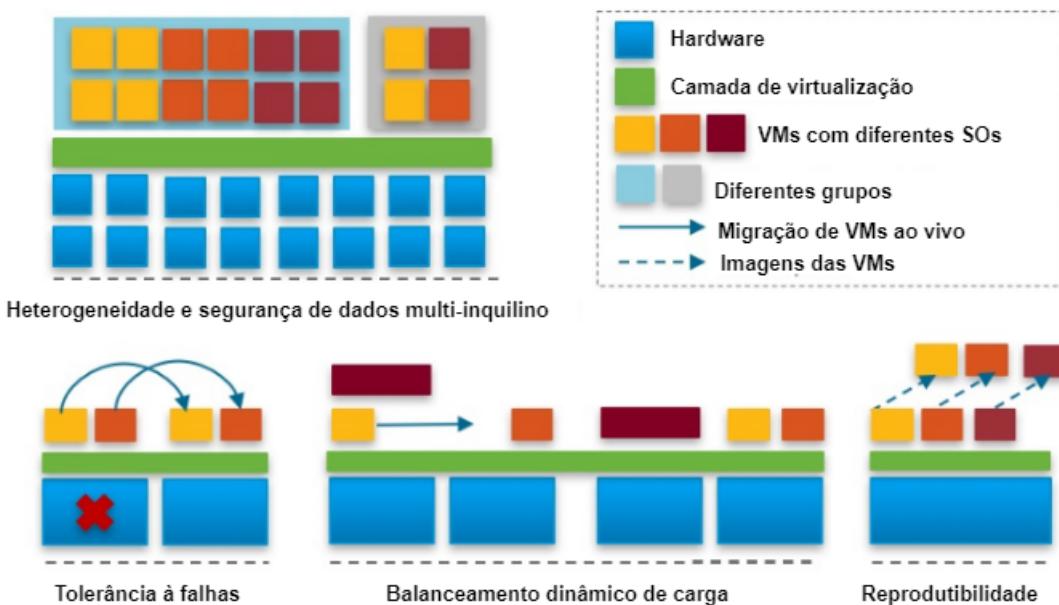


Figura 3 – Benefícios da virtualização em HPC (DELL, 2017).

A ideia principal é que um *Virtual Machine Monitor* (VMM) crie múltiplas VMs, cada uma executando potencialmente um Sistema Operacional (SO) diferente, no mesmo hardware físico. Um VMM também é conhecido como *hypervisor* (TANENBAUM; BOS, 2015).

## 2.5 Hypervisors

O *hypervisor* é uma camada de abstração que permite executar múltiplas VM compartilhando o mesmo *hardware* físico. Existem duas abordagens para virtualização. *hypervisors* do tipo 1 e *hypervisors* tipo 2.

**Hypervisor tipo 1:** é um SO, já que é o único *software* executando no modo privilegiado. Ele é responsável pelo suporte à múltiplas cópias do *hardware* real, chamadas máquinas virtuais, similares aos processos que um SO normal executa. (TANENBAUM; BOS, 2015). O ESXI da VMware, é um tipo de *hypervisor* tipo 1, ele consegue 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, KVM, que fornece a infraestrutura de virtualização de núcleo. Usando o KVM, pode-se executar várias máquinas virtuais que executam SOs como o Linux ou Windows. Cada máquina virtual tem *hardware* virtualizado privado: uma placa de rede, disco, etc. KVM é um virtualizador *opensource* (DELL, 2017).

**Hypervisor tipo 2:** é um *software* que depende de outro SO para alocar e escalonar os recursos computacionais. O *hypervisor* tipo 2 é semelhante a um computador completo com uma CPU e vários dispositivos (TANENBAUM; BOS, 2015). ~~A Figura 4 ilustra os dois tipos de *hypervisors* mencionados.~~

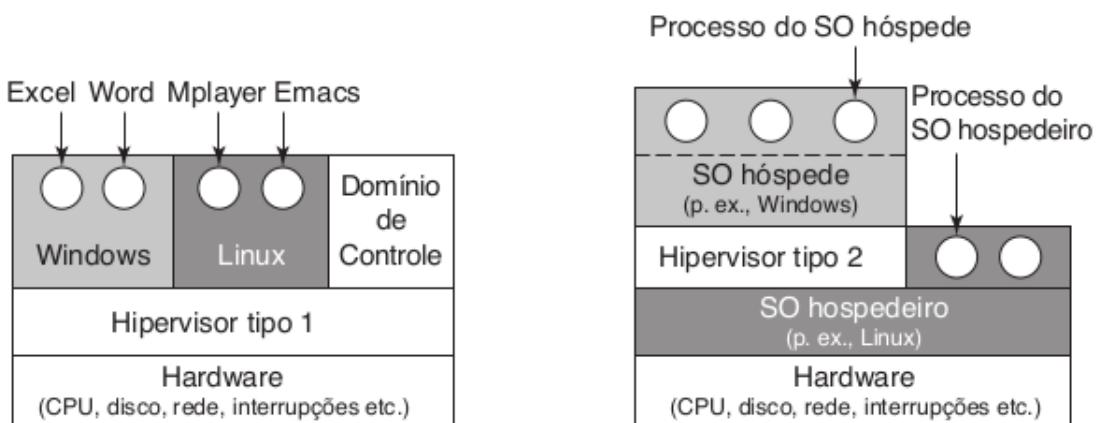


Figura 4 – *Hypervisor* tipo 1 (esquerda) e tipo 2 (direita) (TANENBAUM; BOS, 2015).

## 2.6 Contêineres

Assim como os *hypervisors*, também é possível utilizar *contêineres* na virtualização. Os *hypervisors* fornecem a abstração do *hardware*, no qual resulta em uma sobrecarga em termos da própria virtualização do *hardware* e dos *drivers* virtuais do sistema computacional. Isso significa que, cada VM é totalmente implementada com *hardware* virtual para suportar um SO *guest*, executando sobre o *hypervisor*. Contêineres por outro lado fornecem um isolamento de processo a nível do SO, ou seja, para cada parte virtualizada nova não é criada uma pilha completa de SO dedicado. Contêineres executam sob o mesmo *kernel* do SO hospedeiro em um servidor físico, sendo que um ou mais processos podem executar dentro de cada contêiner (TRINDADE; COSTA, 2018). A Figura 5 ilustra um sistema com dois diferentes tipos de contêineres no mesmo servidor físico.

O **Linux Containers (LXC)** fornece suporte **a** nível do *kernel* para o *Control Groups (cgroups)*. O suporte dos *cgroups* provê ferramentas que controlam os recursos disponíveis para um grupo de processos. Isso informa o *kernel* quais os recursos estão

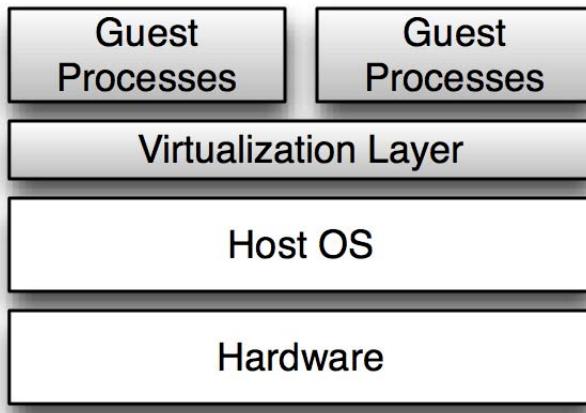


Figura 5 – Virtualização leve (XAVIER et al., 2013).

disponíveis para um determinado processo executando em um contêiner. Um contêiner pode ter acesso limitado à rede e memória, por exemplo. Esse controle faz com que não seja possível causar algum erro em outro contêiner, ou potencialmente danificar o sistema (FLYNT; LAKSHMAN; TUSHAR, 2017).

O Docker utiliza a arquitetura cliente-servidor, os contêineres se comunicam com o *daemon* do Docker na máquina host. O *daemon* Docker é responsável pela construção, execução e distribuição dos contêineres do Docker. Tanto o *daemon* Docker e o contêiner Docker podem ser alocados em um único computador e também podem ser alocados em uma máquina remota. Daemon do Docker e os contêineres se comunicam através de uma bridge, docker0. A Figura 6 mostra a arquitetura do Docker.

O Docker contém três componentes: *Docker images* usada para criar os contêineres; *Docker registries* armazena as imagens do Docker; *Docker containers* é semelhante a um diretório. Ele mantém todos o que é necessário para um aplicativo para executar. Os contêineres são criados *Docker images*. Um contêiner pode ser executado, iniciado, parado, movido e excluído (PREETH et al., 2015).

A virtualização baseada em contêineres é considerada uma alternativa à virtualização baseada em *hypervisors*, uma vez que contêineres implementam isolamento de processos a nível do SO da máquina hospedeira. Assim, evita-se a sobrecarga devido ao hardware virtualizado e aos *drivers* virtualizados. Um contêiner pode ser considerado um ambiente virtual pequeno e isolado, o qual inclui um conjunto de dependências específicas necessárias para executar determinada aplicação (TRINDADE; COSTA, 2018).

## 2.7 Conclusão do capítulo

O objetivo deste capítulo foi apresentar uma visão geral sobre o *benchmarking* de computadores, fornecendo conceitos básicos de virtualização e virtualizadores. Foram apresentadas as características dos tipos de *hypervisors* e também a tecnologia *lightweight*

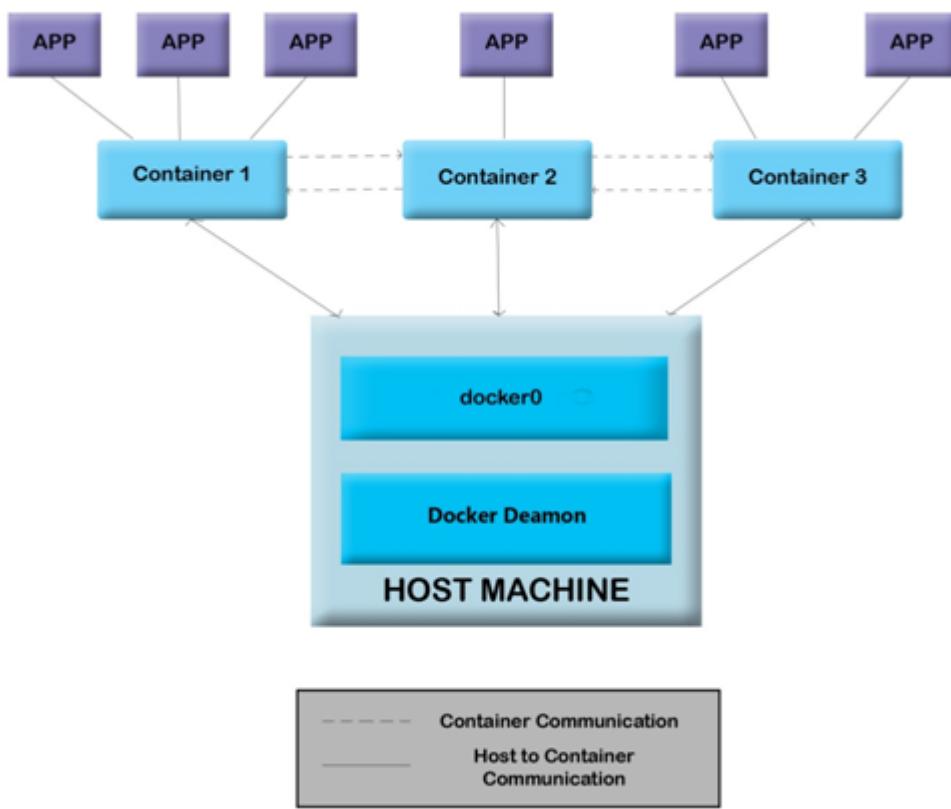


Figura 6 – Arquitetura do Docker (PREETH et al., 2015).

(contêiner). Foi mostrado também alguns serviços que utilizam virtualização como supercomputadores e ambientes para HPC.

### 3 TRABALHOS RELACIONADOS

A virtualização tem ajudado a tornar mais eficiente o uso de recursos computacionais. Por exemplo HPC, é uma das áreas que utiliza virtualização. Segundo (HWANG et al., 2013) O avanço do desempenho dos processadores e o uso de técnicas de virtualização ajudado a reduzir o custo computacional. Porém, ainda ocorrem sobrecargas, principalmente quando múltiplas VMs estão competindo por recursos (HWANG et al., 2013).

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

Para obter o melhor desempenho das aplicações, nos ambientes virtualizados, é importante fazer a escolha certa do tipo de *hypervisor* que utilizaremos para as nossas aplicações, pois é ele que provê o ambiente de *hardware* virtualizado para possibilitar alojar múltiplas VMs a partir de um computador físico. E conforme (ELSAYED; ABDELBAKI, 2013) o tipo de *hypervisor* utilizado influência no desempenho do ambiente de virtualização.

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

#### 3.1 Avaliação de *Hypervisors*

Nesta seção abordaremos 4 trabalhos que realizam a avaliação de desempenho de *hypervisors*. Os trabalhos escolhidos fazem comparações entre diferentes virtualizadores utilizando *benchmarks*, alguns deles comparam os resultados com o ambiente nativo para um *baseline*.

##### 3.1.1 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 trabalho, 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 foi realizado os passos descritos a seguir:

- Foi executado o NPB-3.3 com MPI com até 16 processos (usando 2 máquinas) e OMP (Open Multi-Processing) 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 de HPC não sofrem influência da ferramenta de gerenciamento de nuvem computacional escolhida. Isso porque a tecnologia de virtualização utilizada nos ambientes da experiência foi a mesma, o KVM. De acordo o autor, já era esperado que não houvessem diferenças significativas nos resultados obtidos.

Nesse trabalho foi analisado o mesmo *hypervisor* KVM no ambiente de HPC, e a avaliação de desempenho foi realizada utilizando o NPB-OMP e NPB-MPI com diferentes ferramentas de gerenciamento de nuvens computacionais. O trabalho não fica apenas na camada de virtualização, não há automatização dos experimentos e não consta se foi feito *tunning* para obter melhores resultados.

### 3.1.2 GRANISZEWSKI et al. 2016 - *Performance Analysis of Selected Hypervisors (Virtual Machine Monitors (VMMs))*

No trabalho de (GRANISZEWSKI; ARCISZEWSKI, 2016) é feita uma análise de desempenho dos *hypervisors*: Hyper-V; ESXi; OVM; VirtualBox e Xen (*software* livre). Para tal, foram utilizadas ferramentas de *benchmark* para avaliar o desempenho dos seguintes componentes: CPU (*nbench*), RAM (*ramspeed*), HDD (*Filebench*) e NIC (*netperf*). Os experimentos foram realizados em uma máquina física contendo as seguintes configurações:

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

Cada *hypervisor* foi avaliado de maneira isolada, o HD foi formatado entre as instalações dos virtualizadores. Uma VM com o SO Ubuntu 12.04 LTS foi criada em cada *hypervisor*. Os experimentos de performance foram realizados no mesmo SO, Ubuntu 12.04 LTS. Cada componente (CPU, memória RAM, HD e interface de rede) foi avaliado separadamente. Quando uma VM é criada, um número de *virtual CPU* (vCPU) é alocada. O estudo avaliou o desempenho da VM com uma e duas CPUs. Cada VM avaliada foi configurada com 2GB de memória RAM.

Depois de avaliar cada componente, uma avaliação geral foi realizada, que permitiu uma visão mais geral em cada desempenho das máquinas. O experimento foi a compilação do *Kernel Linux*, que foi feito duas vezes em cada máquina e o tempo necessário para isso foi feita uma média. Componentes avaliados e os *benchmarks* utilizados:

- CPU (*nbench*)
- NIC (*Netperf*)
- *Kernel compilation*
- HDD (*Filebench*)
- RAM (*ramspeed*)

De acordo com os resultados obtidos, o ESXi obteve melhor performance. Os resultados nesse trabalho também indicaram que na maioria dos casos os *hypervisors* do tipo 1 tem uma grande vantagem em relação aos *hypervisors* tipo 2, isso graças ao acesso direto aos recursos do sistema.

O autor deste trabalho concluiu que, a partir da análise dos resultados dos *benchmarks* os *hypervisors* mais adequados para o uso em empresas são os produtos da Microsoft e VMware. A maioria dos sistemas não possuem as características para virtualização em larga escala. XenServer também teve bons resultados pode ser uma boa opção para pequenas empresas. Produtos da Oracle podem ser recomendados para empresas que já utilizam soluções criadas pela companhia, por exemplo, banco de dados Oracle.

No final do trabalho, o autor defende a seguinte ideia quando é preciso escolher o tipo de *hypervisor* que desejamos executar nossas aplicações. De acordo com (GRANISZEWSKI; ARCISZEWSKI, 2016) A escolha deve ser feita levando em consideração os resultados dos *benchmarks*, mas também características e custo. Cada empresa deve selecionar uma solução (*hypervisor*) baseada em suas necessidades e finanças.

Nesse trabalho foi avaliado o desempenho dos componentes computacionais em virtualizadores diferentes, para essa experiência foram utilizados *benchmarks* específicos para avaliar cada componente individualmente. No trabalho não consta automatização dos experimentos, nem se foi feito *tunning* do *kernel* para obter melhor desempenho dos virtualizadores.

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

O trabalho de (HWANG et al., 2013) analisou 4 virtualizadores diferentes: Hyper-V, KVM, vSphere e Xen. Nesse experimento são instanciadas 4 VMs: uma delas é um simples *web service* acessado por um cliente, e as outras três foram criadas para gerar interferência. Esse experimento foi dividido em quatro fases: avaliação de desempenho da

CPU, Memória, Disco e Rede. Durante cada fase, todas as três VMs executam o mesmo *benchmark* e é medido o impacto do desempenho na VM *web service*. O desempenho dos *hypervisors* foi avaliado da seguinte maneira:

- CPU foi avaliado com o *bechmark Bytemark*.
- O *Ramspeed* é uma ferramenta que avalia o desempenho da cache e da largura de banda da memória.
- *Bonnie++* e *FileBench* o primeiro é um *benchmark* de vazão de disco, enquanto o segundo é um gerador de *workload*, que simula um servidor de emails, arquivos e servidor web.
- *Netperf* é utilizado para avaliar o desempenho dos recursos de rede.
- *Application Workloads* foi avaliado através do software (FREEBENCH, 2008), *free-bench* é uma ferramenta de para realizar *benchmark* multi-plataforma, fornecendo codificação de áudio, compressão de dados, processamento científico, HTML, cargas de trabalho de processamento de fotos, criptografia e descompactação.
- *Multi-Tenant Interference* para verificar o impacto de cada componente, CPU, Memória, Disco e rede, foi medido a resposta do HTTP enquanto cada um desses componentes foram estressados com diferentes *benchmarks*.

Nos resultados obtidos o *hypervisor* que teve melhor desempenho foi o vSphere, porém de acordo com o autor é difícil afirmar que um *hypervisor* será a melhor escolha qualquer para tipo de aplicação. O experimento defende que diferentes tipos de aplicações podem necessitar de diferentes tipos de *hypervisors*, a escolha depende muito da aplicação que se deseja executar e quais os recursos necessários.

O trabalho compara diferentes *hypervisors* através de execuções de *benchmarks*.

Nesse trabalho são utilizadas aplicações sintéticas, não consta automatização dos experimentos ou *tunning* do *kernel* para melhor desempenho.

### 3.1.4 ELSAYED *et al.* 2013 - *Performance Evaluation and Comparison of the Top Market Virtualization Hypervisor*

A fim de demonstrar qual o virtualizador mais adequado (ELSAIED; ABDELBAKI, 2013) fez uma pesquisa sobre os melhores virtualizadores disponíveis no mercado. Sendo assim, foi realizada a comparação quantitativa e qualitativa entre os *hypervisors*: VMware ESXi5, Microsoft Hyper-V2008R2 and Citrix Xen Server 6.0.2.

Todos os experimentos foram realizados utilizando o mesmo hardware, uma topologia de rede real, rodando os mesmos métodos de avaliação nas mesmas máquinas virtuais. A comparação foi realizada através de experimentos com ferramentas *open source* para

avaliar o desempenho da máquina sob diferentes *workloads*. Foram utilizados três *Intel-based host servers* com a seguinte especificação:- Dual Processor X5570 2.93 GHz, 24GB de memória DDR3 , 5 X 146GB SAS HD, e 4 X 1Gbps portas de rede. Em cada host server, uma máquina virtual foi criada com as seguintes configurações: 4 CPUs virtuais, 1x50GB HD, 20 GB RAM, Microsoft Windows Server 2008R2 enterprise x64, e SQL server 2008R2.

O objetivo do estudo foi caracterizar o desempenho dos virtualizadores, para isso foi utilizado diferentes *workloads*. A aplicação intensiva do disco é executada em uma única máquina virtual alocada em cada *hypervisor* a avaliação foi feita utilizando a aplicação DS2 e ferramentas de monitoramento de rede PRTG para a comparação dos três *hypervisors*: VMware ESXi5, Microsoft Hyper-V2008R2, e Citrix Xen Server 6.0.2. Os experimentos foram realizados sob a mesma configuração do sistema. Cada cenário foi dividido em dois experimentos.

- O primeiro experimento utiliza o DS2 para identificar qual *hypervisor* tem desempenho mais eficaz do banco de dados SQL;
- O segundo experimento utiliza um banco de Dados (SQL de 10GB) para simular 20 milhões de clientes, 100.000 produtos e 100.000 pedidos/mês como carga de trabalho. O foco é determinar o desempenho da utilização da CPU, memória consumida e E/S do disco.

Os 3 cenários foram configurados da seguinte maneira:

- O primeiro cenário começa com uma carga pequena; apenas uma máquina virtual executando o banco de dados SQL 10GB;
- O segundo cenário aumenta a carga criando 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 ajudou a identificar qual virtualizador é mais eficaz ao lidar com o banco de dados SQL, através dos resultados foi constatado que o VMware ESXi5 foi o melhor por causa do uso da CPU.

O segundo experimento foi feito com a intenção de identificar qual virtualizador faz uso mais eficaz da CPU e memória do servidor host depois de ser colocado sobre eles uma carga pesada. Constatou-se que o Citrix Xen Server 6.0.2 foi melhor seguido pelo Hyper-V2008R2 e o VMware ESXi5 no primeiro cenário. O segundo cenário o Hyper-V2008R2 foi o melhor seguido do VMware ESXi5. O Citrix Xen Server 6.0.2 não foi incluído nessa comparação devido ao baixo desempenho relatado pelo autor. O terceiro cenário demonstra que o Hyper-V2012RC tem melhor desempenho no uso da memória e operações de E/S do disco quando comparado com o Hyper-V 2008R2.

O autor do trabalho analisou 3 *hypervisors*: VMware ESXi5, Microsoft Hyper-V2008R2, e Citrix Xen Server 6.0.2 destes virtualizadores apenas o Citrix Xen Server é um software livre. Mais uma vez os resultados mostram que o desempenho dos *hypervisors* variam de acordo com as aplicações.

As experiências nesse trabalho foram realizadas apenas na camada de virtualização e não houve implementação de uma nuvem computacional. A avaliação de desempenho foi feita através do uso de *benchmarks*. Não consta automatização da avaliação nem detalhes de como os dados foram obtidos. Assim como o autor não relata se foi feito *tunning* dos *kernels* para melhor desempenho.

### 3.2 Avaliação *Hypervisors* x Contêineres

Diferente da seção 3.1 onde falamos sobre os trabalhos onde o foco era comparar o desempenho entre diferentes *hypervisors*, nesta parte do trabalho vamos apresentar 3 trabalhos que fazem a comparação entre *hypervisors* e a tecnologia de virtualização leve.

#### 3.2.1 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 uma avaliação comparativa de aplicações científicas, com uma instância e múltiplas instâncias em nuvens computacionais usando as tecnologias de virtualização KVM e LXC em uma nuvem privada na plataforma Cloudstack. Para a avaliação realizada nesse trabalho foi utilizado o NPB-OMP para simular diferentes tipos de *workloads*. O middleware de nuvem utilizado neste trabalho foi o CloudStack 4.8, três servidores idênticos executando o 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 2.80 GHz, com o *Hyperthreading* desabilitado intencionalmente;
- 24GB RAM (DDR3 1333MHz);
- SATA II HD
- 1 GB Conexão de rede

O *software* e os *benchmarks* são compilados usando o *GNU Compiler Collection* baseado no compilador Fortran na versão 4.8.5(*Red Hat 4.8.5-11*).

A estrutura da nuvem computacional consiste em um nó *front-end* para administração com outros dois nós para executar os *workloads* e as aplicações. Ambas localização de armazenamento, primário e secundário, são exportadas do nó *front-end* via NFS e são utilizadas para armazenar as imagens das VMs, *templates* e imagens do SO. A utilização de RAM alocada nas instâncias são 90% de toda capacidade do nó.

A Figura 7 mostra a visão arquitetural do trabalho realizado pelo autor. O NPB OMP executando e variando o número de *threads* para a avaliação do desempenho dos virtualizadores que estão alocados em um ambiente de nuvem computacional.

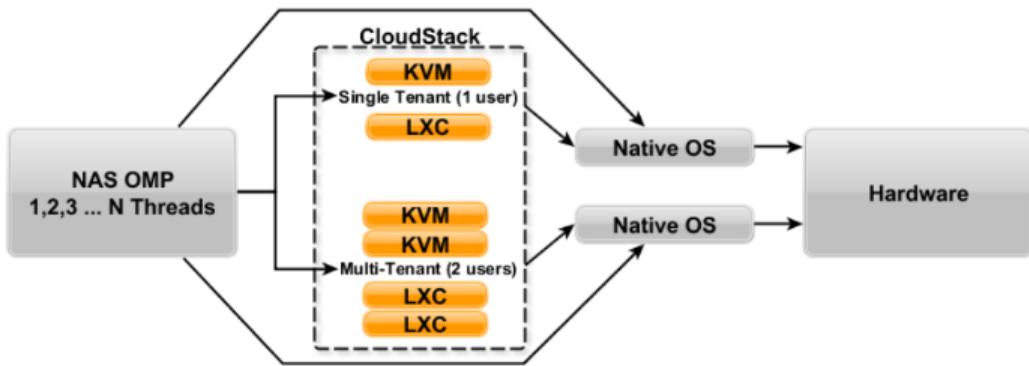


Figura 7 – Representação em alto nível da metodologia utilizada em (MALISZEWSKI et al., 2018).

O NPB-OMP foi utilizado e compilado com a classe B, os *benchmarks* executados foram: EP, FT, IS, MG e CG. Foi criado dois cenários: o primeiro é com uma instância, onde as nuvens computacionais CloudStack baseadas em LXC e KVM foram implementadas com todos os recursos da máquina. O segundo cenário é composto por duas instâncias de nuvens computacionais CloudStack baseada em LXC e duas nuvens computacionais CloudStack baseadas em KVM foram implementadas, e o serviço oferece a metade do total de recursos do *host*. O número de *threads* utilizado foi limitado pelo número de vCPUs disponíveis. No cenário com apenas uma instância foi executado o NPB-OMP até 8 *threads*, nos cenários com múltiplas (2) foi executado o NPB-OMP até 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.

Assim, constatou-se que a sobrecarga 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*, e ainda com uma diferença mínima. 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 . O alto uso da memória e o acesso nessas aplicações impactam significantemente o desempenho das instâncias KVM. Com a virtualização completa, a CPU precisa tratar

mais instruções, e consequentemente mais buferizações, que causa a perda de desempenho se comparado com o ambiente nativo.

A nuvem baseada em LXC é a melhor opção para aplicações científicas no cenário com apenas uma instância. Apesar da nuvem baseada em LXC ter melhores resultados com múltiplas instâncias, o número de resultados que não há diferença entre os ambientes de nuvens são cerca de 45%. Embora a nuvem LXC com múltiplas instâncias ter sido considerada a melhor opção, a nuvem baseada em KVM também teve bons resultados. O bom desempenho que a nuvem baseada em KVM com múltiplas instâncias, é por causa do isolamento de recursos e usuários. O que é o contrário do que acontece no ambiente LXC , onde há melhores resultados com uma instância e pior isolamento de recursos.

Não consta no trabalho do autor a automatização dos experimentos ou *tunning* nos *hypervisors*. Não são mencionados detalhes sobre a coleta dos dados e a criação dos gráficos.

### 3.2.2 MALISZEWSKI *et al.* 2018 - Desempenho em Instâncias LXC e KVM de Nuvem Privada Usando Aplicações Científicas.

Em (MALISZEWSKI; GRIEBLER; SCHEPKE, 2018), foi realizada uma análise de desempenho entre as tecnologias KVM e LXC em um ambiente de nuvem privada gerenciada pela plataforma *CloudStack*. Os experimentos foram realizados em um ambiente de nuvem composto por três servidores de configurações iguais. Um nó (*front-end*) responsável pela administração da nuvem computacional, e dois nós executam os experimentos.

Na avaliação de desempenho foi utilizado o NPB-OMP 3.3.1 executando a classe B. Executou-se para cada instância, 1 a 8 *threads*, resultando em 8 execuções separadas. Foi repetido a execução desse experimento 10 vezes. Uma nuvem baseada em LXC e outra em KVM foram implementadas sob o mesmo hardware. O autor do trabalho realizou uma comparação com o ambiente nativo para se ter um *baseline*.

Em relação ao trabalho de (MALISZEWSKI; GRIEBLER; SCHEPKE, 2018), concluiu-se que em 93,75% dos resultados o LXC foi melhor que o KVM. Nesse cenário em apenas um caso o KVM supera o LXC , com uma diferença mínima. Os autores relataram que o alto uso da memória e o acesso nesses aplicativos afetam o desempenho nas instâncias KVM. Isso ocorre porque há adição de mais instruções do que a CPU precisa gerenciar devido à virtualização completa, influenciando assim a perda de desempenho quando comparado com o ambiente nativo.

Nesse trabalho o autor relatou que a execução do experimento foi repetida 10 vezes, porém não deixa claro se foi feita de forma manual ou automatizada, também não consta no trabalho se houve *tunning* dos virtualizadores para obter melhor resultado nos experimentos, nem como foi realizado o *parsing* dos resultados.

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

Em (MORABITO; KJÄLLMAN; KOMU, 2015) é feita uma análise comparativa entre *hypervisors* e virtualizadores e a tecnologia *lightweight*. O *benchmark* foi realizado nos seguintes sistemas de virtualização KVM, LXC , Docker e OSv. Foi utilizado o desempenho do ambiente nativo para, comparar os resultados com o desempenho das demais tecnologias a fim de medir a sobrecarga dos virtualizadores. As ferramentas de *benchmarks* usaram *workloads* para medir CPU, Memória, E/S de Disco e E/S de Rede. A seguir é apresentado

- O desempenho da CPU foi medida utilizando o *Y-cruncher*. A aplicação foi executada 30 vezes. Foi utilizada a ferramenta *NBENCH*, para medir a performance da CPU. A diferença dessa ferramenta para a *Y-cruncher*, é que ela o *NBENCH single threaded*, executada em uma única *thread*. O *benchmark Linpack* avalia a performance do sistema usando um sistema de Álgebra Linear simples. Nesse algoritmo em particular foi utilizada, uma matriz de tamanho N=1000;
- Disco: Para avaliar o desempenho do disco, foi utilizado o *benchmark Bonnie++*, foi utilizado um arquivo de 25 GB, executadas leituras e escritas em sequência;
- Memória: o *benchmark STREAM* foi utilizado para avaliar o desempenho e a memória nos ambientes, os resultados são obtidos a partir de quatro operações: *Copy*, *Scale*, *Add*, e *Triad*;
- Rede (Network I/O Performance): foi utilizado o *benchmark Netperf* para medir o desempenho da rede entre dois *hosts*. Os experimentos foram configurados para executar de forma unidirecional e requisição/resposta de transferência de dados com os protocolos *TCP* e *UDP*.

Esses *benchmarks* foram executados com a finalidade de entender as diferenças de capacidade de processamento, armazenamento, memória e rede. De acordo com os resultados obtidos, as tecnologias baseadas em contêineres (LXC e Docker) tiveram melhor desempenho quando comparadas com KVM. Nesse trabalho consta que alguns *benchmarks* foram executados repetidas vezes, porém também não deixa claro se foi feito de forma manual ou não, o autor não consta *tunning* nos *kernels* para obter melhor desempenho nos resultados.

A Tabela 4 apresenta um o que foi feito nos trabalhos relacionados e faz uma comparação com o trabalho proposto. Não consta nos trabalhos relacionados se os experimentos foram realizados de forma automatizada ou não. Alguns desses trabalhos mencionam a execução dos *benchmarks* repetidas vezes, mas não é possível entender como realmente foi realizado. Outro detalhe interessante para este trabalho é que os autores

não comentam se foi feito algum tipo de melhoria a nível de *kernel* para obter melhor desempenho nos experimentos.

Autores	Plataformas	Workloads	Automatização	Tunning
DA SILVA <i>et al.</i> , 2018	Linux KVM VMWare ESXi Docker LXC	Aplicação sintética Aplicação real	BASH Python	Consta
MALISZEWSKI <i>et al.</i> , 2018 (a)	KVM LXC	Aplicação real NPB-OMP	NC	NC
MALISZEWSKI <i>et al.</i> , 2018 (b)	LXC, KVM	NPB-OMP	NC	NC
VOGEL <i>et al.</i> , 2016	KVM	NPB-3.3, MPI	NC	NC
GRANISZEWSKI <i>et al.</i> , 2016	Hyper-V ESXi OVM VirtualBox Xen	NBENCH Netperf Filebench Ramspeed	NC	NC
MORABITO <i>et al.</i> , 2015	KVM LXC Docker OSv	Aplicação real Y-cruncher NBENCH Linpack Bonnie++ Netperf	NC	NC
HWANG <i>et al.</i> , 2013	Hyper-V KVM vSphere Xen	Aplicação real Bytemark Ramspeed Bonnie++ FileBench Netperf	NC	NC
ELSAYED et at., 2013	VMware ESXi5 Citrix Xen Server 6.0.2 Hyper-V2008R2	Aplicação real DS2 PRTG	NC	NC

Tabela 4 – Tabela comparativa dos trabalhos relacionados.

### 3.3 Conclusão do capítulo

Este capítulo teve como objetivo apresentar o uso da virtualização para HPC.

Foram apresentados trabalhos que realizaram a avaliação de virtualizadores e contêineres com *benchmarks*. Inicialmente foi feita uma breve introdução sobre o uso das tecnologias. Depois, discorremos sobre cada trabalho, falando o que foi feito no trabalho, a metodologia utilizada, os resultados obtidos e o que não consta nos trabalhos. Foi possível constatar que nos trabalhos relacionados a conclusão sobre qual o virtualizador apresenta melhor desempenho varia de acordo com as aplicações e *workloads* utilizados nos experimentos, dessa maneira foi possível concluir que não há um *hypervisor* ou contêiner específico que terá o melhor desempenho para todo tipo de aplicação. Não consta em nenhum dos

trabalhos relacionados a automatização dos experimentos ou se realizaram *tunning* dos kernels para obter melhores resultados.



## 4 METODOLOGIA

Neste capítulo é introduzida a metodologia proposta para a avaliação. O foco deste trabalho é comparar o desempenho entre dois virtualizadores ou contêineres executando aplicações sintéticas de maneira automática, uma dessas aplicações proposta neste trabalho é o *benchmarks* NPB-MPI, outros *benchmarks* estão sendo estudados como alternativa para essa experiência. Na Seção 4.1 discorremos sobre como será realizada a avaliação e a sua automatização. Na Seção 4.2 apresentamos o ambiente computacional escolhido para a realização dos experimentos. Na Seção 4.3 são demonstradas as métricas que serão levadas em consideração nos experimentos. E por fim, na Seção 4.4, é apresentada a conclusão do capítulo.

### 4.1 Avaliação de desempenho

Avaliaremos o desempenho de dois virtualizadores ou contêineres com o auxílio de *benchmarks*. Para atingir o objetivo deste trabalho, será desenvolvido um *script* para automatizar a avaliação da experiência. Os autores dos trabalhos utilizados como referência neste trabalho não mencionam automatização ao executar os *benchmarks* em seus trabalhos durante os experimentos, do mesmo modo que não fica claro como foi a etapa de coleta de dados ou como os gráficos foram criados. Logo, uma das principais motivações para o desenvolvimento deste *script* é contribuir para a reproduzibilidade dos experimentos deste trabalho, assim como de outros trabalhos futuros que desenvolveremos a partir deste, ao automatizar a avaliação de desempenho. Autores de outros trabalho também poderão utilizar o *script* para gerar os gráficos dos resultados de seus experimentos. Para tanto, o *script* será desenvolvido através da integração de BASH e Python. Seu código será *open source* e vamos disponibilizá-lo na plataforma de hospedagem de código fonte GitHub Inc. Embora o foco deste trabalho seja a avaliação de dois *hypervisors*, é necessário ressaltar que será possível realizar a mesma avaliação de desempenho em contêineres. Além de realizar a avaliação com o uso de aplicações sintéticas, em direções futuras pretendemos utilizar aplicações reais para comparar os resultados dos virtualizadores.

Neste trabalho vamos avaliar a performance dos virtualizadores KVM e VMware ESXi em diferentes cenários através da execução automática da ferramenta de *benchmarking* NPB-MPI. Essa suite de *benchmarks* foi escolhida pela sua grande variedade de aplicações científicas e também por ser um dos *benchmarks* utilizados nos trabalhos relacionados ((MALISZEWSKI et al., 2018) e (VOGEL et al., 2016)) que mais se assemelham com o objetivo do trabalho proposto. Para a avaliação de desempenho todas as execuções serão realizadas de forma automatizada em servidores. Será avaliado um virtualizador de cada vez, ou seja, todas as execuções serão realizadas com o primeiro virtualizador KVM e quando todas as execuções tiverem sido terminadas vamos realizar os mesmos experimentos com o segundo virtualizador (VMware ESXi). Ao final vamos comparar os resultados obtidos. Uma das metodologias que usaremos será semelhante à utilizada pelo

autor (MALISZEWSKI et al., 2018) onde é feito o *benchmarking* de dois virtualizadores em diferentes cenários.

Antes de começar os experimentos, será preciso preparar o ambiente nativo. Para isso, vamos instalar no servidor físico o SO Ubuntu, seguido da instalação do virtualizador KVM e fazer algumas configurações necessárias. Depois que o SO e o virtualizador KVM forem ~~corretamente~~ instalados iniciaremos a configuração dos cenários **para a avaliação**. Os cenários serão configurados da seguinte maneira: **1Guest** (1 máquina virtual ou **contêiner**), **nGuests** (n máquinas virtuais ou contêineres). Sendo assim, no primeiro caso vamos avaliar o desempenho do KVM, assim que concluirmos a avaliação desse virtualizador nos cenários definidos, avaliaremos o segundo *hypervisor* (ESXi) utilizando os mesmos cenários do KVM.

Cada cenário (número de *guests*) será avaliado individualmente. Partindo do princípio de que a configuração do ambiente computacional foi concluída com sucesso, iniciaremos o processo de criação automatizada dos sistemas virtualizados para cada cenário, conforme as etapas abaixo no ambiente nativo:

- Definir o número de vCPUs;
- Definir a quantidade de memória **RAM**;
- Definir o tamanho **do disco rígido** virtualizado;
- Instalar o SO Ubuntu Server na(s) VM(s) ou contêineres;
- Copiar os *benchmarks*;

Após essas configurações, o *script* estará pronto para realizar a avaliação do *hypervisor* ~~neste cenário~~. O NPB-MPI é uma suíte de *benchmarks* para realizar avaliações de desempenho acerca de recursos computacionais. A Figura 8 ilustra a automatização do processo de avaliação dos virtualizadores. O *script* será responsável por executar um *benchmark* após o outro e também automatizar os seguintes processos:

- Execução dos *benchmarks*;
- Coleta dos resultados obtidos;
- Geração de gráficos.

Os resultados gerados não são salvos na máquina onde o *benchmark* foi executado, a não ser que o usuário especifique antes da execução para a determinada aplicação salvar os resultados. Esse é um dos objetivos da automatização, fazer com que o NPB execute um *benchmark* após o ~~termino~~ do outro sem precisar ~~a~~ interferência do usuário, e cada final da execução do *benchmark* gerar um arquivo com os resultados obtidos. Esses arquivos serão utilizados para criar os gráficos dos resultados da avaliação.

A Figura 8 representa em alto nível o processo de avaliação. Em primeiro lugar existe a fase de configuração manual, que é a instalação do SO no ambiente nativo juntamente com algumas configurações necessárias. Logo após encontra-se a parte que o *script* automatizará: criação dos cenários, execução dos *benchmarks*, coleta de dados e geração dos gráficos.

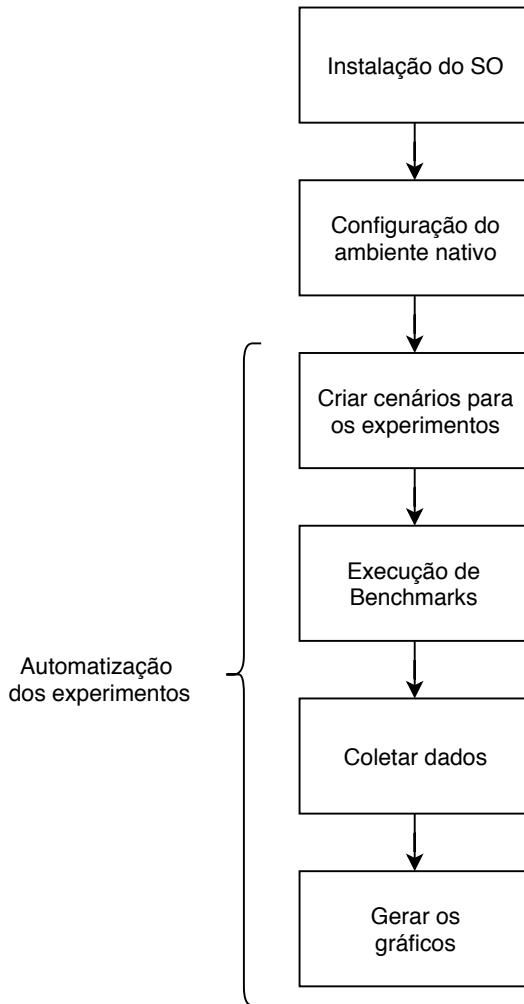


Figura 8 – Configurações dos experimentos.

Assim que todos os *benchmarks* forem executados para os experimentos no primeiro cenário, criaremos com auxílio do script o próximo cenário de avaliação. Daremos como finalizada a fase de avaliação do primeiro virtualizador assim que todos os cenários estiverem sido avaliados com sucesso, ou seja, a execução dos *benchmarks* sem erros e a coleta dos resultados obtidos das execuções. Para a continuação dos experimentos vamos formatar o disco rígido da **Workstation**, com o intuito de instalar o próximo virtualizador (VMware ESXi) para realizar a sua avaliação. Para avaliar o segundo virtualizador, são repetidos os mesmos passos para avaliar o KVM, que são: configuração do ambiente nativo e configurações necessárias, seguido da automatização da criação dos cenários, *benchmarking* e coleta dos dados.

Ao final dos experimentos teremos o resultado do desempenho dos virtualizadores KVM e VMware ESXi em cada um dos diferentes cenários que foram previamente descritos.

Assim vamos avaliar:

- O desempenho do mesmo virtualizador em diferentes cenários;
- O desempenho dos dois virtualizadores no mesmo cenário;

## 4.2 Ambiente dos experimentos

Os experimentos serão realizados em uma **Workstation** da Unipampa, campus Alegrete. A Tabela 5 apresenta a descrição das características do computador.

Componente	Informação
Modo(s) operacional da CPU	32-bit, 64-bit
Byte Order	Little Endian
CPU(s)	32
Thread(s) per núcleo	2
Núcleo(s) por soquete	8
Soquete(s)	2
Nó(s) de NUMA	2
ID de fornecedor	GenuineIntel
Família da CPU	6
Model name	Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz
CPU MHz	1995.235
CPU max MHz	2800,0000
CPU min MHz	1200,0000
Virtualização	VT-x
cache de L1d	32K
cache de L1i	32K
cache de L2	256K
cache de L3	20480K

Tabela 5 – Informação do servidor onde os virtualizadores serão instalados para avaliação.

## 4.3 Métricas de desempenho

Para entender as diferenças dos resultados e fazer uma análise comparativa dos virtualizadores, uma das técnicas que utilizaremos para a avaliação é semelhante a metodologia realizada em (ROSSO, 2015). Algumas das métricas de desempenho que pretendemos utilizar para avaliar os dois *hypervisors* neste trabalho são:

**Média do Tempo de Execução:** o tempo total de execução que será obtido através da média de 16 execuções onde serão descartadas os dois melhores e os dois piores tempos.

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

**Speed-Up:** É uma métrica que será utilizada para expressar quantas vezes a execução do *benchmark* paralelo ficou mais rápido 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.

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

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 mostra como foi 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 abaixo ilustra essa razão, onde  $S(p)$  é o speed-up e  $p$  é o número de unidades de processamento.

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

A eficiência ideal seria quando cada unidade de processamento ficasse ativa 100% durante todo o tempo de execução. No entanto, geralmente essas unidades tem que aguardar 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.

## 4.4 Conclusão do Capítulo

Neste capítulo abordamos a metodologia que será utilizada neste trabalho. Inicialmente falamos sobre como será realizada a avaliação do desempenho dos virtualizadores e a motivação para a sua automatização. Após, apresentamos as configurações do ambiente nativo onde serão realizados os experimentos, seguido pelas métricas de avaliação. É importante ressaltar que estamos realizando estudos sobre outros *benchmarks* que podem ser utilizados, e também há a possibilidade de avaliar tanto VMs como contêineres.



## 5 AVALIAÇÃO DE DESEMPENHO

A proposta deste trabalho consiste em avaliar o desempenho de sistemas de virtualização. Para isso, desenvolveremos um *script* que automatize o processo de execução de *benchmarks* e que, a partir dos resultados obtidos, consiga gerar gráficos de forma automática para análise. Assim, será possível verificar onde ocorrem, e se ocorrem, diferenças de desempenho dos virtualizadores quando executamos os *benchmarks*. A automatização da avaliação dos virtualizadores facilitará a continuidade e a reprodução do trabalho.

### 5.1 Proposta do trabalho

Para a avaliação automática do desempenho dos virtualizadores KVM e VMware ESXi, desenvolveremos um script que utilizará a suíte de *benchmarks* NPB-MPI.

A Figura 9 mostra a visão arquitetural do trabalho proposto. A primeira camada representa os componentes de *hardware*, que incluem, CPU, RAM e HD, do servidor físico onde serão realizados os experimentos. Na camada acima é localizado o *kernel*, que será responsável pelo escalonamento dos recursos físicos para os componentes de virtualização. A próxima camada, acima do *kernel*, é onde serão alocados os sistemas de virtualização para a realização dos experimentos. Os resultados obtidos da avaliação do desempenho serão utilizados para gerar os gráficos.

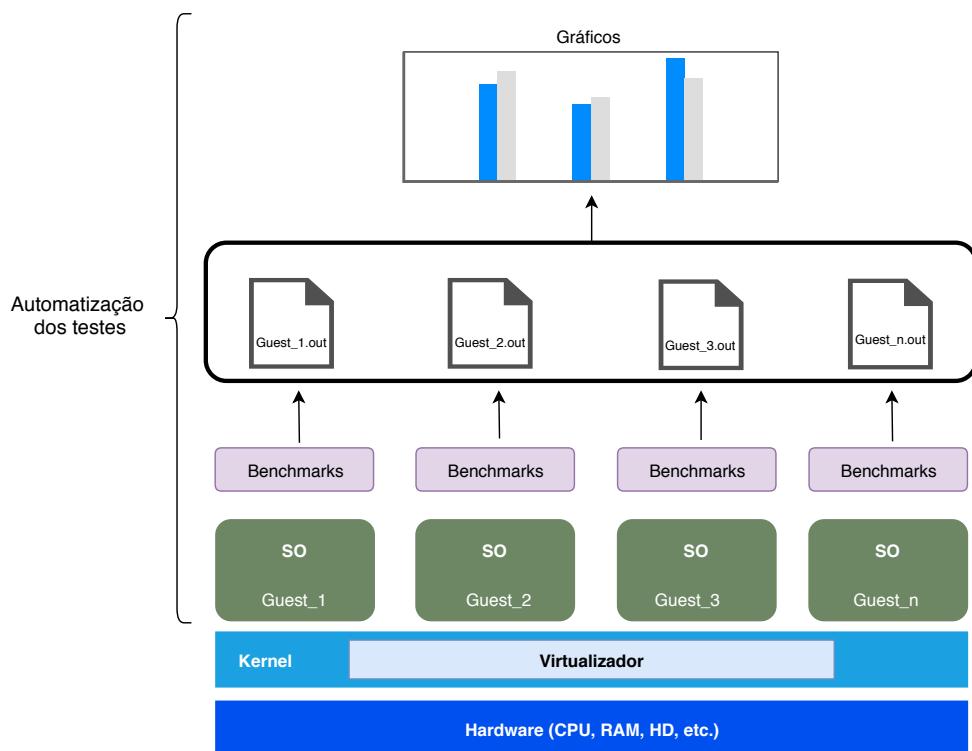


Figura 9 – Visão arquitetural do trabalho proposto.

Algumas das funções do script serão: instanciar *guests* para os experimentos, coletar os dados obtidos e criar gráficos. Para melhor compreensão de como será feita a

automatização destes passos descritos, desenvolvemos um pseudocódigo contendo algumas funcionalidades do script.

### loa 1 – Automatização dos experimentos

```

1: função PROVISIONAMENTO( )
2:     Definir arquivos de configuração dos benchmarks
3:     para i ← 1 até N faça
4:         Criar máquina virtual ou contêiner
5:         Instalar softwares requeridos por o NPB-MPI
6:         Instalar NPB-MPI
7:     fim para
8: fim função
9:
10: função EXECUÇÃO DOS benchmarks( )
11:    para i ← 1 até nBenchmarks faça
12:        Escolha benchmark em (IS, EP, CG, MG, FT, BT, SP, LU):
13:        para i ← 1 até 16 faça
14:            Executar benchmark escolhido
15:            Coletar resultados
16:        fim para
17:    fim para
18: fim função
19:
20: função PARSING DOS RESULTADOS E GERAÇÃO DOS GRÁFICOS(saídas)
21:    para 1 ← N até nSaídas faça
22:        parsing do arquivo: tempo de execução e medida de desempenho FLOPs
23:        geração de gráfico
24:    fim para
25: fim função
```

A primeira função do *script* será o **Provisionamento**. Nessa fase serão criadas as máquinas virtuais ou contêiners para os experimentos. Para isso, vamos utilizar o software *open-source* Vagrant (VAGRANT, 2018). Através do *script* e com o auxílio do Vagrant, vamos definir um número de máquinas virtuais ou contêineres para um cenário. A seguinte etapa será alocá-los. No próximo passo instalaremos os softwares que o NPB-MPI necessita para executar corretamente. Após vemos instalar o NPB-MPI . Por fim definiremos os arquivos de configurações dos *benchmarks*, por exemplo: quais os *benchmars* serão executados, a ordem de execução dos *benchmarks*, o tamanho da classe dos *benchmarks*, etc. Ao final da execução da função Provisionamento teremos um cenário pronto para realizar os experimentos.

A **Execução dos *Benchmarks*** será a próxima função que o script executará. A quantidade de *benchmarks* definida anteriormente controla o laço de repetição mais externo dessa função. Será escolhido o primeiro *benchmark* da fila para executar, também definido na função **Provisionamento**. Serão realizadas um total de 16 execuções para este *benchmark*, a saída de cada execução é armazenada em um arquivo individual para

esse *benchmark*. Ao final das 16 execuções, o *script* seleciona o próximo *benchmark* da lista de *benchmarks* e repete os passos anteriores, 16 execuções e coleta dos resultados, até que todos os *benchmark* sejam executados. Ao término desta parte, teremos os resultados de todos os experimentos.

A última função do *script* será o **Parsing dos Resultados e Geração dos Gráficos**. Essa função recebe como parâmetro os arquivos de saída dos resultados dos *benchmarks*. A partir dos dados obtidos desses arquivos o *script* criará 2 tipos de gráficos, um *contendo* o tempo de execução e outro com a medida de desempenho FLOPs de cada *benchmark*. Ao final da execução do script teremos os gráficos comparando os dois virtualizadores em todos os cenários criados, com os resultados dos benchmarks utilizados para a avaliação. A saída do *script* será semelhante à Figura 10 do trabalho de (MALISZEWSKI et al., 2018).

Os dados contidos na Figura 10 são de um experimento realizado pelo autor (MALISZEWSKI et al., 2018), esse exemplo é o resultado do *benchmark* IS da suíte NPB-OMP que avaliou a performance do *hypervisor* KVM, do contêiner LXC e comparou com o ambiente nativo, em ambientes distintos. O eixo *Number of threads* é o número de *threads* utilizadas em cada ambiente e eixo *Seconds* representa o tempo da execução do *benchmark* em segundos.

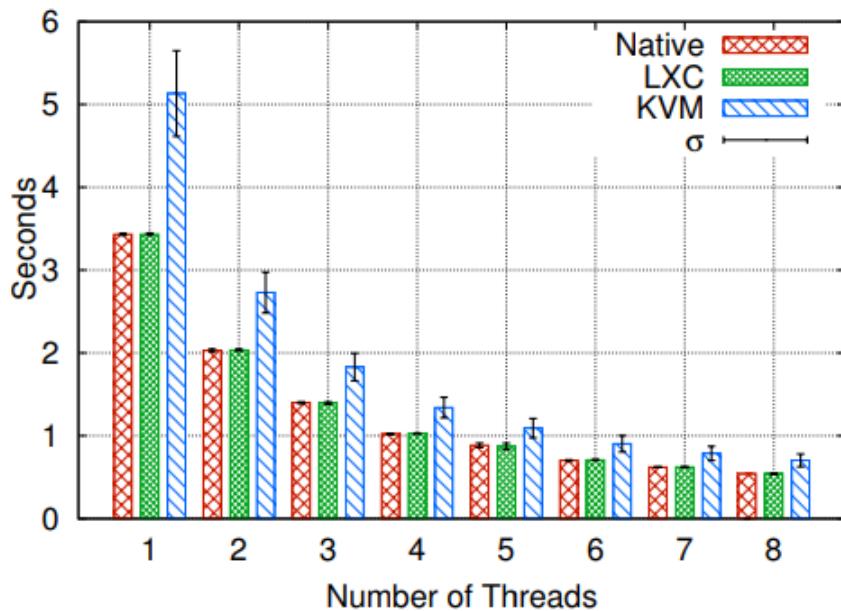


Figura 10 – Exemplo de saída do gráfico (MALISZEWSKI et al., 2018).

As diferenças dos gráficos do trabalho proposto em relação aos gerados em (MALISZEWSKI et al., 2018) é que, além de serem gerados automaticamente, vamos levar em consideração o tempo de execução e os FLOPs. Logo no gráfico tempo de execução, constará o tempo em segundos que o *benchmark* levará para executar, e a quantidade de

*guests* alocados em cada cenário. O outro gráfico constará a quantidade de FLOPs, será semelhante ao gráfico do tempo de execução, contendo os *guests* definidos em cada um dos cenários onde foram realizado os experimentos.

## 5.2 Conclusão do capítulo

Este capítulo teve como objetivo apresentar a proposta de avaliação dos virtualizadores KVM e VMware ESXi, assim como a proposta de desenvolver um *script* para automatizar os experimentos e coletar os dados das execuções dos *benchmarks*. Para isso foi apresentada uma visão arquitetural do trabalho proposto e um esboço do *script* em pseudocódigo. Por fim, abordamos algumas das principais funções do *script* e quais saídas são esperadas como resultado da sua execução exemplificando como saída; um dos gráficos criados em um dos trabalhos relacionados.

## 6 CONSIDERAÇÕES FINAIS

A computação de alto desempenho vem recebendo crescente atenção na área acadêmica e na área comercial, pelo fato de possuir vantagens como: tolerância à falhas e aproveitamento eficiente dos recursos computacionais disponíveis. É importante saber escolher o *hypervisor* certo para obter o melhor desempenho possível para as aplicações que desejamos executar. Dessa forma, este trabalho **têm** como objetivo realizar a avaliação do desempenho de 2 virtualizadores diferentes.

Realizaremos os experimentos com aplicações sintéticas. Um dos *benchmarks* que serão utilizados para os experimentos é o NPB, lembrando que outros *benchmarks* estão sendo estudados para serem utilizados no trabalho. Assim como estamos estudando a possibilidade de executar aplicações reais nessa experiência.

Outra contribuição deste trabalho é desenvolver um *script* para automatizar este processo de avaliação, dessa forma tornaremos mais simples a reproduzibilidade deste trabalho, assim como os trabalhos futuros. No final, pretendemos obter os resultados para avaliar se houve discrepâncias entre esses o desempenho dos virtualizadores, e se não houver, entender porque não. Além disso, se outros autores não desejarem reproduzir todos os experimentos realizados neste trabalho, pelo fato do *script* ser modular, pesquisadores e estudantes poderão utilizar o *script* para agilizar o processo de criação de gráficos científicos.

### 6.1 Cronograma de atividades

As tarefas serão distribuídas conforme a Tabela 6

ATIVIDADES	2018					2019					
	Ago	Set	Out	Nov	Dez	Jan	Fev	Mar	Abr	Mai	Jun
Revisão da literatura	X	X	X	-	-	-	-	-	-	-	-
Estudo sobre virtualização, virtualizadores e benchmarks	-	X	X	X	-	-	-	-	-	-	-
Esboço teórico da automatização	-	-	X	X	-	-	-	-	-	-	-
Entrega do TCC I	-	-	-	X	-	-	-	-	-	-	-
Pesquisa de <i>benchmarking</i> automatizado	-	-	-	-	X	X	-	-	-	-	-
Implementação o <i>script</i>	-	-	-	-	-	-	X	X	-	-	-
Realização dos testes	-	-	-	-	-	-	-	X	X	-	-
Coleta dos dados	-	-	-	-	-	-	-	-	X	X	-
Submissão de artigos	-	-	-	-	X	-	-	-	-	-	X
Entrega do TCC II	-	-	-	-	-	-	-	-	-	-	X

Tabela 6 – Cronograma de atividades.



## REFERÊNCIAS

- ARORA, R. **Conquering Big Data with High Performance Computing.** [S.l.]: Springer, 2016. Citado na página 23.
- 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 na página 27.
- BUYYA, R.; VECCHIOLA, C.; SELVI, S. T. **Mastering cloud computing: foundations and applications programming.** [S.l.]: Newnes, 2013. Citado 4 vezes nas páginas 13, 25, 26 e 27.
- CHAO, L. **Virtualization and Private Cloud with VMware Cloud Suite.** [S.l.]: CRC Press, 2017. Citado na página 30.
- DELL. **Virtualized HPC Performance with VMware vSphere 6.5 on a Dell PowerEdge C6320 Cluster.** 2017. [Online; accessed 5-June-2018]. Disponível em: <[http://en.community.dell.com/techcenter/high-performance-computing/b/general\\_hpc/archive/2017/04/05/virtualized-hpc-performance-with-vmware-vsphere-6-5-on-a-dell-powerededge-c6320-cluster](http://en.community.dell.com/techcenter/high-performance-computing/b/general_hpc/archive/2017/04/05/virtualized-hpc-performance-with-vmware-vsphere-6-5-on-a-dell-powerededge-c6320-cluster)>. Citado 3 vezes nas páginas 13, 29 e 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 33 e 36.
- FEO, J. T. An analysis of the computational and parallel complexity of the livermore loops. **Parallel Computing**, v. 7, n. 2, p. 163 – 185, 1988. ISSN 0167-8191. Disponível em: <<http://www.sciencedirect.com/science/article/pii/0167819188900373>>. Citado na página 27.
- FLYNT, C.; LAKSHMAN, S.; TUSHAR, S. **Linux Shell Scripting Cookbook.** Packt Publishing, 2017. ISBN 9781785882388. Disponível em: <<https://books.google.com.br/books?id=yHc5DwAAQBAJ>>. 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 36.
- GRANISZEWSKI, W.; ARCISZEWSKI, A. Performance analysis of selected hypervisors (virtual machine monitors-vmmms). **International Journal of Electronics and Telecommunications**, De Gruyter Open, v. 62, n. 3, p. 231–236, 2016. Citado 2 vezes nas páginas 34 e 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 3 vezes nas páginas 21, 33 e 35.
- KVM. **Main Page — KVM.** 2016. [Online; accessed 5-June-2018]. Disponível em: <[https://www.linux-kvm.org/index.php?title=Main\\_Page&oldid=173792](https://www.linux-kvm.org/index.php?title=Main_Page&oldid=173792)>. Citado 3 vezes nas páginas 9, 11 e 22.

LEIRIA, R. D. Monitoramento energético em nuvens computacionais. Universidade Federal do Pampa, 2016. Citado na página 21.

MALISZEWSKI, A. M.; GRIEBLER, D.; SCHEPKE, C. Desempenho em instâncias lxc e kvm de nuvem privada usando aplicações científicas. 2018. Citado 2 vezes nas páginas 33 e 40.

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 7 vezes nas páginas 13, 21, 38, 39, 45, 46 e 53.

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 41.

MPI. MPI. 2018. [Online; accessed 12-november-2018]. Disponível em: <<https://www.open-mpi.org/>>. Citado na página 27.

NAS. NASA Advanced Supercomputing Division. 2018. [Online; accessed 2-november-2018]. Disponível em: <<https://www.nas.nasa.gov/publications/npb.html>>. Citado 2 vezes nas páginas 27 e 28.

OPENMP. Home - OpenMP. 2018. [Online; accessed 10-November-2018]. Disponível em: <<https://www.openmp.org/>>. Citado na página 26.

PREETH, E. et al. Evaluation of docker containers based on hardware utilization. In: **IEEE Control Communication & Computing India (ICCC), 2015 International Conference on**. [S.l.], 2015. p. 697–700. Citado 3 vezes nas páginas 13, 31 e 32.

ROSSO, J. P. Análise de desempenho de aplicações científicas em ambiente de nuvem privada. Universidade Federal do Pampa, 2015. Citado na página 48.

STERLING, T.; ANDERSON, M.; BRODOWICZ, M. **High Performance Computing: Modern Systems and Practices**. [S.l.]: Morgan Kaufmann, 2017. Citado 2 vezes nas páginas 25 e 27.

TANENBAUM, A.; BOS, H. **Modern Operating Systems**. Pearson, 2015. ISBN 9780133591620. Disponível em: <<https://books.google.com.br/books?id=9gqnngEACAAJ>>. Citado 4 vezes nas páginas 13, 21, 29 e 30.

TOP500. **TOP500 SUPERCOMPUTER SITES**. 2018. [Online; accessed 10-november-2018]. Disponível em: <<https://www.top500.org/>>. Citado 3 vezes nas páginas 15, 24 e 25.

TRINDADE, L. V. P.; COSTA, L. H. M. Análise do desempenho da virtualização leve para ambientes com edge computing baseada em nfv. In: **Simpósio Brasileiro de Redes de Computadores (SBRC)**. [S.l.: s.n.], 2018. v. 36. Citado 2 vezes nas páginas 30 e 31.

TRINITIS, C.; WEIDENDORFER, J. **Co-scheduling of HPC Applications**. [S.l.]: IOS Press, 2017. v. 28. Citado 2 vezes nas páginas 28 e 29.

VAGRANT. **Vagrant by HashiCorp.** 2018. [Online; accessed 20-november-2018]. Disponível em: <<https://www.vagrantup.com/>>. Citado na página 52.

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 9, 11 e 22.

VOGEL, A. et al. Medindo o desempenho de implantações de openstack, cloudstack e opennebula em aplicações científicas. **16th Escola Regional de Alto Desempenho do Estado do Rio Grande do Sul (ERAD/RS)**, p. 279–282, 2016. Citado 3 vezes nas páginas 33, 34 e 45.

XAVIER, M. G. et al. Performance evaluation of container-based virtualization for high performance computing environments. In: IEEE. **Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on**. [S.l.], 2013. p. 233–240. Citado 2 vezes nas páginas 13 e 31.



## ÍNDICE

CPU, 21, 25, 30, 33–37, 39–41, 48, 51  
PE, 26  
FLOP, 15, 23–25, 52–54  
GB, 34, 38, 41  
HD, 27, 51  
HPC, 13, 23, 28, 29, 32–34, 42  
KVM, 9, 11, 21, 22, 30, 33–35, 38–41, 45–  
48, 51, 53, 54  
LXC, 30, 38–41, 53  
MIMD, 26  
MPI, 22, 26, 27, 34, 45, 46, 51, 52  
NPB, 22, 27, 28, 34, 38–40, 42, 45, 46,  
51, 52, 55  
OMP, 26, 34, 38–40, 53  
PC, 23  
RAM, 27, 38, 51  
SO, 29–31, 33, 34, 38, 46, 47  
vCPU, 34, 39, 46  
VM, 21, 29, 30, 33–38, 46, 49  
VMM, 29