

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 do bem que,
como profissionais contribuem para um mundo melhor.



AGRADECIMENTOS

Em primeiro lugar agradeço à Deus por ter me guiado durante essa longa jornada. À minha família e em especial a minha mãe, por sua capacidade de sempre acreditar em mim. À 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 jamais mediram esforços para me ajudar sempre que eu precisei. Muito obrigado à todos vocês, sempre levarei comigo um pouco de cada um para o resto da minha vida.

“Não vos amoldeis às estruturas deste mundo,
mas transformai-vos pela renovação da mente,
a fim de distinguir qual é a vontade de Deus:
o que é bom, o que Lhe é agradável, o que é perfeito.
(Bíblia Sagrada, Romanos 12:2)

RESUMO

Virtualizadores são componentes fundamentais para a computação de alto desempenho, 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 o virtualizador escolhido. Não existe um virtualizador que seja o mais adequado para todas aplicações, na verdade o desempenho do virtualizador depende do tipo de aplicação que será executada. O presente trabalho tem como objetivo avaliar o desempenho de dois tipos de virtualizadores: *Linux KVM* (KVM, 2016) e *VMware ESXi* (VMWARE, 2018), verificar discrepâncias e entender qual deles teve o melhor desempenho nos experimentos a serem realizados. Para a avaliação utilizaremos *benchmarks*. Por fim, vamos propor a automação das seguintes funções: execução dos *benchmarks*, coleta de dados e geração de gráficos.

Palavras-chave: Virtualizadores. Linux KVM. VMware ESXi. Benchmark. Automação.

ABSTRACT

Hypervisors are key components for high-performance computing, they are responsible for hardware abstraction and providing virtualized resources to guests. Depending on the type of application that will run in virtualized environments the performance may vary significantly according to the chosen hypervisor. There is no virtualizer that is best suited for all applications, in fact the performance of the hypervisor depends on the type of application that will be performed. The present work aims to evaluate the performance of two types of virtualizers Linux KVM (KVM, 2016) and VMware ESXi (VMWARE, 2018), check discrepancies and understand which of them had the best performance in the experiments to be performed. For the evaluation we will use benchmarks. Finally, we will propose the automatization of the following functions: benchmarking, data collection and generation of graphs

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

LISTA DE FIGURAS

Figura 1 – Representação de um sistema computacional MIMD (BUYYA; VEC-	
CHIOLA; SELVI, 2013)	26
Figura 2 – Sistema computacional com memória compartilhada (esquerda) e me-	
mória distribuída (direita) (BUYYA; VECCHIOLA; SELVI, 2013) . . .	27
Figura 3 – Benefícios da virtualização em HPC (DELL, 2017)	29
Figura 4 – Hypervisor tipo 1 (esquerda) e tipo 2 (direita) (TANENBAUM; BOS,	
2015)	30
Figura 5 – Virtualização leve	31
Figura 6 – Representação em alto nível da metodologia de avaliação do trabalho	
proposto (MALISZEWSKI et al., 2018)	41
Figura 7 – Configurações dos experimentos	49
Figura 8 – Visão arquitetural do trabalho proposto	53
Figura 9 – Exemplo de saída do gráfico (MALISZEWSKI et al., 2018)	55


LISTA DE TABELAS

Tabela 1 – Desempenho computacional em operações de ponto flutuante por segundo (Flops)	23
Tabela 2 – A tabela apresenta o primeiro do lugar de cada ano dos supercomputadores com melhor capacidade de processamento do mundo de acordo com o site top500.org	24
Tabela 3 – A tabela mostra o primeiro do lugar de cada ano dos supercomputadores mais potentes do mundo em eficiência energética de acordo com o site top500.org	25
Tabela 4 – Comparativo dos Trabalhos relacionados	45
Tabela 5 – Informação da Workstation onde os benchmarks serão executados . . .	50
Tabela 6 – Cronograma de atividades	57

LISTA DE SIGLAS

SUMÁRIO

1	INTRODUÇÃO	21
1.1	Objetivos e contribuições	22
1.2	Organização do documento	22
2	CONTEXTUALIZAÇÃO	23
2.1	Computação de Alto Desempenho (HPC)	23
2.2	Paralelismo	26
2.3	Benchmarking	27
2.4	Virtualização para HPC	29
2.5	Hypervisors	30
2.6	Contêineres	31
2.7	Computação em nuvem	32
2.8	Conclusão do capítulo	33
3	TRABALHOS RELACIONADOS	35
3.1	Avaliação de Hypervisors	35
3.1.1	VOGEL, A. et al. 2016 - Medindo o Desempenho de Implantações de Openstack, Cloudstack e Opennebula em Aplicações Científicas.	35
3.1.2	GRANISZEWSKI et al. 2016 - Performance Analysis of Selected Hypervisors (Virtual Machine Monitors (VMMs)) . . .	36
3.1.3	HWANG, J. et al. 2013 - A Component-Based Performance Comparison of Four Hypervisors.	38
3.1.4	ELSAYED et al. 2013 - Performance Evaluation and Comparison of the Top Market Virtualization Hypervisor	39
3.2	Avaliação Hypervisors x Contêineres	40
3.2.1	MALISZEWSKI, A. M. et al. 2018 - The Nas Benchmark Kernels for Single and Multi-Tenant Cloud Instances with LXC/KVM	40
3.2.2	MALISZEWSKI et al. 2018 - Desempenho em Instâncias LXC e KVM de Nuvem Privada Usando Aplicações Científicas. . .	42
3.2.3	MORABITO et al. 2015 - Hypervisors vs. Lightweight Virtualization: A Performance Comparison	43
3.3	Conclusão do capítulo	44
4	METODOLOGIA	47
4.1	Avaliação de desempenho	47
4.2	Ambiente dos experimentos	50
4.3	Métricas de desempenho	50

4.4	Conclusão do Capítulo	51
5	AVALIAÇÃO DE DESEMPENHO 	53
5.1	Proposta do trabalho	53
5.2	Conclusão do capítulo	56
6	CONCLUSÃO E TRABALHOS FUTUROS	57
6.1	Cronograma de atividades	57
	REFERÊNCIAS	59

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 simplifica o gerenciamento de máquinas virtuais (VMs), acelera a implantação de novos computadores e melhora a eficiência do uso dos recursos disponíveis. Além disso, também permite que usuários executem diferentes tipos de sistemas operacionais e várias VMs 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 um, mas duas técnicas de virtualização desenvolvidos de forma independente: SIMMON e CP-40 (TANENBAUM; BOS, 2015)

Um das principais características dessa tecnologia é fazer com que *workloads* de uma máquina física possam ser multiplicados em vários outros usuários (VMs). Dessa forma, é possível fazer o uso eficiente do computador físico ao atribuir tarefas a um conjunto de nós, além disso economiza-se energia, pois estaremos utilizando um ambiente virtual numa mesma máquina, poupando o uso servidores físicos que consomem uma quantidade significativa de energia e contribuem com a emissão de CO₂ no meio ambiente, estudos já estão sendo realizados para monitorar esse consumo energético (LEIRIA, 2016).

Apesar das vantagens obtidas através do seu uso, existe um custo. O *hypervisor*, primeira camada mais perto do *bare metal hardware* que é responsável pela virtualização, adiciona uma sobrecarga pelo fato de ter que adicionar uma camada de abstração entre a VM e os recursos físicos (Hardware) que ela usa (MALISZEWSKI et al., 2018). Vários aspectos do software e sistema operacional podem afetar o desempenho dos virtualizadores e VMs, por exemplo, como o *hypervisor* escalona recursos como, CPU, memória, disco, e rede são fatores críticos (HWANG et al., 2013). A combinação de novas arquiteturas de 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, VMWare, Hyper-V, etc., que podem ser instalados em quase todos computadores. Cada um deles possui suas próprias características, assim cada tipo de tecnologia realiza o gerenciamento dos recursos de computadores de maneira diferente. Essa variedade de *hypervisors* trás novos desafios em entender qual a melhor escolha a ser feita dependendo do tipo de aplicação que se deseja executar no ambiente virtualizado. A plataforma de virtualização que é mais adequada para uma dada aplicação depende da natureza dos requisitos de recursos e os tipos de aplicações que serão executadas.

1.1 Objetivos e contribuições

Neste trabalho vamos propor uma análise comparativa entre dois *hypervisors*: Kernel-based Virtual Machine (KVM) (KVM, 2016) e VMware ESXi (VMWARE, 2018), os testes serão realizados em diferentes cenários com um número definido de *guests*, focando também no paralelismo das aplicações. Para isso, usaremos os *benchmarks* MPI da ferramenta de benchmark desenvolvida pela NASA Advanced Supercomputing Division (NAS), o NAS Parallel Benchmarks (NPB) (NASA, 2018), que representa uma grande variedade de classes de aplicações científicas. Por fim, vamos avaliar os resultados obtidos deste trabalho e propor a automatização dos experimentos e *parsing* dos dados.



1.2 Organização do documento

O trabalho está organizado da seguinte maneira. No Capítulo 2 é apresentado alguns conceitos básicos sobre os virtualizadores, computação de alto desempenho e *benchmarking*. No Capítulo 3 estão os artigos utilizados como referência para o desenvolvimento deste trabalho. No Capítulo 4 veremos 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 e também um pseudo código que será usado como base para o desenvolvimento do script. E por fim, no Capítulo 6 encontra-se a conclusão e direções de trabalhos futuros.

2 CONTEXTUALIZAÇÃO

O objetivo deste capítulo é apresentar conceitos básicos sobre computação de alto desempenho, *benchmarks*, virtualizadores e contêineres. Em primeiro lugar na seção 2.1 falamos sobre a computação de alto desempenho e algumas métricas utilizadas para avaliação de computadores e supercomputadores. Na seção 2.3 falamos sobre as aplicações utilizadas para realizar avaliação do desempenho desses computadores. Na seção 2.4 introduzimos uma técnica utilizada para aumentar o desempenho dos computadores. Na seção 2.5 e na seção 2.6 falamos das tecnologias que são necessárias para a implantação da virtualização. Na seção 2.7 mostramos um exemplo do uso da virtualização. E na seção 2.8 a conclusão do capítulo.

2.1 Computação de Alto Desempenho (HPC)

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


Uma maneira de avaliar o desempenho dos computadores em *HPC* é a usar a métrica: Operações de ponto flutuante por segundo ou Flops. O Flop é uma adição ou multiplicação de dois números reais (ou ponto flutuante) representado numa forma que o computador consegue processar e manipular. Os primeiros computadores tinham tecnologia para alcançar apenas cerca de 1 KiloFlops (KFlops). Atualmente um simples computador comercial desempenha alguns GigaFlops (GFlops). Já um supercomputador tem uma capacidade de processamento que é muito mais poderosa que PCs de acordo com essa métrica, alcançando a casa dos Teraflops (TFlops). A Tabela 1 mostra em ordem crescente a capacidade de processamento de computadores em Flops.


Esse método 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 mais potentes de cada ano, utilizando sempre os dados do mês de novembro, a partir do ano 2000 até o ano de


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 operações de ponto flutuante por segundo (Flops)

Ano	Nome supercomputador	Cores	Rmax (TFlop/s)	Rpeak (TFlop/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 do lugar de cada ano dos supercomputadores com melhor capacidade de processamento do mundo de acordo com o site top500.org 

2018. A Tabela 2 está organizada da seguinte maneira, a coluna **Ano** indica qual o supercomputador mais potente 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 uma dessas máquinas possui. A quarta coluna **Rmax (TFlops/s)** mostra o desempenho máximo alcançado pelo supercomputador quando é executado um aplicação  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 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 de processamento (**Rmax Flop/s**) é as 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, nem sempre a capacidade de processamento está ligada à quantidade de núcleos que o computador possui, por exemplo o computador que está em primeiro lugar na lista desse ano (2018) possui uma quantidade inferior de cores que o 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 dessas 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 do processamento do mundo nos meses de junho e novembro anualmente, a crescente preo-

Ano	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 lugar de cada ano dos supercomputadores mais potentes do mundo em eficiência energética de acordo com o site top500.org

cupação com o meio ambiente e o consumo energético desses servidores chamou atenção da equipe do top500.org, que em 15 de novembro de 2007 eles inauguraram a primeira lista Green500.

A lista Green500 classifica os 500 melhores supercomputadores do mundo em eficiência energética. O foco das operações computacionais de desempenho a qualquer custo levou ao surgimento de supercomputadores que consomem grandes quantidades de energia elétrica e produzem tanto calor que grandes instalações de resfriamento precisam ser construídas para garantir um desempenho adequado. Para lidar com essa tendência, a lista Green500 valoriza o desempenho energeticamente eficiente para a supercomputação sustentável. A Tabela 3 mostra as máquinas com melhor 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 de 2018 - o supercomputador é do mês de junho).

A Tabela 3 é um pouco diferente da tabela anterior, visto que ela leva em consideração a capacidade de processamento por Watts. Na coluna **Ano** temos os computadores 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 faz por Watt (**GFlop/W**), quanto maior for a quantidade de flops feitos por Watt melhor é a classificação do computador. Por fim a última coluna mostra a quantidade de energia consumida pela máquina.

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

A verdadeira capacidade de um supercomputador está em sua habilidade de obter resultados satisfatórios dado uma meta, através de simulação por exemplo. Outra métrica também levada em consideração é o tempo que o computador leva para resolver um determinado problema(STERLING; ANDERSON; BRODOWICZ, 2017). Existem aplicações computacionais que são utilizados especificamente para esse tipo de avaliação.

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 vários processos (*tasks*) que são executados simultaneamente para resolver um determinado problema. Uma tarefa (*task*) é dividida em múltiplas sub tarefas (*subtasks*) através de técnicas como divisão e conquista por exemplo, e cada uma dessas sub tarefas é processada em uma unidade de processamento (CPU) diferente (BUYA; VECCHIOLA; SELVI, 2013).

Muitas aplicações atualmente necessitam de um maior poder computacional do que os computadores que não eram capazes de paralelizar as tarefas conseguiam oferecer. Com o paralelismo é possível proporcionar soluções viáveis para esses 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 conseguimos elevar o desempenho de um computador, e alcançar uma melhor performance (BUYA; 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 elemento de processamento MIMD possui instruções e fluxos de dados separados. Computadores MIMD trabalham de maneira assíncrona (BUYA; VECCHIOLA; SELVI, 2013). Essas máquinas são classificadas como memória compartilhada e memória distribuída, pela maneira que os elementos de processamento são alocados na memória principal. A Figura 1 representa um sistema computacional MIMD.

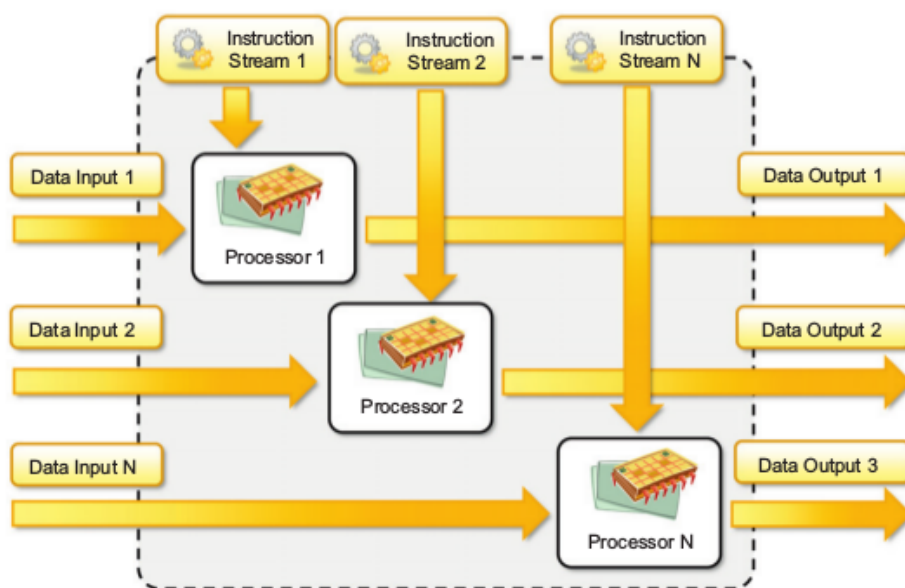


Figura 1 – Representação de um sistema computacional MIMD (BUYA; VECCHIOLA; SELVI, 2013)

No modelo *memória compartilhada* todos elementos de processamento estão conectados em uma única memória principal e todos eles tem acesso à ela [2]. A comunicação entre os elementos de processamento é feita através a memória que é compartilhada entre eles, as modificações dos dados feitas na memória por um dos EPs é visível para todos os outros EPs.

Na memória *distribuída*, todos os EPs possuem memória local. A comunicação entre os EPs nesse modelo é feita através da conexão à rede. Cada EP 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 (BUYA; VECCHIOLA; SELVI, 2013).

A Figura 2 ilustra os dois tipos de memórias mencionadas.

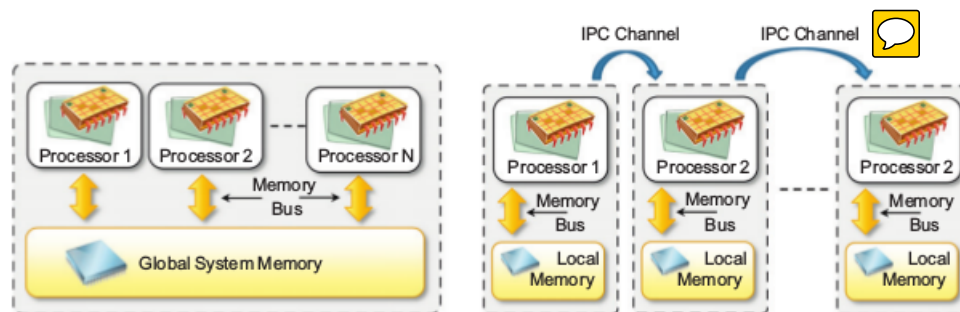


Figura 2 – Sistema computacional com memória compartilhada (esquerda) e memória distribuída (direita) (BUYA; VECCHIOLA; SELVI, 2013)

Para executar aplicações em computadores de forma **paralela** é possível utilizar interfaces de programação **paralela**, como por exemplo o OpenMP, para memória compartilhada e o MPI para memória distribuída.

O OpenMP 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.

2.3 Benchmarking

Dá-se o nome de **benchmarking** a prática de executar um *benchmark* (ou um conjunto de *benchmarks*) para avaliar o desempenho de componentes de um computador, como por exemplo: disco (hd), memória (RAM), rede, I/O, virtualizadores, entre outros. Um exemplo de *benchmark* é o NAS Parallel Benchmarks (NPB) historicamente o NPB não foi o primeiro *benchmark* desenvolvido, antes dele existiam outros (BAILEY et al., 1991) como o *LINPACK Benchmark*, que é utilizado até hoje para avaliar o desempenho dos computadores nas tabelas 2 e 3 e o outro *benchmark* era o *Livermore loops*.

O benchmark *Linpack* HPL (*High Parallel Linpack*) resolve uma série de equações lineares na forma matricial. 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. Eles variam de 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 são um conjunto de *benchmarks* que tem como objetivo realizar a avaliação de desempenho de computadores. Eles são desenvolvidos e mantidos pela NASA Advanced Supercomputing (NAS) (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. 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 a máquinas altamente paralelas.

O NAS Parallel Benchmarks (NPB) são conjuntos de *benchmarks* desenvolvidos para avaliar o desempenho de supercomputadores em que é possível executar aplicações em paralelo. Esses *benchmarks* são derivados das aplicações dinâmica dos fluidos computacionais (CFD) e é composto por 5 *kernels* e 3 pseudo-aplicações, são eles:


- 5 *Kernels*

- IS (Integer Sort) - Ordenação de inteiros, acesso randômico à memória.
- EP - Embarrassingly Parallel
- CG - Gradiente Conjugado, acesso a memória irregular e comunicação
- MG - Multi-Grid em uma seqüência de malhas, comunicação de longa e curta distância, memória intensiva
- FT - Transformada rápida de Fourier 3D discreta, comunicação all-to-all

- 3 Pseudo-aplicações

- BT - Solucionador tri-diagonal de blocos
- SP - Solucionador pentádico escalar
- LU - Lower-Upper Gauss-Seidel solver

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

- 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 computadores tenham alta capacidade de processamento atualmente, apenas algumas aplicações de HPC conseguem explorar a verdadeira capacidade computacional. Por isso, considera-se a virtualização uma opção para tirar o máximo de proveito dos supercomputadores (TRINITIS; WEIDENDORFER, 2017).

A virtualização é a capacidade de criar computadores virtuais dentro de um computador físico (máquina host). Essa máquina provê a plataforma de hardware para a máquina virtual (VM). Múltiplas máquinas virtuais podem ser alocadas em uma máquina host. Essa tecnologia possui benefícios como tolerância a falhas e segurança (DELL, 2017). A Figura 3 destaca ilustrativamente as características da virtualização.

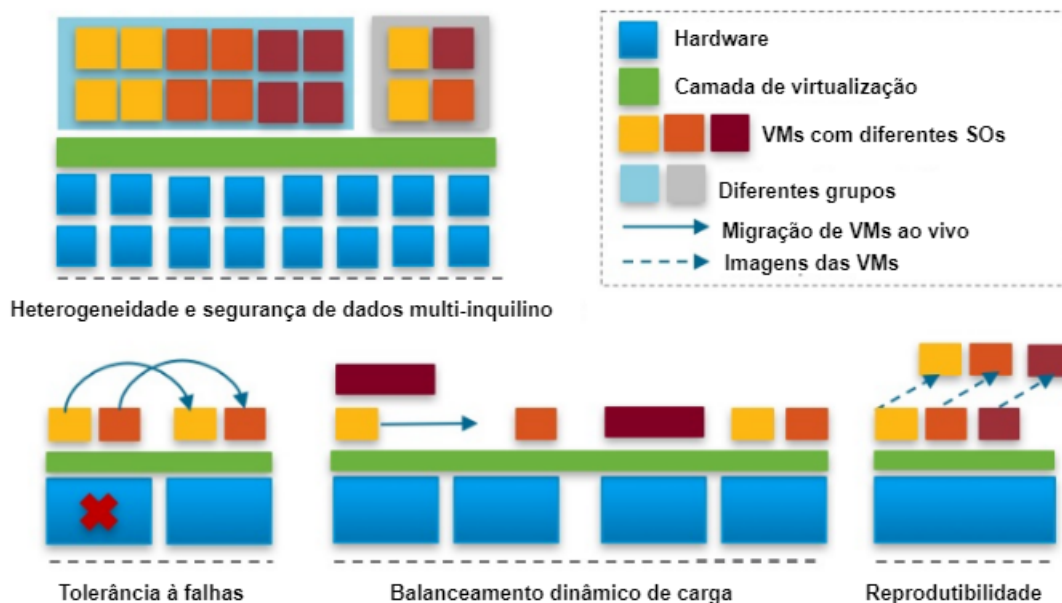


Figura 3 – Benefícios da virtualização em HPC (DELL, 2017)

A virtualização permite que um único computador seja o hospedeiro de múltiplas máquinas virtuais, cada uma executado potencialmente um sistema operacional diferente.

A ideia principal é que um Monitor de Máquina Virtual (VMM — Virtual Machine Monitor) cria a ilusão de múltiplas máquinas (virtuais) 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 máquinas virtuais compartilhando o mesmo hardware físico. Existem duas abordagens para virtualização. Hypervisors do tipo 1 e tipo 2.

Hypervisor tipo 1: é como um sistema operacional, já que é o único *software* executando no modo mais privilegiado. O seu trabalho é dar suporte a múltiplas cópias do hardware real, chamadas máquinas virtuais, similares aos processos que um sistema operacional 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 ele está instalado. Esse virtualizador não depende de um sistema operacional 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).

Hypervisor tipo 2: é um *software* que depende de outro sistemas operacional para alocar e escalonar recursos. O *hypervisor* tipo 2 simula ser um computador completo com uma CPU e vários dispositivos (TANENBAUM; BOS, 2015). O *Kernel-based Virtual Machine* ou simplesmente KVM é um *hypervisor* tipo 2. Ele consiste em um módulo de kernel carregável, KVM. ko, 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).

A figura 4 a seguir ilustra os dois tipos de hypervisors mencionados.

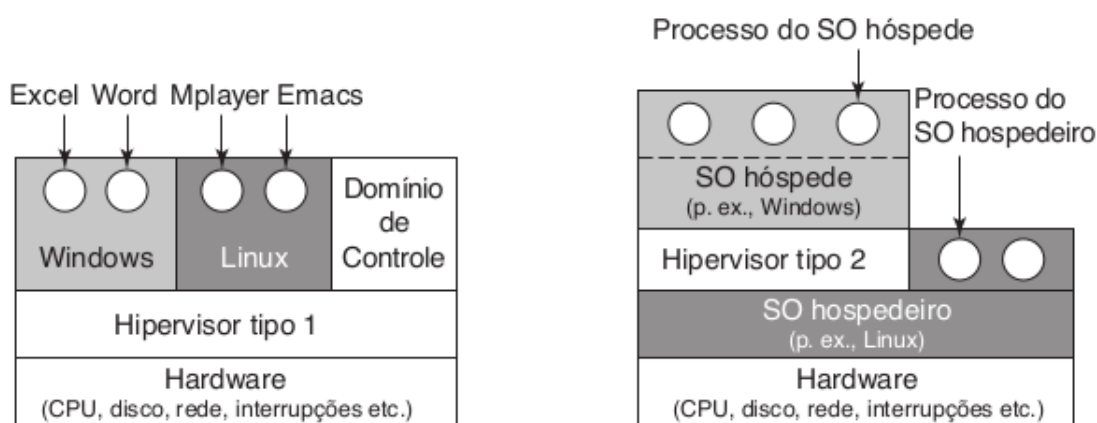


Figura 4 – Hypervisor tipo 1 (esquerda) e tipo 2 (direita) (TANENBAUM; BOS, 2015)

Além dessa tecnologia (*hypervisors*) utilizada para a virtualização, também podemos fazer o uso de *contêineres* através dela também é possível fazer o uso da virtualização.

2.6 Contêineres

Conforme mencionado anteriormente, os *hypervisors* fornecem uma abstração de hardware, a qual resulta em uma sobrecarga em termos da própria virtualização do hardware e dos *drivers* virtuais do dispositivo. Isso significa que cada instância de VM é totalmente implementada com hardware virtual para suportar um sistema operacional guest não modificado, executando sobre o *hypervisor*. Contêineres por outro lado fornecem um isolamento de processo a nível do sistema operacional, ou seja, para cada instância nova não é criada uma pilha completa de sistema operacional dedicado. Contêineres executam sobre o mesmo kernel do sistema operacional 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 contêiner (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 programas. Isso informa o kernel quais os recursos que estão disponíveis para um determinado processo executando num contêiner. Um contêiner pode ter acesso limitado a outros dispositivos, conexão a internet, memória e assim por diante. Esse controle faz com que não seja possível causar algum erro em outro, ou potencialmente danificar o sistema (FLYNT; LAKSHMAN; TUSHAR, 2017).

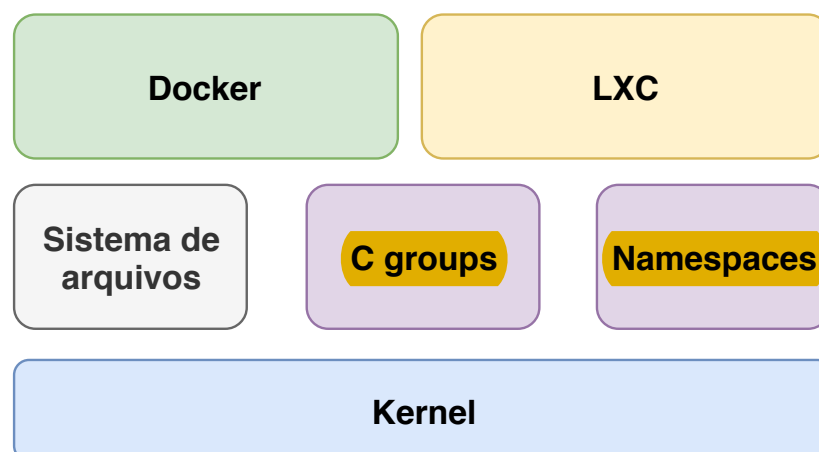



Figura 5 – Virtualização leve 

A virtualização baseada em contêineres é considerada uma alternativa leve à virtualização baseada em *hypervisors*, uma vez que contêineres implementam isolamento de processos a nível do sistema operacional da máquina hospedeira. Assim, evita-se a



sobrecarga devido ao **hardware** virtualizado e aos *drivers* virtualizados de dispositivos. 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 Computação em nuvem

Independente da tecnologia utilizada para a virtualização, é possível usar qualquer uma delas em nuvens computacionais. A computação em nuvem é o termo utilizado para descrever um ambiente de computação baseado em uma imensa rede de servidores, sejam estes virtuais ou físicos. Em outras palavras, é um conjunto de recursos como capacidade de processamento, armazenamento, conectividade, plataformas, aplicações e serviços disponibilizados na Internet. O resultado é que a nuvem pode ser vista como um estágio mais evoluído do conceito de virtualização, a virtualização do próprio *datacenter*. (TAURION, 2009)

De acordo com (TANENBAUM; BOS, 2015) a tecnologia de virtualização tem um papel fundamental no crescimento da computação em nuvem. Existem tipos diferentes de nuvens, algumas são **públicas** e estão disponíveis para qualquer um disposto a pagar pelo uso desses recursos; outras são **privadas**, pertencem a uma organização. Diferentes nuvens computacionais oferecem diferentes recursos. Acesso ao hardware físico ou ao seu ambiente virtualizado, algumas oferecem acesso diretamente as máquinas, virtuais ou não, outras oferecem um software para usuários desenvolverem novos serviços. Elas podem ser divididas em diferentes categorias (TANENBAUM; BOS, 2015):

- **IAAS (Infrastructure As A Service — Infraestrutura como um serviço)** oferece acesso direto a uma máquina virtual, a qual o usuário pode usar da maneira que ele achar melhor. Desse modo, a mesma nuvem pode executar sistemas operacionais diferentes, possivelmente no mesmo hardware.
- **PAAS (Platform As A Service — Plataforma como um serviço)** proporciona um ambiente que inclui questões como um SO específico, banco de dados, servidor da web, e assim por diante)
- **SAAS (Software As A Service — Software como um serviço)** oferece acesso a softwares específicos, como o Microsoft Office 365, ou Google Apps) e muitos outros tipos.

(TAURION, 2009) cita algumas das principais características das nuvens computacionais:

- **O usuário tem a sensação** de disponibilidade de recursos infinitos, acessíveis sob demanda.


- **Elimina a necessidade** de adquirir e provisionar recursos antecipadamente.
- **Oferece elasticidade**, permitindo que empresas (ou usuários) utilizem os recursos que forem necessários, aumentando e diminuindo a capacidade computacional de forma dinâmica.
- **O pagamento dos serviços em nuvem** é pela quantidade de recursos utilizados *pay-per-use*


A computação em nuvem aparece como uma alternativa, pois aloca recursos computacionais à medida que eles sejam demandados. Se houver maior demanda de transações, mais recursos são alocados. Se a demanda diminuir, esses recursos são liberados para outras aplicações. A computação em nuvem é portanto uma maneira bastante eficiente de maximizar e flexibilizar os recursos computacionais (TAURION, 2009)


2.8 Conclusão do capítulo


O objetivo **desse** capítulo foi apresentar uma visão geral sobre o *benchmarking* de computadores, fornecendo conceitos básicos de virtualização e virtualizadores **para o entendimento do restante deste trabalho**. Foram apresentados **as** características **essenciais dos** tipos de *hypervisors* e também **a** tecnologia *lightweight* (contêiner). Foi mostrado **também** os principais serviços que utilizam **vantagem** da virtualização **como nuvens computacionais, supercomputadores e ambientes para HPC**.

3 TRABALHOS RELACIONADOS

A virtualização tem ajudado a tornar mais eficiente o uso de recursos computacionais, por exemplo: *datacenters*, *cloud computing* e *HPC*, são algumas das áreas que utilizam virtualização. 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 ocorrem sobrecargas, principalmente quando múltiplas VMs estão competindo por recursos. 

De acordo com (MALISZEWSKI; GRIEBLER; SCHEPKE, 2018) a sobre carga ocorre porque há uma 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 sobre carga é menor. 

Um cenário ideal seria não houvesse essa perda de desempenho por causa da virtualização, ou pelo menos, que o desempenho do ambiente virtualizado fosse semelhante ao do ambiente nativo, algo que não é possível até o momento. 

Equanto não podemos obter o máximo de desempenho do ambiente virtualizado é importante que façamos a escolha certa do tipo de *hypervisor* que vamos utilizar para as nossas aplicações, pois é ele que provê o ambiente de hardware virtualizado para que seja possível instanciar múltiplas VMs a partir de um computador físico, e conforme (ELSAYED; ABDELBAKI, 2013) o tipo de *hypervisor* utilizado influencia no desempenho do ambiente de virtualização. 

Assim, foi realizada uma pesquisa para entender como os diferentes tipos de *hypervisors* e contêineres se comportam em diferentes situações de avaliações. 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

Nessa seção abordaremos 4 trabalhos que realizam a avaliação de desempenho de *hypervisors*. Os artigos escolhidos fazem comparações entre diferentes virtualizadores utilizando *benchmarks* e alguns deles comparam os resultados com o ambiente nativo para uma *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 experimento foi avaliado apenas o KVM

e o ambiente nativo para a realização dos experimentos e comparação dos resultados.

No ambiente nativo e nas instâncias foi utilizado o sistema operacional 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 (Message-Passing Interface) 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. Portanto, já era esperado que não houvessem diferenças significativas nos resultados obtidos.

Nesse trabalho foi analisado o mesmo hypervisor KVM em 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 artigo 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 (net-perf). Os experimentos foram realizados em uma máquina física contendo as seguintes configurações:

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

Cada hypervisor foi testado 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. Deve ser destacado que o Oracle VirtualBox, foi o único hypervisor tipo 2 que foi executado no Windows Server 2012. Os experimentos de performance foram realizados

no mesmo SO, Ubuntu 12.04 LTS. Cada componente (CPU, memória RAM, HD e interface de rede) foi testado separadamente. Quando uma VM é criada, um número de CPUs (vCPU) é alocada. O estudo testou o desempenho da VM com uma e duas vCPUS. Cada VM testada foi configurada com 2GB de memória RAM.

Depois de testar 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 foi líder em performance. Os resultados nesse artigo também indicaram que claramente a maioria dos casos os *hypervisors* do tipo 1 levam uma grande vantagem em relação aos *hypervisors* tipo 2, isso graças ao acesso direto aos recursos do sistema.

O autor concluiu que a partir da análise dos resultados dos *benchmarks* os principais candidatos 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. (GRANISZEWSKI; AR-CISZEWSKI, 2016) diz: *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 artigo foi avaliado o desempenho dos componentes computacionais em virtualizadores diferentes, para essa experiência foram utilizados *benchmarks* específicos para avaliar cada componente individualmente. No artigo 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.

Esse artigo é usado como referência em (GRANISZEWSKI; ARCISZEWSKI, 2016). 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: *benchmarking* 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 *benchmark* *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 testado através do software (FREEBENCH, 2008), *freebench* é uma ferramenta de *benchmarking* multi-plataforma, fornecendo codificação de áudio, compressão, 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* é melhor 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.

(HWANG et al., 2013) relatou que o desempenho de cada virtualizador pode variar drasticamente dependendo do tipo de aplicação e da alocação de recursos. Com os resultados obtidos, chegou-se a conclusão de que diferentes tipos de *workloads* podem ser executados de forma mais eficaz por diferentes virtualizadores. O estudo demonstra o quão importante é para datacenters e ambientes de nuvens computacionais, suportar diferentes tipos de virtualizadores e *hardware*.

O artigo 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 para as áreas de TI e usuários (ELSAYED; 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, 1 X 50GB H.D., 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* o benchmarking foi feito 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) que simula 20 milhões de clientes, 100.000 produtos e 100.000 pedidos / mês como carga de trabalho pesada. O foco é determinar o desempenho da utilização da CPU, memória consumida e E/S do disco.

Os 3 cenários são configurados da seguinte maneira:

- O primeiro cenário começa com uma carga leve; apenas uma máquina virtual executando o banco de dados SQL 10GB.



- O segundo cenário aumenta a carga criando 4 VMs; cada uma executando o **10GB SQL** 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, seguido em ordem decrescente pelos Citrix Xen Server 6.0.2 e o Hyper-V2008R2 no primeiro cenário; e seguido em ordem decrescente pelo Hyper-V2008R2 e o Citrix Xen Server 6.0.2.

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 seguindo do VMware ESXi5, o Citrix Xen Server 6.0.2 ficou fora dessa comparação devido ao desempenho ruim quando comparado com os outros dois. 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 artigo 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 artigo 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 artigos onde o foco era comparar o desempenho entre diferentes *hypervisors*, nesta parte do trabalho vamos apresentar 3 artigos 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 apenas um **inquilino** e múltiplos **inquilinos** em nuvens computacionais **usando as** tecnologias de virtualização KVM e LXC em uma nuvem privada na plataforma Cloudstack. Para **a experiência** realizada nesse trabalho foi utilizado 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 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
- 1GbE Conexão de rede

O software e *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 as *workloads* e 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 sistema operacional. A utilização de memória alocada (RAM) nas instâncias são 90% de toda capacidade do nodo.

A Figura 6 mostra a uma visão arquitetural do trabalho realizado pelo autor. O NAS OMP sendo executado variando o número de *threads* para a avaliar o desempenho dos virtualizadores que estão alocados em um ambiente de nuvem computacional.

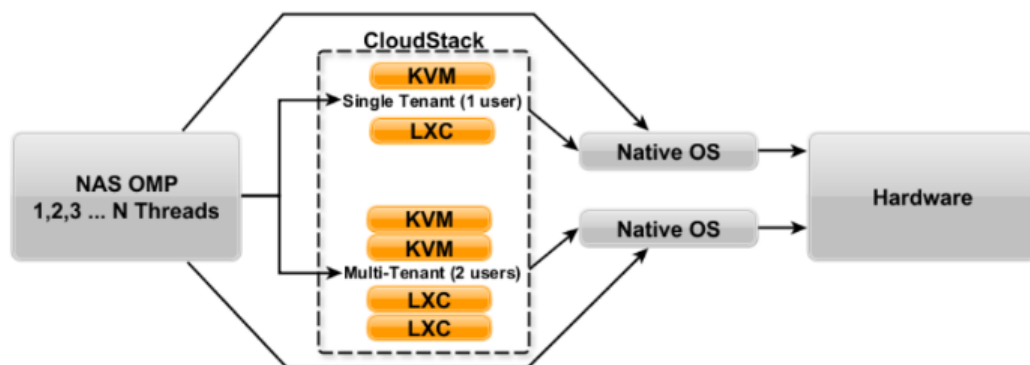


Figura 6 – Representação em alto nível da metodologia de avaliação do trabalho proposto (MALISZEWSKI et al., 2018)

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 implantadas

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 implantadas, e o serviço oferece a metade do total de recursos do host. O número de *threads* utilizados foi limitado pelo número de vC-PU's disponíveis. No cenário com apenas uma instância foi executado o NPB-OMP até 8 *threads*, no cenários com múltiplas (2) foi executado o NPB-OMP até 4 *threads*. Os experimentos também foram executados no ambiente nativo sem virtualização. Os cenários são 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 significativamente 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 também ter tido melhores resultados com múltiplas intâncias, o número de resultados que não há diferença entre os ambientes de nuvens são cerca de 45%. Embora a nuvem LXC com múltiplas intâncias ter sido considerada a melhor opção, a nuvem baseada em KVM pode ser considerada uma opção adequada. 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.

O autor do trabalho não consta 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 (hypervisor) e LXC (contêiner) em um ambiente de nuvem privada gerenciada pela plataforma CloudStack. Os experimentos foram realizados

em um ambiente de nuvem composto por três servidores de configuração iguais. Um nó (*front-end*) responsável pela administração da nuvem computacional, e dois nós são responsáveis pela execução dos experimentos.

Na avaliação de desempenho foi utilizado o NPB-OMP 3.3.1 compilando a classe B. Executou-se para cada instância de 1 a 8 *threads*, resultando 8 execuções separadas. Foi repetido a execução desse experimento 10 vezes. Uma nuvem baseada em LXC e outra em KVM foram implementadas sobre o mesmo hardware. Foi feito também uma comparação com o ambiente nativo para se ter uma *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, e ainda assim com uma diferença mínima. Os autores relatam que o alto uso da memória e o acesso nesses aplicativos afetam o desempenho das instancias KVM. Isso ocorre porque há uma adição de mais instruções que a CPU precisa gerenciar devido a virtualização completa, influenciando assim a perda de desempenho quando comparado com o ambiente nativo.

Nesse trabalho é realizada a comparação de desempenho entre as tecnologias LXC e KVM através do uso do *benchmark* NASPB-OMP 3.3.1, o autor relata que a execução foi repetida a execução do experimento 10 vezes, porém não deixa claro se foi feita de forma manual ou automatizada, também não consta no artigo 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

O autor de (GRANISZEWSKI; ARCISZEWSKI, 2016) usou esse trabalho como referência. Em (MORABITO; KJÄLLMAN; KOMU, 2015) é feita uma análise comparativa entre *hypervisors* baseados em virtualizadores e novas tecnologias como *lightweight*. O *benchmarking* foi realizado sobre os ambientes KVM, LXC, Docker, e OSv. Foi utilizado o desempenho do ambiente nativo para comparar as métricas com as demais tecnologias a fim de medir a sobrecarga dos virtualizadores. As ferramentas de *benchmarks* usaram cargas genéricas de *workloads* para medir CPU, Memória, E/S de Disco e E/S de Rede:



- O desempenho da CPU foi medida utilizando o *Y-cruncher*. A aplicação foi executada 30 vezes. Também foi utilizada a ferramenta *NBENCH*, que é utilizada para medir a performance da CPU. A diferença dessa ferramenta para a *Y-cruncher*, é que ela é *single threaded*, executada em uma única *thread*. O benchmark *Linnpack* testa a performance do sistema usando um sistema de álgebra linear simples, nesse algoritmo em particular foi utilizada uma matriz de tamanho $N=1000$, e executada 15 vezes.

- **DISCO:** Para avaliar o desempenho do disco, foi utilizado o benchmark *Bonnie++*, foi utilizado um arquivo de 25 GiB, executadas leituras e escritas em sequência.
- **MEMÓRIA:** o benchmark *STREAM* foi utilizado para avaliar o desempenho a memória nos ambientes, os resultados são obtidos a partir de quatro operações: Copy, Scale, Add, and 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 em contêiner (LXC e Docker) tiveram melhor desempenho quando comparada 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. E não consta a metodologia do *parsing* dos resultados.

A Tabela 4 descreve resumidamente os experimentos realizados nos artigos utilizados como referência neste trabalho e faz uma comparação com o trabalho que vamos propor. Nenhum dos trabalhos relacionados comenta se os experimentos foram feitos de forma automatizada ou não, alguns desses artigos mencionam a execução dos *benchmarks* n vezes, mas não é possível entender como realmente foi realizado, tampouco a construção dos gráficos contém informações relevantes sobre como foi feita. Outro detalhe interessante é que os autores não comentam se foi feita algum tipo de melhoria a nível de kernel para obter melhor desempenho dos experimentos ou para trabalhos futuros.

3.3 Conclusão do capítulo

Este capítulo teve como objetivo apresentar o uso da virtualização para HPC e em outros contextos. Foi apresentado um total de 7 artigos sobre o assunto. Inicialmente foi feita uma breve introdução sobre o uso dessas tecnologias e o que esperar do desempenho. Depois discurremos sobre cada artigo, falando o que foi feito, a metodologia utilizada, os resultados obtidos e o que não consta nos artigos. Foi possível constatar que entre os trabalhos relacionados utilizados a conclusão sobre qual o virtualizador apresentava melhor desempenho sempre variava de acordo com as aplicações e workloads utilizados nos experimentos, dessa maneira foi possível concluir que não há um hypervisor ou contêiner específico que será ter o melhor desempenho para qualquer tipo de aplicação. Não consta em nenhum dos trabalhos relacionados a automatização dos experimentos ou se fizeram *tunning* para obter melhores resultados.

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

Tabela 4 – Comparativo dos Trabalhos relacionados

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 quando executado ferramentas de benchmarking de maneira automática, um dos benchmarks proposto neste trabalho é o NPB-MPI, outros benchmarks estão sendo estudados como alternativa para essa experiência. Na Seção 4.1 discutiremos sobre como será realizada a avaliação e a sua automatização. Na Seção 4.2 descreveremos 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 artigos utilizados como referência neste trabalho não mencionam automatização ao executar os benchmarks em seus artigos 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, umas das principais motivações para o desenvolvimento deste script é contribuir para a reprodutibilidade 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 do foco deste trabalho seja a avaliação de dois hypervisors, é necessário ressaltar que será possível realizar a mesma avaliação de desempenho em contêineres, uma vez que as execuções serão automatizadas. 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 todos 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 às 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  *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 ?? ilustra a execução dos *benchmarks* mencionados anteriormente. 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 dados em um arquivo. Esse é um dos objetivos da automatização, fazer com o 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 de cada aplicação.

A Figura 7 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.

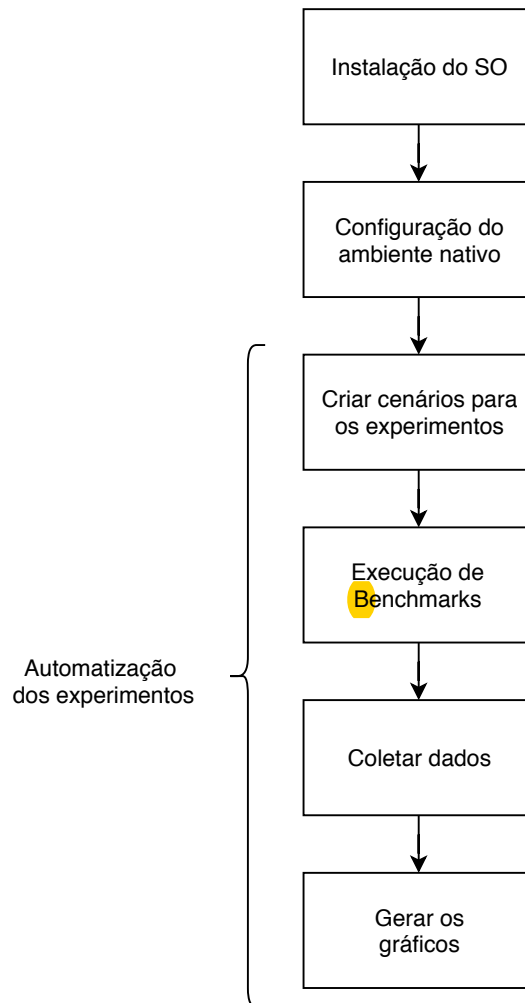


Figura 7 – Configurações dos experimentos

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 da Workstation onde os benchmarks serão executados

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 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 automatização da avaliação. 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 na avaliação, e também há a possibilidade de avaliar VMs e 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*. 

5.1 Proposta do trabalho

Para a avaliação de desempenho dos virtualizadores KVM e VMware ESXi utilizaremos a suíte de *benchmarks* NPB-MPI, e vamos automatizar esse processo com auxílio do *script* que vamos desenvolver. A Figura 8 mostra a visão arquitetural do trabalho proposto. A primeira camada representa os componentes de *hardware*, como por exemplo, CPU, memória RAM e disco rígido (HD), do servidor físico onde serão realizados os experimentos. Na camada acima temos o *kernel*, que será responsável pelo escalonamento dos recursos físicos para os componentes de virtualização. A próxima camada, acima do *kernel*, é onde serão alocadas os sistemas de virtualização para a realização dos experimentos. Os resultados obtidos da avaliação do desempenho serão utilizados para gerar os gráficos.

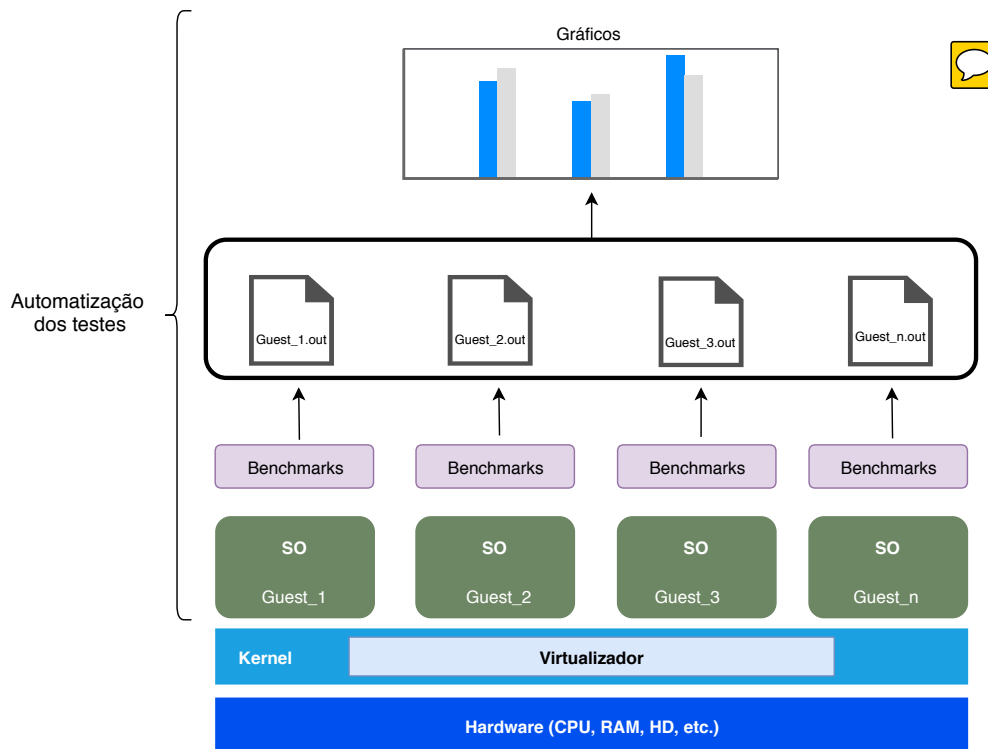


Figura 8 – Visão arquitetural do trabalho proposto

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

loa 1 – Automação dos experimentos

```

1: função PROVISIONAMENTO( )
2:   para i ← 1 até N faça
3:     Criar máquina virtual ou contêiner N
4:     Instalar softwares requeridos por o NPB-MPI
5:     Instalar NPB-MPI
6:     Definir arquivos de configuração dos benchmarks
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(saidas)
21:   para 1 ← N até nSaidas faça
22:     parsing do arquivo: tempo de execução e medida de desempenho (flops)
23:     geração de gráfico
24:   fim para
25: fim função

```

feita a automaização desses passos descritos criamos um pseudocódigo contento alguns detalhes sobre o script.

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. Através do script e com o auxílio do Vagrant vamos definir N números de máquinas virtuais ou contêiners para um cenário, o seguinte passo será instanciá-los. No próximo passo instalaremos todos os softwares que o NPB-MPI necessita para executar corretamente, após isso vamos instalar o NPB-MPI. Por fim definiremos os arquivos de configuração dos benchmarks, por exemplo: quais os benchmarks 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 esse benchmark, a saída de cada execução é armazenada em um arquivo individual para

esse *benchmark*. Ao final das 16 execuções o script escolhe o próximo *benchmark* da fila e repete os passos anteriores, 16 execuções e coleta dos resultados, até que todos os *benchmark* sejam executados. Ao término dessa 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 contento 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 *benchmarks* utilizados para a avaliação. A saída do script será semelhante à Figura 9 do trabalho de (MALISZEWSKI et al., 2018).

Os dados contidos na figura são de um experimento utilizando o *benchmark* IS do NPB-OMP que avaliou a performance dos KVM, LXC e comparou com o ambiente nativo, em ambientes distintos. O eixo X é o tempo da execução do *benchmark* em segundos e o eixo Y o número de *threads* utilizadas em cada ambiente.

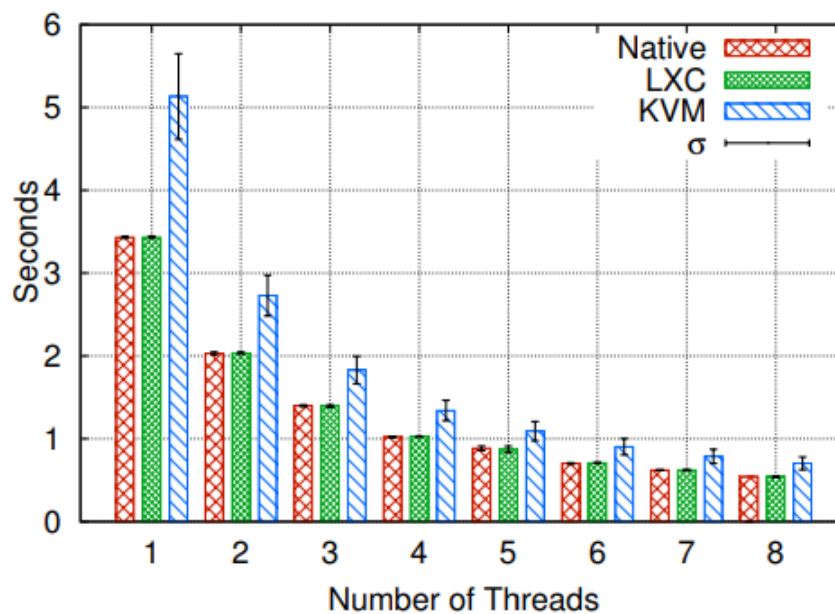


Figura 9 – Exemplo de saída do gráfico (MALISZEWSKI et al., 2018)

A diferença dos gráficos do trabalho proposto em relação à (MALISZEWSKI et al., 2018), além de serem gerados automaticamente, é que vamos levar em consideração o tempo de execução e os *flops*. Logo no gráfico tempo de execução, o eixo X será o tempo em segundo que o *benchmark* levará para executar e o eixo Y a quantidade de nós alocados em cada cenário. O outro gráfico conterá no eixo X os *flops*, e o eixo Y será semelhante ao gráfico do tempo de execução, contendo os nós definidos em cada um dos cenários onde foi realizados 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 coleta de dados. 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 dando como exemplo um dos gráficos criado em um dos trabalhos relacionados.

6 CONCLUSÃO E TRABALHOS FUTUROS

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. Pensando nisso, este trabalho tem 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. No final pretendemos obter os resultados para avaliar se houve discrepâncias entre esses dois virtualizadores e entender o porque que houve ou não.

Outra contribuição deste trabalho é desenvolver um script para automatizar este processo de avaliação, dessa forma tornaremos mais simples a reprodutibilidade deste trabalho, assim como os trabalhos futuros. Além disso, se outros autores não desejarem reproduzir todos os esses experimentos realizados, 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	-	-	-	-	-
Implementar o script	-	-	-	-	-	-	X	X	-	-	-
Realizar testes	-	-	-	-	-	-	-	X	X	-	-
Coletar os 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 3 vezes nas páginas 13, 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>. 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 35 e 39.
- 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 28.
- 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 38.
- 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 4 vezes nas páginas 36, 37, 38 e 43.
- 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, 35 e 38.
- 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 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 3 vezes nas páginas 35, 42 e 43.
- 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, 40, 41, 47, 48 e 55.
- 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.
- NASA. **NAS Parallel Benchmarks**NAS Parallel Benchmarks. 2018. [Online; accessed 5-June-2018]. Disponível em: <<https://www.nas.nasa.gov/publications/npb.html>>. Citado na página 22.
- ROSSO, J. P. Análise de desempenho de aplicações científicas em ambiente de nuvem privada. Universidade Federal do Pampa, 2015. Citado na página 50.
- 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 28.
- 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, 30 e 32.
- TAURION, C. **Cloud computing-computação em nuvem**. [S.l.]: Brasport, 2009. Citado 2 vezes nas páginas 32 e 33.
- 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 31 e 32.
- TRINITIS, C.; WEIDENDORFER, J. **Co-scheduling of HPC Applications**. [S.l.]: IOS Press, 2017. v. 28. Citado na página 29.
- 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 35, 36 e 47.