UNIVERSIDADE FEDERAL DO CEARÁ

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

PEDRO ROGER MAGALHÃES VASCONCELOS

AVALIAÇÃO DE DESEMPENHO DE VIRTUALIZAÇÕES OPENVZ E KVM EM NUVENS COMPUTACIONAIS USANDO HADOOP MAPREDUCE

SOBRAL

2015

PEDRO ROGER MAGALHÃES VASCONCELOS

AVALIAÇÃO DE DESEMPENHO DE VIRTUALIZAÇÕES OPENVZ E KVM EM NUVENS COMPUTACIONAIS USANDO HADOOP MAPREDUCE

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e de Computação da Universidade Federal do Ceará, como requisito parcial para obtenção do título de mestre em Engenharia Elétrica e de Computação. Área de Concentração: Sistemas de Informação

Orientadora: Prof^{a}. Dr^{a}. Gisele Azevedo de Araújo Freitas

SOBRAL

2015

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

V451a Vasconcelos, Pedro Roger Magalhães.
Avaliação de Desempenho de Virtualizações OpenVZ e KVM em Nuvens Computacionais usando Hadoop MapReduce / Pedro Roger Magalhães Vasconcelos. – 2015.
62 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Campus de Sobral, Programa de Pós-Graduação em Engenharia Elétrica e de Computação, Sobral, 2015.
Orientação: Profa. Dra. Gisele Azevedo de Araújo Freitas.

1. Computação em nuvem. 2. Virtualização. 3. Desempenho. I. Título.

CDD 621.3

PEDRO ROGER MAGALHÃES VASCONCELOS

AVALIAÇÃO DE DESEMPENHO DE VIRTUALIZAÇÕES OPENVZ E KVM EM NUVENS COMPUTACIONAIS USANDO HADOOP MAPREDUCE

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e de Computação da Universidade Federal do Ceará, como requisito parcial para obtenção do título de mestre em Engenharia Elétrica e de Computação. Área de Concentração: Sistemas de Informação

Aprovada em: __/__/____.

BANCA EXAMINADORA

Prof^a. Dr^a. Gisele Azevedo de Araújo Freitas (Orientadora)
Universidade Federal do Ceará (UFC)

Prof. Dr. Francisco Heron de Carvalho Júnior
Universidade Federal do Ceará (UFC)

Prof. Dr. José Cláudio do Nascimento
Universidade Federal do Ceará (UFC)

Prof. Dr. Danniel Cavalcante Lopes
Universidade Federal Rural do Semi-Árido (UFERSA)

À minha mãe-avó, Irene.

AGRADECIMENTOS

Agradeço a Deus que me acompanhou e guiou nas dificuldades.

À minha família, principalmente minha avó Irene, pelo incentivo e suporte na educação desde o ensino básico, e à memória de meus pais.

À minha noiva Suyanne pelo seu amor expresso no incentivo e compreensão quanto a minha ausência.

À minha orientadora Profa. Gisele Azevedo pela orientação, sugestões de conferências, ajuda nas publicações e apresentações de artigos.

Ao Diretor do Núcleo de Tecnologia da Informação da Universidade Estadual Vale do Acaraú, sr. Jorge Morais, pela liberação que me permitiu cursar o mestrado e pela utilização da infraestrutura computacional para execução de experimentos.

RESUMO

A Computação em nuvem permite acesso a um conjunto de recursos como máquinas virtuais, armazenamento e rede como serviços. Nesse contexto, a virtualização tem sido usada como uma plataforma para aplicações que fazem uso intensivo de recursos como, por exemplo, o Hadoop. As motivações desse uso são as seguintes características: consolidação de servidores, escalabilidade e melhor uso dos recursos físicos. OpenVZ e KVM são plataformas de virtualização amplamente utilizadas com modos de virtualização distintos: virtualização de sistemas operacionais e virtualização completa, respectivamente. Neste trabalho foi realizada uma série de experimentos para avaliar o desempenho de *clusters* Hadoop instalados em nuvens privadas OpenNebula utilizando OpenVZ e KVM como plataformas de virtualização. Os resultados mostraram que o *cluster* hospedado na nuvem com OpenVZ apresentou um melhor desempenho na maioria dos testes realizados com menor sobrecarga de virtualização de CPU e maior vazão de leitura de arquivos em disco e memória. O *cluster* hospedado na nuvem com KVM, por sua vez, apresentou taxas superiores apenas em testes específicos de escrita de dados em disco. O desempenho da virtualização de rede foi muito próximo do nativo em ambos os *clusters*.

Palavras-chave: Computação em nuvem. Virtualização. Desempenho.

ABSTRACT

Cloud computing provides access to a set of resources such as virtual machines, storage and network as services. In this context, virtualization has been used as a platform for resource-intensive applications, like Hadoop. The motivations of this use are the following features: server consolidation, scalability and better resources usage. OpenVZ and KVM are very popular and widely used virtualization platforms with distinct virtualization modes: container-based and full-virtualization, respectively. In this work, a set of benchmarks were conducted to measure the performance of Hadoop clusters deployed on OpenNebula private clouds with KVM and OpenVZ as virtualization platforms. The results show that the cluster hosted in the cloud with OpenVZ performs better than KVM in most tests, reaching a lower CPU virtualization overhead, higher disk I/O reading and memory throughput. The cluster hosted in the cloud using KVM, in turn, present better rates only in specifics tests of disk writing. The performance of network virtualization is very close to native in both clusters.

Keywords: Cloud computing. Virtualization. Performance.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

CPU	*Central Processor Unit*/Unidade Central de Processamento
API	*Application Programming Interface*/Interface de Programação de Aplicação
AWS	*Amazon Web Services*
CAD	Computação de Alto Desempenho
CC	*Cluster Controller*
CLC	*Cloud Controller*
E/S	Entrada/Saída
EBS	*Elastic Block Store*
EC2	*Elastic Compute Cloud*
Eucalyptus	*Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems*
GFLOPS	*Giga FLoating-point Operations Per Second*/Bilhões de Operações de ponto-flutuante por segundo
HDFS	*Hadoop Distributed File System*/Sistema de Arquivos Distribuído Haddop
HVM	*Hardware Virtual Machine*
IaaS	*Infraestructure as a Service*/Infraestrutura como Serviço
IOMMU	*Input/Output Memory Management Unit*/Unidade de Gerenciamento de E/S de Memória
iSCSI	*Internet Small Computer System Interface*
KVM	*Kernel-based Virtual Machine*
LXC	*Linux Containers*
MMV	Monitor de Máquinas Virtuais
MV	Máquina Virtual
NASA	*National Aeronautics and Space Administration*
NC	*Node Controller*
NFS	*Network File System*
NIST	*National Institute of Science and Technology*
OCCI	*Open Cloud Computing Interface*
Paas	*Platform as a Service*/Plataforma como Serviço
PCI	*Peripheral Component Interconnect*
PV	Paravirtualização

RPC	*Remote Procedure Call*/Chamada de Procedimento Remoto
S3	*Simple Storage Service*
SaaS	*Software as a Service*/Software como Serviço
SAN	*Storage Area Network*
SAS	*Serial Attached Scsi*
SC	*Storage Controller*
SO	Sistema Operacional
SOS	*Scalable Object Storage*
SSH	*Secure Shell*
XML	*eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

A computação em nuvem emergiu como um novo paradigma de computação distribuída em larga escala que possibilitou transferir o poder de computação e os dados das estações de trabalho e dos dispositivos móveis para os grandes *data centers* (ARMBRUST *et al.*, 2009). É um modelo que permite acesso ubíquo, prático e sob demanda por rede a um conjunto compartilhado de recursos computacionais (como redes, servidores, armazenamento e serviços) que podem ser rapidamente provisionados e liberados com um esforço mínimo de gerenciamento ou interação com o provedor de serviços.

Uma nuvem computacional típica é hospedada em uma infraestrutura com uma grande quantidade de computadores. Tais máquinas provêm os recursos de *hardware*. O provedor da nuvem (a organização que hospeda os servidores) oferece uma interface aos usuários para pagar por uma certa quantidade de processamento, armazenamento ou computadores em um modelo de negócios. Estes recursos podem ser incrementados ou diminuídos sob demanda, assim os usuários precisam focar somente nos seus próprios recursos alocados, enquanto o provedor cuida da manutenção, segurança e rede de dados.

Atualmente os *data centers* possuem servidores multiprocessados com alta capacidade de armazenamento e grande largura de banda de rede. Porém, a maioria do tempo estes servidores permanecem subutilizados, pois, o principal motivo de manter essa alta capacidade de processamento é atender a períodos de picos de carga. Desse modo, a virtualização representa um papel importante no melhor aproveitamento da capacidade de computação através da consolidação de servidores e isolamento. Utilizando a virtualização é possível multiplexar os recursos de *hardware* e permitir que usuários executem diferentes sistemas operacionais sob os mesmos recursos físicos.

Existe uma grande variedade de plataformas de virtualização que diferem umas da outras pela abordagem de virtualização, licença de uso (*software* livre ou proprietário), necessidade de assistência via *hardware*, entre outros fatores. Essas plataformas também diferem com relação ao desempenho e funcionalidades. Este trabalho tem por finalidade promover um estudo sobre o desempenho de aplicações executadas em ambientes virtualizados e em nuvem, levando em consideração algumas das principais plataformas de virtualização e computação em nuvem de código aberto.

Esta dissertação está organizada na forma descrita a seguir. O Capítulo **??** aborda os fundamentos de Computação em Nuvem, seus modelos de serviços e implantação e a plataforma

de computação em nuvem utilizada. O Capítulo **??** descreve os conceitos, os vários tipos de virtualização e as plataformas virtuais utilizadas nos experimentos realizados. O Capítulo **??** aborda trabalhos relacionados com o tema. O Capítulo **??** descreve a solução proposta e os resultados alcançados. Finalmente, o último capítulo descreve as conclusões e perspectivas de trabalhos futuros.

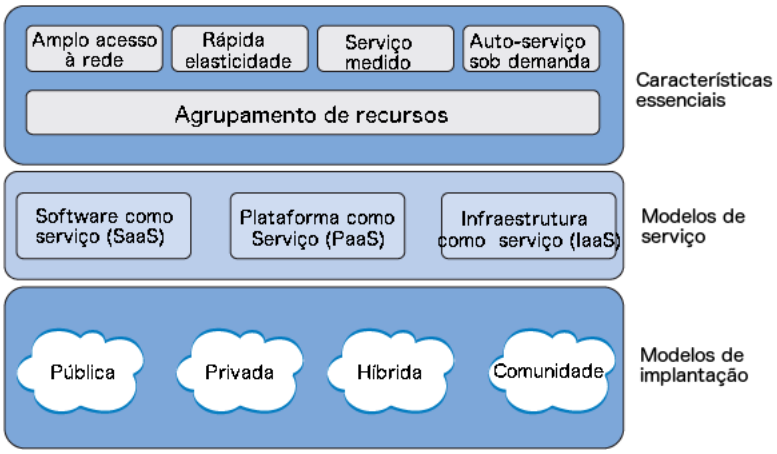
2 COMPUTAÇÃO EM NUVEM

A computação em nuvem é um tipo de computação distribuída que permite um acesso ubíquo via rede a um conjunto compartilhado de recursos computacionais como redes, servidores, armazenamento, aplicações e serviços, que podem ser rapidamente provisionados e liberados com o mínimo esforço de gerenciamento ou interação com o provedor de serviços (MELL; GRANCE, 2011). O *hardware* e a instalação de *softwares* básicos são fornecidos pelo provedor da nuvem como um serviço, permitindo aos assinantes instalarem *softwares* e gerenciarem recursos com maior facilidade do que seria em uma infraestrutura local. O termo nuvem refere-se a representação típica da Internet em diagramas: uma nuvem.

A virtualização é fundamental na computação em nuvem, pois através dela os provedores de serviço podem maximizar o poder de processamento de suas infraestruturas e obter elasticidade. Elasticidade é a possibilidade dos usuários escalarem as instâncias das quais necessitam. Ela também ajuda a fornecer outras duas características essenciais de uma nuvem: suporte multi-inquilino, o compartilhamento de recursos logicamente isolados entre diferente consumidores; e escalabilidade massiva, a capacidade de ter uma enorme quantidade de sistemas de processamento e áreas de armazenamento (MATHER *et al.*, 2009).

O *National Institute of Science and Technology* (NIST), entidade federal não-regulatória vinculada ao Departamento de Comércio do Governo dos Estados Unidos, propõe um modelo de referência para a computação em nuvem que considera a existência de três distintas dimensões: (a) características essenciais, (b) modelos de serviços e (c) modelos de implantação (MELL; GRANCE, 2011). Este modelo é ilustrado na Figura 1 e definido a seguir:

Figura 1 – Definições do NIST para computação em nuvem



Fonte: (IBM, 2015).

2.1 Características

A computação em nuvem possui as seguintes características principais:

- **Auto-serviço sob demanda**: Os consumidores podem, unilateralmente, provisionar capacidade computacional, à medida das suas necessidades e automaticamente, sem que seja requerida interação humana com os provedores dos serviços.
- **Faturamento e medição por uso**: Como o usuário tem a opção de requisitar e utilizar somente a quantidade de recursos e serviços que ele julgar necessário, os serviços devem ser precificados com uma base em um uso de baixa duração, como por exemplo, em medido em horas de uso.
- **Acesso amplo à rede**: Os recursos estão disponíveis na rede e são acessados por meio de mecanismos padronizados que possibilitam o uso através de diferentes plataformas de hardware.
- ***Pool* de recursos**: Os recursos computacionais dos provedores são agrupados, de modo a servir a múltiplos consumidores simultaneamente, com distintos recursos físicos e virtuais alocados e realocados dinamicamente de acordo com a demanda; há um certo grau de independência quanto à localização, fazendo com que os consumidores não decidam sobre os locais exatos a partir dos quais acessarão os recursos, mas possam especificar essa localização em alto nível (por exemplo, país, estado ou até mesmo *data center*).
- **Rápida elasticidade**: Os recursos podem ser provisionados de forma elástica, em alguns casos automaticamente, para escalarem baseados na demanda.

2.2 Modelos de Serviço

A computação em nuvem é baseada na oferta de serviços, podendo ser classificada em um dos três modelos propostos pelo NIST. Esta classificação é conhecida como modelo SPI (SaaS, Paas, IaaS), formado por *Software as a Service*/Software como Serviço (SaaS), *Platform as a Service*/Plataforma como Serviço (Paas) e *Infraestructure as a Service*/Infraestrutura como Serviço (IaaS) (OWOPETU, 2013):

2.2.1 Software como Serviço

É um modelo de entrega de software no qual as empresas clientes pagam, não pela propriedade do *software*, mas, pelo uso do mesmo e as companhias fornecedoras provêm

manutenção e suporte técnico aos seus clientes (MELO *et al.*, 2007). Os serviços representam aplicações disponibilizadas pela Internet, normalmente interpretadas utilizando-se um navegador *web*. O *hardware* e *software* utilizados são totalmente abstraídos dos usuários. Estes usuários obtêm acesso às informações e funcionalidades através do navegador *web*. O usuário não gerencia ou controla a infraestrutura de nuvem adjacente como redes, servidores, sistemas operacionais, armazenamento ou mesmo características individuais das aplicações, exceto de configurações específicas (MELL; GRANCE, 2011). Entre as vantagens da utilização do modelo SaaS estão a não obrigatoriedade de uma estrutura complexa de tecnologia por parte dos usuários, pois, a principal carga de trabalho é executada na nuvem, e redução de despesas de treinamento de manutenção e instalação de *software*.

Como exemplos deste modelo de serviço podemos destacar os aplicativos Google Drive e Netflix, e os serviços de gestão de relacionamento com o cliente da Salesforce.

2.2.2 Plataforma como Serviço

Neste modelo o provedor de serviços fornece linguagens de programação, bibliotecas, serviços e ambientes de desenvolvimento para as aplicações, auxiliando a implementação e desenvolvimento de *softwares*. Os serviços disponibilizados possibilitam executar aplicativos desenvolvidos pelos consumidores ou por estes adquiridos de terceiros; os consumidores não gerenciam essa infraestrutura, sejam redes, servidores, sistemas operacionais ou áreas de armazenagem, mas, apenas controlam os aplicativos e podem configurar o ambiente.

Estes serviços são focados na implantação de aplicações, serviços on-line ou pilhas de soluções permitindo ao desenvolvedor gerenciar o *hardware* e o *software* necessários. Este serviço inclui todo o ciclo de vida da implantação da aplicação/serviço como projeto, implementação, teste, implantação e integração com base de dados (FOLCH, 2011). O Microsoft Azure, o Google App Engine e a plataforma Salesforce1 são exemplos de PaaS.

2.2.3 Infraestrutura como Serviço

Os serviços disponibilizados são o provisionamento e uso de recursos de infraestrutura, como capacidade de processamento, áreas de armazenagem, redes e outros recursos computacionais básicos, de modo a possibilitar a execução de aplicativos e sistemas operacionais; os consumidores não gerenciam essa infraestrutura, mas, controlam os sistemas operacionais, áreas de armazenamento, aplicativos e, em alguns casos, podem até mesmo configurar com-

ponentes (por exemplo, *firewalls*) (FOLCH, 2011). O Amazon *Elastic Compute Cloud* (EC2), Eucalyptus, OpenStack e OpenNebula são exemplos de IaaS.

2.3 Modelos de implantação

Quanto ao acesso e à disponibilidade, há diferentes tipos de modelos de implantação para os ambientes de computação em nuvem. A restrição ou abertura de acesso depende do processo de negócios, do tipo de informação e do nível de visão desejado (COUTINHO, 2014):

— Modelo de implantação público

Uma nuvem pública compreende aquelas nas quais o provedor e os usuários em potencial não pertencem à mesma organização. Tais nuvens são acessíveis ao público e geralmente oferecem um portal de gerenciamento *web* no qual os usuários podem contratar os recursos desejados. Os serviços são contabilizados e cobrados de forma baseada nos recursos atualmente usados em determinado período de tempo. Nessa infraestrutura há dados de vários usuários, porém, obviamente, eles não podem acessar as informações que não lhes pertence. As nuvens públicas são extremamente populares e mais adequadas a situações como a implantação inicial de um novo *website* por conta de sua excepcional escalabilidade, podendo abranger desde as mais simples configurações até as mais complexas (ARMBRUST *et al.*, 2009). Exemplos de nuvens públicas são Amazon EC2, Windows Azure, Google App Engine e IBM Blue Cloud.

— Modelo de implantação privado

Esse modelo consiste na hospedagem de aplicações privadas, armazenamento ou computação na mesma companhia, do mesmo modo que nas nuvens disponibilizadas pela Internet, porém, apenas para uso privado.

Nuvens privadas são geralmente *data centers* usados em redes privadas, portanto, restringe o acesso não autorizado aos dados utilizados pela instituição (GIACOMO; BRUNZEL, 2010).

As nuvens privadas são ideais quando as companhias possuem dados ou serviços que requerem monitoramento e controle restritos. Em tais situações há, frequentemente, uma necessidade de alterações e gerenciamento delicados na infraestrutura. Nuvens privadas geralmente são a melhor escolha quando uma companhia deve estar em conformidade com acordos

restritos de segurança e privacidade. OpenStack, Eucalyptus e OpenNebula são exemplos de plataformas para implantação de nuvens privadas.

— Modelo de implantação híbrido

As nuvens híbridas são ideais para as companhias que podem se beneficiar de ambas as nuvens públicas e híbridas, combinando as vantagens desses dois modelos de implantação em um modo bastante atrativo. Uma nuvem híbrida pode ser considerada quando não for viável manter todos os dados em uma nuvem privada. Devido aos altos custos de operação das nuvens privadas, mover alguns dados e aplicações para nuvens públicas promove economia e geralmente é o motivo de se considerar as nuvens híbridas (MELL; GRANCE, 2011).

Como cenário de exemplo de uma nuvem híbrida podemos imaginar uma empresa que possui uma nuvem privada local para hospedar cargas de trabalho sensíveis, mas, utiliza uma nuvem pública de terceiros como a Amazon EC2 para hospedar recursos menos críticos como ambientes de teste e desenvolvimento.

— Modelo de implantação comunidade

Nas nuvens em comunidade a infraestrutura é provisionada para o uso exclusivo de uma comunidade de clientes de organizações que possuem interesses em comum como missões, requisitos de segurança e considerações de conformidade. Nesse modelo de implantação a nuvem pode ser gerenciada, operada e de propriedade de mais de uma organização na comunidade, terceiros ou uma combinação destes (MELL; GRANCE, 2011).

2.4 Plataformas de computação em nuvem privada

Um sistema de computação em nuvem não necessariamente precisa ser contratado. Empresas ou usuários individuais podem utilizar a sua própria infraestrutura e implantar uma nuvem privada. Entre as plataformas de computação em nuvem privada destacam-se:

— OpenStack

O OpenStack é um projeto de código aberto para a implantação de nuvens que teve seu desenvolvimento iniciado em julho de 2010 sendo a *National Aeronautics and Space Administration* (NASA), agência espacial americana, e Rackspace os seus principais desenvolve-

dores. O projeto inclui três produtos: Nova (software similar ao Amazon EC2), Swift (sistema de armazenamento escalável similar ao Amazon *Simple Storage Service* (S3)), e Glance (uma *Application Programming Interface*/Interface de Programação de Aplicação (API) que provê serviços de descoberta, registro e entrega de imagens de discos virtuais). Nova possui suporte aos hipervisores *Kernel-based Virtual Machine* (KVM) e Xen, é responsável pela parte de IaaS e gerenciamento de todas as atividades que são necessárias ao ciclo de vida de instâncias dentro do OpenStack.

OpenStack é uma plataforma popular entre uma vasta comunidade de especialistas e é suportada e utilizada por companhias como Cisco, Dell, NASA, Intel, AMD, Citrix, Rackspace e RightScale. A arquitetura OpenStack é distribuída, possui uma instalação relativamente complicada e um forte sistema de segurança baseado em *tokens*. Os seus serviços são disponibilizados através de APIs compatíveis com Amazon EC2 e S3, assim ferramentas escritas para a *Amazon Web Services* (AWS) podem também ser utilizadas com OpenStack.

A comunidade OpenStack realiza semestralmente em diferentes localizações a conferência OpenStack Summit para desenvolvedores, usuários e administradores de sistemas OpenStack. As últimas edições do evento alcançaram mais de 5.000 participantes.

— Nimbus

Nimbus é conjunto de ferramentas de código-aberto focado em prover capacidades de IaaS para a comunidade científica. Permite aos provedores de serviço construir nuvens de IaaS privadas ou em comunidade. O *Nimbus Workspace Service* permite aos usuários provisionarem recursos e lançarem instâncias de máquinas virtuais em tais recursos. Uma ferramenta complementar, chamada Cumulus, provê a implementação de um sistema de armazenamento em nuvem baseado em cotas. Cumulus é projetado para possuir escalabilidade e permitir que os provedores o configurem em vários dispositivos de armazenamento.

Nimbus também oferece ferramentas de escalabilidade para permitir aos usuários automaticamente escalarem seus serviços entre os nós de processamento do provedor. Essas ferramentas são chamadas de *sky computing tools* por seres operadas em ambientes multi-nuvem combinando infraestruturas públicas e privadas. O *Nimbus Workspace Service* pode ser configurado para suportar diferentes implementações de virtualização (Xen ou KVM), opções de gerenciadores de recursos, interfaces (incluindo compatibilidade com Amazon EC2) e outras opções.

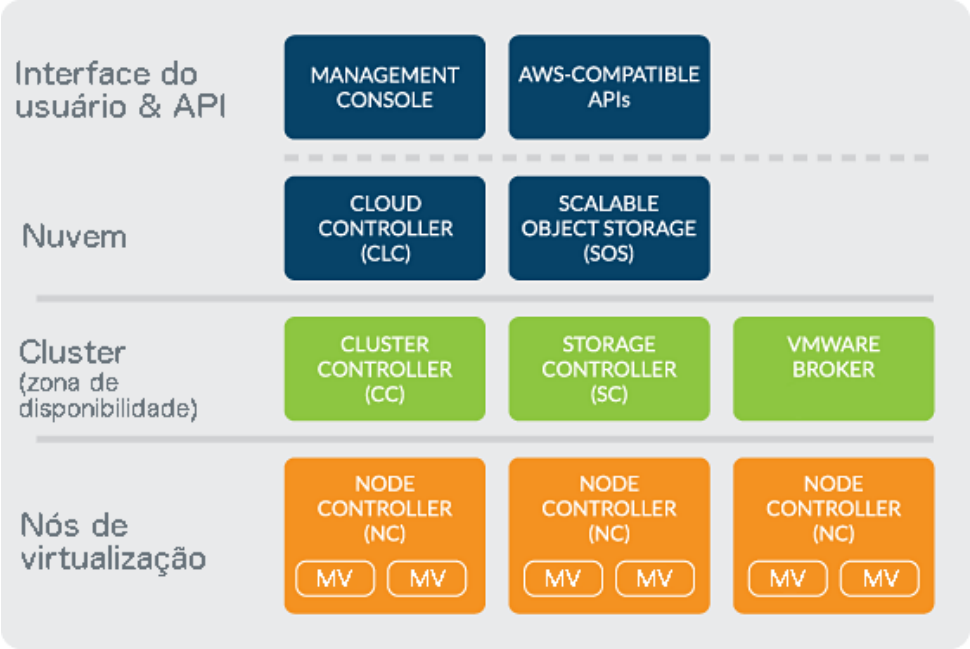
— Eucalyptus

O *Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems* (Eucalyptus) é um *framework* de código aberto para a implantação de nuvens privadas de IaaS. Ele permite que os seus usuários executem e controlem máquinas virtuais utilizando uma dada infraestrutura física. Eucalyptus provê uma API baseada nos serviços da AWS como o EC2 e S3, permitindo compatibilidade com esses importantes serviços (FOLCH, 2011).

Eucalyptus possui uma arquitetura modular e hierárquica na qual cada módulo é implementado como o serviço *web* a fim de alcançar mais compatibilidade e segurança. Os seis módulos que compõem a plataforma Eucalyptus são: *Cloud Controller* (CLC), *Scalable Object Storage* (SOS), *Cluster Controller* (CC), *Storage Controller* (SC), *Node Controller* (NC) e um módulo opcional chamado *VMware Broker* (VB) (Figura 2) (EUCALYPTUS, 2014).

- **Cloud Controler:** O *Cloud Controler* é o ponto de entrada da nuvem para os seus administradores, desenvolvedores e usuários finais. O CLC consulta informações de outros componentes, realiza decisões de agendamento e realiza requisições ao *Cluster Controler*. Como uma interface para a plataforma de gerenciamento, o CLC é responsável por expor e gerenciar os recursos virtualizados subjacentes (servidores, rede e armazenamento). Apenas um CLC pode existir por nuvem.

Figura 2 – Componentes Eucalyptus



Fonte: (EUCALYPTUS, 2014).

- **Scalable Object Storage:** É um serviço Eucalyptus similar ao Amazon S3. O *Scalable Object Storage* é um serviço que permite aos administradores da infraestrutura terem a flexibilidade de implementar armazenamento escalável para nuvens de larga escala. Eucalyptus também provê um sistema de armazenamento básico, chamado Walrus, mais adequado para avaliações e nuvens de pequena escala.
- **Cluster Controller:** Um *cluster* é equivalente a uma zona de disponibilidade AWS, e um nuvem Eucalyptus pode possuir vários clusters. O *Cluster Controller* age como um *frontend* para um *cluster* dentro de uma nuvem Eucalyptus e se comunica com o *Storage Controller* e o *Node Controller*. O CC gerencia a execução de instâncias e acordos de níveis de serviço por cluster.
- **Storage Controller:** É um componente escrito em java e é equivalente ao Amazon *Elastic Block Store* (EBS). O *Storage Controller* realiza comunicações com o *Cluster Controller* e *Node Controller* e gerencia volumes e *snapshots* de dispositivos de bloco para as instâncias dentro de um *cluster*. O SC realiza interfaces com vários sistemas de armazenamento como *Network File System* (NFS), *Internet Small Computer System Interface* (iSCSI) e *Storage Area Network* (SAN).
- **VMware Broker:** É um módulo opcional que permite ao Eucalyptus lançar máquinas virtuais em uma infraestrutura VMware.
- **Node Controller:** O *Node Controller* é executado nas máquinas que hospedam máquinas virtuais. O NC controla atividades das máquinas virtuais como execução, inspeção e término de instâncias.

— OpenNebula

O OpenNebula é uma plataforma para a construção e gerenciamento de nuvens corporativas e *data centers* virtualizados. Tem sido desenvolvido com o código aberto desde seu início, com o fim de garantir interoperabilidade e compatibilidade entre todos os componentes possíveis. Consequentemente, OpenNebula não utiliza um único hipervisor e não possui requisitos específicos para ambientes (VASCONCELOS; FREITAS, 2014a).

Adicionalmente, OpenNebula difere de outras plataformas de código aberto devido ao uso de *Remote Procedure Call*/Chamada de Procedimento Remoto (RPC) a objetos *eXtensible Markup Language* (XML), que podem ser utilizadas para acessar a API diretamente. OpenNebula também utiliza a *Open Cloud Computing Interface* (OCCI) e suporte a Amazon

EC2 a fim de expandir os seus recursos e formar uma nuvem híbrida. A Figura 3 ilustra essa nuvem (SEMPOLINSKI; THAIN,).

OpenNebula apresenta-se ao usuário através de uma interface web, chamada *Sunstone* ou da interface de clinha de comando e utiliza conexões seguras *Secure Shell* (SSH) para realizar a comunicação com os outros componentes da nuvem.

Diferentemente de outras alternativas livres, OpenNebula não possui requisitos quanto a um hipervisor em particular, a comunidade OpenNebula é bastante ativa e desenvolve frequentemente *drivers* para a utilização de novos e recursos e hipervisores, como o OpenVZ.

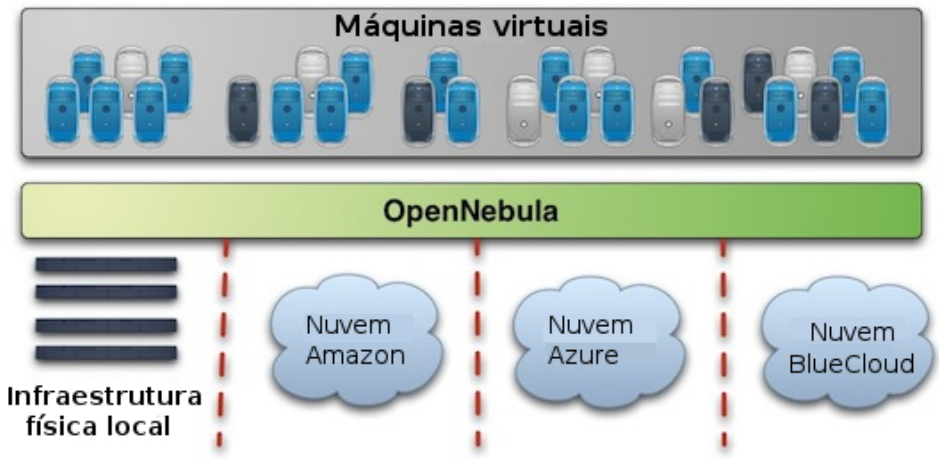
Dessa forma, uma instância de gerenciamento OpenNebula pode ser utilizada para combinar infraestrutura local com várias nuvens remotas, construindo um ambiente de hospedagem altamente escalável.

A escolha do OpenNebula como plataforma de computação em nuvem para este trabalho foi motivada pelo suporte ao hipervisor OpenVZ e pela já existente infraestrutura OpenNebula instalada no ambiente de execução. As funcionalidades de *spanshot* e *templates* de máquinas virtuais do OpenNebula facilitaram a instalação e lançamento de novas instâncias necessárias à execução dos testes.

2.5 Aplicações em Computação em Nuvem

A computação em nuvem pode ser plataforma para variados tipos de aplicações, a sua capacidade de rápida elasticidade proporciona um ambiente propício para a execução de

Figura 3 – Nuvem híbrida OpenNebula



Fonte: Fonte: (C12LABS, 2015).

serviços para cargas de trabalho que exigem muito poder de computação e grande infraestrutura. O balanceamento de carga é um conceito fundamental nessa ambiente pois é um mecanismo que distribui o excesso de carga de trabalho uniformemente entre todos os nós. É utilizado para alcançar uma alta satisfação de usuários e melhor taxa de utilização dos recursos, certificando-se que nenhum nó esteja sobrecarregado e contribuindo para a melhora do desempenho global do sistema (VASCONCELOS; FREITAS, 2014c). Entre essas aplicações que requerem uma grande quantidade de nós de computação encontra-se o Hadoop MapReduce.

2.5.1 Modelo de programação MapReduce

O MapReduce é um modelo de programação para o processamento de grande quantidades de dados utilizando computação distribuída. Foi desenvolvido e patenteado pelo Google, também é conhecidamente utilizado por outras grandes companhias como Yahoo!, Facebook, MySpace e Twitter. No Facebook, logs de eventos do seu *website* são importados em um *cluster* MapReduce de 600 nós, onde são usados para uma variedade de aplicações incluindo inteligência de negócios, detecção de *spam* e otimização de publicidades. Esse *datawarehouse* armazena 2PB de dados e cresce aproximadamente 15TB por dia (VASCONCELOS; FREITAS, 2013).

As tarefas, ou *jobs*, MapReduce são inerentemente paralelos, provendo assim, análise de dados em larga escala para qualquer pessoa que tenha máquinas suficientes à disposição (WHITE, 2012).

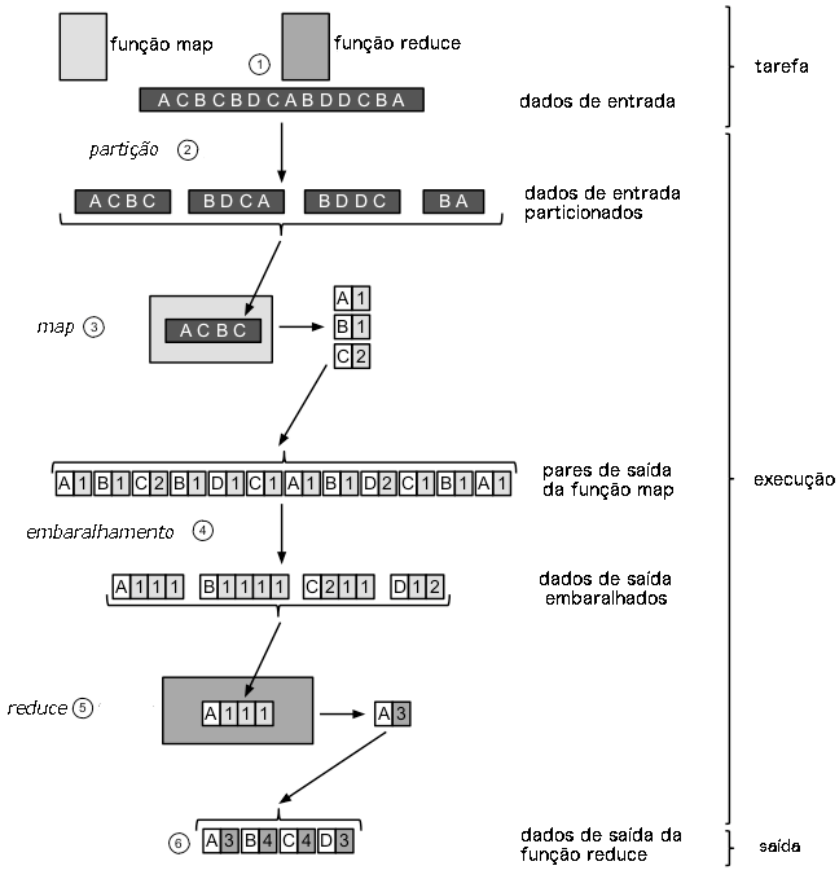
Um programa MapReduce é uma combinação de funções *Map* e *Reduce*, a função *Map* é aplicada aos dados de entrada e a função *Reduce* é aplicada na saída da função *Map* . A Figura 4 ilustra uma visão global do funcionamento do MapReduce que é explicado a seguir (RUITER, 2012):

1. Um *job* MapReduce consiste de uma função map, uma função *Reduce* e os dados de entrada (em um sistema de arquivos distribuído).
2. Primeiro, os dados de entrada são particionados em blocos de dados menores.
3. Então, para cada bloco de dados uma tarefa *Map* é executada. O resultado de cada tarefa *Map* é uma coleção de pares chave-valor.
4. A saída de todas as tarefas *Map* é embaralhada, de modo que, para cada par distinto na saída da tarefa map, uma coleção é criada contendo todos os valores correspondentes da saída da tarefa map.

5. Então, para cada coleção de chaves resultantes da fase de embaralhamento, uma tarefa *Reduce* é executada. O resultado é um único par chave-valor.
6. A coleção de todos os pares chave-valor resultantes da fase *Reduce* é a saída do *job* MapReduce.

O modelo é inteiramente construído baseado em pares chave-valor, estes pares são os únicos itens que são comunicados entre as diferentes partes da estrutura. As chaves e os valores podem ser implementados pelo usuário, mas eles são obrigados a serem serializados uma vez que serão transmitidos através da rede. Chaves e valores podem variar de tipos primitivos simples para grandes tipos de dados. Ao se processar uma tarefa MapReduce, esta deve ser capaz de ser dividida em *n* partes, em que *n* é, pelo menos, a quantidade de nós de processamento MapReduce no *cluster*. As diferentes fases do processamento de uma tarefa MapReduce trabalham em paralelo tanto quanto possível. As fases de embaralhamento e redução podem ser iniciadas assim que uma tarefa *Map* tenha sido concluída (NILSSON, 2011).

Figura 4 – Visão global do MapReduce



Fonte: Fonte: (RUITER, 2012).

— Fase Map

A função *Map* tem como entrada um único par de chave-valor, e produz como saída um número qualquer de novos pares de chave-valor. É fundamental que a operação *Map* opere um par de cada vez. Isto permite a fácil paralelização, pois, diferentes entradas podem ser processadas por máquinas distintas (KLOSS, 2013).

— Fase Reduce

A função *Reduce* converte todos os valores associados com uma única chave, e gera um conjunto de pares chave-valor com a mesma chave. Uma vez que o redutor tem acesso a todos os valores com a mesma chave, pode realizar cálculos sequenciais com estes valores. Na etapa reduce, o paralelismo é explorado ao observar que chaves diferentes podem ser executadas simultaneamente (KLOSS, 2013).

2.5.2 Framework Hadoop MapReduce

O Hadoop MapReduce é um *framework* de programação distribuída e um ambiente de execução para programas MapReduce desenvolvido pela Fundação Apache, que contém diferentes sistemas destinados a armazenamento de arquivos, análise e processamento de grandes quantidades de dados. O software Hadoop é escrito em Java e possui API para integração com outras linguagens de programação.

Existem dois tipos de nós que controlam o processo de execução de um *job*: *JobTracker* e *TaskTrackers* (XAVIER *et al.*, 2014).

— Hadoop Distributed File System (HDFS)

O *Hadoop Distributed File System*/Sistema de Arquivos Distribuído Haddop (HDFS) é um sistema de arquivos distribuído designado a armazenar arquivos muito grandes e a prover alto desempenho para acesso a esses arquivos. Todos os dados no Hadoop são armazenados como arquivos HDFS, compostos de blocos de dados de tamanho fixo (64MB cada, por padrão) e distribuídos entre múltiplos nós. Há dois tipos de nós em um cluster HDFS: um *NameNode* e vários *DataNodes*. O *NameNode* mantém os metadados do sistema de arquivos, que inclui informação sobre os arquivos, a árvore de diretórios e a localização em que cada bloco de dados está fisicamente armazenado. Os *DataNodes* armazenam os blocos de dados em si. Cada vez que

um cliente necessita ler um arquivo do HDFS, ele primeiro contacta o *NameNode* objetivando determinar o *DataNode* no qual todos os blocos para aquele arquivo estão localizados. Então, o cliente inicia a leitura dos blocos de dados diretamente dos *DataNodes* descobertos (XAVIER *et al.*, 2014).

2.6 Conclusão

Este capítulo aborda a definição formal de computação em nuvem, suas características essenciais, modelos de serviço e implantação, bem como exemplos de softwares e companhias que utilizam tais modelos. Foram apresentadas algumas das principais plataformas de computação em nuvem para infraestruturas privadas como OpenStack, Nimbus, Eucalyptus e OpenNebula, bem como a justificativa da adoção do OpenNebula nos experimentos realizados neste trabalho. O modelo de programação MapReduce foi descrito como uma aplicação para análise de grande volumes de dados que requer uma grande quantidade de nós de processamento que podem ser facilmente instanciados dentro de uma nuvem.

3 VIRTUALIZAÇÃO

Virtualização é geralmente referenciada como uma camada de *software* que abstrai as características físicas do *hardware*, fornecendo recursos virtualizados para as aplicações de alto nível. A virtualização foi desenvolvida principalmente para obter melhor uso do *hardware* disponível. Além disso, cada ambiente virtualizado pode ser independente, podendo ter seus próprios aplicativos, serviços e sistema operacional, ou seja, atua como se estivesse instalado isoladamente em uma máquina física (WOTTRICH *et al.*, 2012). O conceito de virtualização permite uma visão abstrata dos recursos físicos que incluem servidores, armazenamento, redes e *softwares*. A ideia básica é agrupar recursos físicos e gerenciá-los como um todo, as requisições podem então ser servidas a partir destes recursos.

As tecnologias de virtualização são baseadas no conceito de sistemas operacionais de tempo compartilhado e memória virtual. Nos primórdios da computação, a memória RAM era cara e uma solução que permitisse executar um programa maior que a memória disponível era extremamente necessária. A solução foi a criação dos conceitos de memória virtual e técnicas de paginação, que tornaram mais fácil ter grandes programas em memória e permitiram multiprogramação. O conceito de sistemas operacionais de tempo compartilhado permitiu a execução de várias tarefas de múltiplos usuários ao mesmo tempo, gerando uma melhor utilização do *hardware* (MAGNUS; OPSAHL, 2013).

Através da utilização de tecnologias de virtualização consegue-se uma série de benefícios (BAUN *et al.*, 2011):

- **Utilização de recursos:** É possível automatizar o gerenciamento de recursos. Máquinas virtuais podem ser criadas e configuradas automaticamente quando requeridas.
- **Consolidação:** Diferentes tipos de aplicações podem ser consolidadas para serem executadas em um menor número de componentes físicos. A consolidação tende a aumentar a eficiência, o que leva a uma redução de custos de operação.
- **Consumo de energia:** A quantidade de energia requerida para abastecer grandes *data centers* é muito elevada e o custo da energia necessária para operar um servidor ao longo de sua vida útil é, frequentemente, superior ao seu preço de compra. A consolidação reduz o número de componentes físicos, e consequentemente, as despesas com fornecimento de energia elétrica são reduzidas.
- **Menor requerimento de espaço:** Cada metro quadrado de um *data center* é escasso e caro. Com a consolidação, um desempenho similar pode ser obtido em um espaço menor

e a expansão de um *data center* já existente pode, possivelmente, ser evitada.

- **Planejamento de emergência:** É possível mover máquinas virtuais de um recurso físico para outro, o que garante melhor disponibilidade dos serviços e torna mais fácil cumprir acordos de nível de serviço, além de diminuir as janelas de manutenção de *hardware*.

Uma vez que grandes provedores de serviço, notadamente serviços em nuvem, tendem a construir grandes *data centers*, a virtualização não só leva a uma vantagem de tamanho, mas também a uma situação economicamente favorável. Do ponto de vista do cliente a virtualização resulta em várias vantagens (BAUN *et al.*, 2011):

- **Comportamento dinâmico:** Requisições podem ser atendidas em tempo e sem atrasos. Em casos de gargalos de recursos, uma máquina virtual pode ser instanciada e prover recursos adicionais.
- **Disponibilidade:** Os serviços são disponibilizados dia e noite sem interrupção. Em caso de atualizações de tecnologias, é possível realizar migrações em tempo real porque as máquinas virtuais podem facilmente ser movidas para um sistema atualizado.
- **Isolamento:** A camada de virtualização isola cada máquina virtual das outras e da infraestrutura física. Desse modo, os sistemas virtualizados possuem capacidade multi-inquilino e, pela utilização do conceito de papéis, é possível delegar funcionalidades de gerenciamento aos clientes.

Virtualização envolve uma variedade de termos e conceitos, que diferem em relação a suas implementações. Dentre os principais termos temos:

- **Máquina Virtual (MV):** É uma abstração de uma máquina real em execução, que 'pensa' estar sendo executada sob *hardware* real, quando na verdade está funcionando em uma camada de abstração que fica entre a camada de virtualização e o *hardware*.
- **Monitor de Máquinas Virtuais (MMV):** O MMV, também chamado de hipervisor, é a camada de *software* que fica entre a MV e o *hardware*. Existem duas classificações de MMVs que diferenciam-se entre:
  - Nativo: Localiza-se diretamente no topo do *hardware*. Utilizada principalmente na virtualização tradicional de sistemas da década de 60 e na plataforma de virtualização Xen (XEN,).
  - Hospedado: Localiza-se no topo de um sistema operacional (Sistema Operacional (SO)) existente. Mais proeminente nos sistemas de virtualização modernos.

  Um MMV possui três características (MAGNUS; OPSAHL, 2013):

- **Equivalência**: Qualquer programa em execução sob o MMV deve possuir comportamento idêntico ao executado sob a máquina física. Porém, não necessariamente idêntico quando existem outras MVs presentes no MMV que possam ocasionar conflitos de agendamento entre MVs.
- **Eficiência**: O MMV deve ser capaz de executar um subconjunto de instruções diretamente no processador real, sem nenhuma intervenção de *software* pela MMV.
- **Controle de recursos**: O MMV deve possuir total controle dos recursos do sistema, o que significa um processo não pode acessar recursos que não foram explicitamente alocados a ele.

Um MMV é usualmente classificado como Tipo I, Tipo II ou Híbrido (GOLDBERG, 1973) descritos a seguir:

— MMV Tipo I

Também é conhecido como hipervisor nativo ou *bare metal*. É executado diretamente sob a máquina física, significando que o MMV possui comunicação direta com o *hardware* subjacente, como mostrado na figura 5 (a). Nesse caso, o MMV é um pequeno código cuja responsabilidade é realizar o agendamento e alocação dos recursos do sistema entre as MVs já que não há um sistema operacional abaixo dele. São exemplos de MMVs Tipo I o VMware ESX (VMWARE,) e Xen (GALVIN, 2009).

— MMV Tipo II

O MMV é executado como uma aplicação em um SO, chamado de sistema operacional hospedeiro. O SO hospedeiro não tem conhecimento sobre o MMV Tipo II e trata o MMV como qualquer outro processo, figura 5 (b). O SO convidado envia requisições de entrada/saída (E/S) que são capturadas pelo SO hospedeiro, que por sua vez, a envia ao *driver* de dispositivo que executa a operação de E/S (GALVIN, 2009). O Linux KVM (KVM,), VMware Workstation e VirtualBox (VIRTUALBOX,) são representantes de MMV Tipo II, baseados em módulos de *kernel* e uma aplicação a nível de usuário.

Figura 5 – Tipos de MMV

a. MMV Tipo I b. MMV Tipo II

Fonte: Fonte: (DESAI *et al.*, 2013).

— MMV Híbrido

O MMV Híbrido é implementado quando os MMV Tipo I e II podem ser suportados pelo processador. Todas as instruções privilegiadas são interpretadas em *software*, e *drivers* especiais devem ser escritos para o SO visitante. Um exemplo de MMV Híbrido é o Xen utilizando *drivers* paravirtualizados, abordagem em que o SO visitante é modificado de modo a permitir uma comunicação direta entre a MV e o *hardware* subjacente.

3.1 Técnicas de virtualização de sistemas

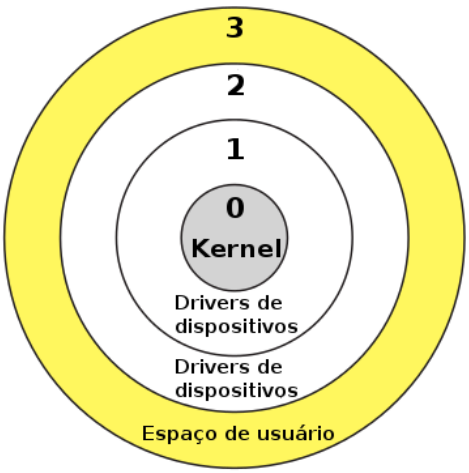
Esta seção sumariza as principais abordagens de virtualização existentes e realiza uma introdução a conceitos utilizados em diversas técnicas de virtualização. As abordagens seguem abaixo:

— Virtualização assistida a hardware

A virtualização assistida a *hardware* é realizada na *CPU* e *chipsets*, permitindo a tradução de instruções de *CPU* e endereços de memória entre o *hardware* real e as máquinas virtuais. O suporte às instruções em *CPU* são denominadas Intel VT-x e AMD-V e permitem às *CPU*s terem um nível de privilégio adicional para executarem máquinas virtuais (MUDA, 2010).

Esse tipo de virtualização tornou-se necessária devido a arquitetura de proteção em anel das *CPU*s Intel x86. Estas *CPU*s possuem quatro níveis de execuções, também chamados de anéis, onde o anel 0 possui o maior privilégio e o anel 3 o menor. *Kernels* são executados no anel 0 e aplicações são executadas no anel 3, ilustrados na Figura 6. O hipervisor deve ser executado em uma camada acima da camada do *kernel*, que já está ocupando a camada com maior nível de privilégios no sistema de proteção em anel.

Figura 6 – Anéis de execução da arquitetura x86



Fonte: Fonte: (MUDA, 2010).

Outro recurso necessário é a tradução do espaço de memória entre os endereços de dispositivos virtuais e reais, que é manipulada pelo *Input/Output Memory Management Unit*/Unidade de Gerenciamento de E/S de Memória (IOMMU).

Como exemplo desta abordagem, o *kernel* Linux possui recursos de assistência à virtualização, como o VirtIO. O módulo VirtIO provê suporte de E/S de disco e rede, alocação de espaço de memória de máquinas virtuais e emulação de dispositivos *Peripheral Component Interconnect* (PCI). O propósito do VirtIO é desenvolver um *driver* de dispositivo especialmente para E/S de dispositivos virtuais, e prover uma *interface* comum para múltiplas implementações de virtualização de *kernel* (MUDA, 2010).

— Virtualização Completa

Neste tipo de virtualização, o *hardware* hospedeiro é completamente abstraído e todas as características de um equipamento virtual são emulados, ou seja, todas as instruções solicitadas pelo sistema convidado são interpretadas no MMV. O sistema hospedado ignora a existência da máquina virtual e opera como se funcionasse diretamente sobre o sistema operacional para o qual foi projetado para funcionar. A virtualização completa é mais flexível em termos de SO convidado, uma vez que este não precisa ser modificado para implementação dessa técnica. Todas as instruções são interpretadas pelo MMV. Em compensação, essa interpretação de cada instrução provoca perda de desempenho de processamento, uma vez que o monitor de máquina virtual se utiliza de dispositivos de virtualização que atendem a uma gama de aplicativos

e, por isso, possuem uma baixa especialização (COELHO *et al.*, 2009).

— Paravirtualização

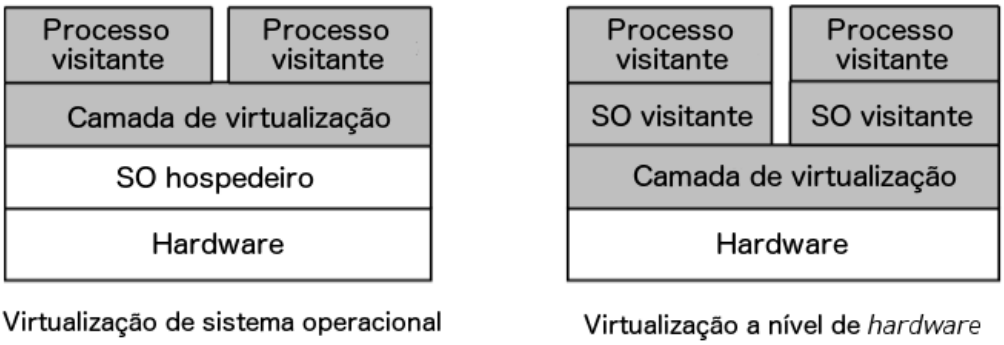
Os sistemas de paravirtualização oferecem *interfaces* especiais, chamadas *hypercalls*, aos sistemas operacionais convidados para alcançarem um desempenho próximo ao das execuções nativas. Nessa abordagem, o SO da máquina virtual não é idêntico àquele que seria executado no equipamento físico original, com o objetivo que o sistema hospedado possa enviar as instruções mais simples diretamente para o *hardware*, restando apenas as instruções de nível mais alto para serem interpretadas pelo MMV. Entretanto, esse procedimento requer que o sistema operacional convidado seja modificado para interagir com o MMV e selecionar quais instruções devem ser interpretadas nele ou diretamente no *hardware* hospedeiro. A paravirtualização é menos flexível, pois requer modificações no sistema operacional convidado para que este possa trabalhar perfeitamente. Porém, o fato do sistema operacional convidado saber que opera sobre uma máquina virtual e, com isso, mandar as instruções mais simples diretamente para o *hardware* diminui a sobrecarga no MMV e permite uma maior especialização dos dispositivos de virtualização. Dessa forma, os aplicativos operam mais próximos de sua capacidade máxima, melhorando seu desempenho em comparação à virtualização completa. Além disso, na paravirtualização, a complexidade das máquinas virtuais a serem desenvolvidas diminui consideravelmente (COELHO *et al.*, 2009).

— Virtualização de sistema operacional

A virtualização de sistema operacional é uma tecnologia que particiona o sistema operacional para criar múltiplas máquinas virtuais isoladas. Isso significa que todas as máquinas virtuais deste tipo compartilham o mesmo *kernel* do sistema operacional hospedeiro. Uma máquina virtual desta categoria é um ambiente de execução virtual que pode ser instanciada instantaneamente a partir do sistema operacional hospedeiro. Este tipo de virtualização tem sido amplamente utilizado para melhorar a segurança, gerenciamento e disponibilidade de ambientes complexos com baixa sobrecarga de recursos e alterações mínimas à infraestrutura computacional (YU, 2007). A Figura 7 compara as camadas da virtualização de sistema operacional e virtualização de *hardware*.

O principal desafio com este tipo de virtualização é como alcançar um forte isolamento entre todas as máquinas virtuais que compartilham o mesmo sistema operacional

Figura 7 – Comparação entre as camadas de virtualização



Fonte: Fonte: (YU, 2007).

hospedeiro.

— Migração de máquinas virtuais

Um dos grandes benefícios da virtualização é a migração de máquinas virtuais. Utilizando este recursos, as máquinas virtuais podem ser movidas de um hipervisor para outro sem o término de execução da máquina virtual. A migração permite que máquinas virtuais sejam executadas temporariamente em outro hipervisor enquanto sua máquina física de origem esteja em manutenção, ou que máquinas virtuais ociosas sejam migradas para determinado hipervisor visando diminuir a quantidade de máquinas físicas necessárias para suas execuções ou mesmo para realizar uma alocação inteligente de máquinas virtuais que demandem grande quantidade de recursos a determinados *hosts* objetivando evitar que a disputa por recursos impacte o desempenho dos sistemas (MUDA, 2010).

3.2 Plataformas de virtualização

Existem várias plataformas de virtualização disponíveis, entre elas Microsoft Hyper-V, VMWare ESX, VirtualBox, KVM, Xen e OpenVZ. A seguir descreve-se as principais plataformas livres para sistemas Linux:

— KVM

O KVM é um monitor de máquinas virtuais que utiliza virtualização completa, inicialmente desenvolvido pela Qumranet em Israel. É um módulo adicionado ao *kernel* Linux, o que permite aproveitar todas as vantagens do *kernel* Linux padrão e virtualização assistida a

hardware, Intel VT ou AMD-V.

O KVM implementa a virtualização adicionando um novo modo de execução ao *kernel* Linux chamado visitante, ilustrado na Figura 8, que possui seu próprio *kernel* e modo usuário, além de respostas para execução de códigos de sistemas operacionais visitantes. Quando um processo visitante está executando uma operação que não seja de E/S, ele a executará no modo visitante. No caso de execuções de E/S ou outras instruções especiais, a MV se comunica diretamente com o *driver* KVM do kernel.

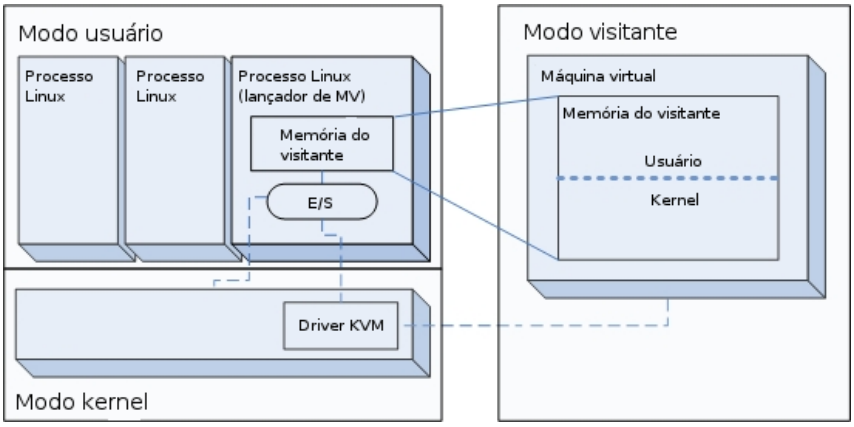
O KVM consiste de dois componentes: o módulo e o espaço de usuário. O módulo enxerga a virtualização para o *kernel* Linux e requer um processador com suporte a extensões de virtualização. Na virtualização completa, o MMV existe entre os sistemas operacionais virtualizados e o *hardware*. Essa camada multiplexa os recursos entre várias instâncias de sistemas operacionais concorrentes. O MMV é composto de vários componentes que geralmente incluem um agendador, gerenciador de memória, pilha de E/S e também drivers de dispositivos. O KVM, diferentemente do Xen, possui implementação focada na virtualização do visitante, permitindo que o *kernel* Linux atue como MMV (MAGNUS; OPSAHL, 2013).

Na arquitetura KVM, cada MV é implementada e tratada como um processo comum. O KVM utiliza o emulador QEMU para simular o *hardware* como controlador de memória, interfaces de rede e BIOS.

— Xen

O hipervisor Xen é uma camada de *software* que reside diretamente sob o *hardware* abaixo do SO hospedeiro, originalmente desenvolvido na Universidade de Cambridge. É

Figura 8 – Modos de execução KVM



Fonte: Fonte: Adaptado de (PROTTI, 2009).

responsável pelo agendamento de *CPU* e particionamento de memória de várias MVs executadas sob o *hardware*.

A arquitetura Xen possui dois domínios distintos: Dom 0 e Dom U. O Domínio Dom 0 é um SO Linux modificado com privilégios especiais. Possui acesso direto a recursos físicos de E/S e capacidade de interação com outras MVs executadas no mesmo *hardware* físico.

Um ambiente de virtualização Xen é formado de vários componentes, ilustrados na Figura 9, que trabalham em conjunto:

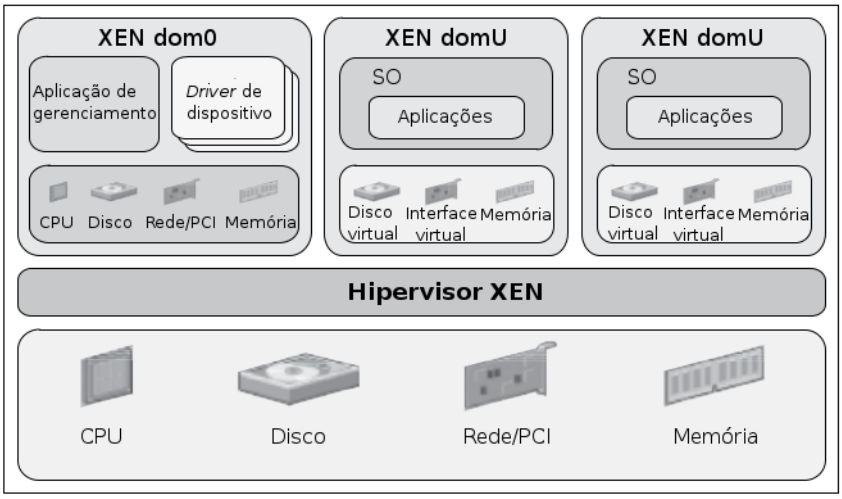
- Hipervisor Xen
- Visitante Domínio 0 (Dom0)
- Visitante Domínio U (DomU)
  - Visitante de Paravirtualização (PV)
  - Visitante *Hardware Virtual Machine* (HVM) (Virtualização assistida a *hardware*)

O domínio Dom0 é um *kernel* Linux modificado que funciona como uma máquina virtual única que possui privilégios especiais para acessar os recursos de E/S e interagir com outras MVs executadas no sistema. A execução do Dom 0 é pré-requisito para que qualquer outra máquina virtual Xen seja iniciada.

Dois *drivers* estão inclusos no Dom0 com o objetivo de prover suporte a requisições de rede e disco dos DomU (PV e HVM). Tais *drivers* comunicam-se diretamente com o *hardware* local para processar todas as requisições das MVs.

Os visitantes DomU não possuem acesso direto ao *hardware* da máquina, diferen-

Figura 9 – Componentes Xen



Fonte: Fonte: (XEN, 2009).

temente do Dom0. Todas as MVs paravirtualizadas executadas em um hipervisor Xen são chamadas DomU PV e são necessariamente sistemas operacionais modificados (Linux, Solaris, FreeBSD e outros sistemas baseados em UNIX). As MVs baseadas em virtualização completa são denominadas DomU HVM e podem executar sistemas operacionais não modificados como o Microsoft Windows.

Os visitantes DomU PV possuem conhecimento que não têm permissão para acessar diretamente o *hardware* e que devem direcionar as suas requisições ao Dom0, diferentemente de sistemas visitantes DomU HVM que não têm conhecimento que compartilham *hardware* e que outras MVs estão presentes.

As máquinas virtuais DomU PV utilizam os *drivers* de dispositivos paravirtualizados para realizar comunicações de E/S. As máquinas virtuais DomU HVM não possuem os drivers PV localizados dentro da MV, em vez disso, um serviço chamado Qemu-dm que tem por finalidade emular o *hardware* para a MV, é iniciado para cada MV HVM para fornecer recursos de E/S para os visitantes HVM (XEN, 2009).

— OpenVZ

O OpenVZ é uma tecnologia de virtualização de sistema operacional baseada no SO Linux. OpenVZ permite a um servidor físico executar diversas instâncias isoladas de sistemas operacionais, conhecidas como contêineres ou Servidores Virtuais Privados. Diferentemente de KVM e Xen onde cada visitante possui seu próprio *kernel*, no OpenVZ todos os contêiners possuem apenas um *kernel* em comum com o sistema operacional hospedeiro, essa é uma importante diferenciação de outras plataformas em termos de isolamento e desempenho. Logo, apenas sistemas operacionais Linux podem ser virtualizados com OpenVZ. Uma falha, ou comprometimento de segurança, do kernel causado por algum visitante, pode também comprometer outros sistemas operacionais (VASCONCELOS; FREITAS, 2014b).

Uma grande vantagem do OpenVZ é que este possui uma menor sobrecarga de processamento que outras plataformas de virtualização, permitindo um tempo de resposta mais rápido aos clientes que acessam seus serviços, pois, cada processo em um SO virtualizado é um processo comum em execução no SO da máquina física (REGOLA; DUCOM, 2010).

O OpenVZ inclui um *kernel* Linux customizado com funções de virtualização, isolamento, *checkpointing*, gerenciamento de recursos e e várias ferramentas de usuário (CHE *et al.*, 2010).

3.3 Conclusão

A virtualização é a tecnologia chave para as nuvens de IaaS devido a suas características de consolidação de servidores, isolamento e melhor utilização de recursos. O estudo do desempenho de plataformas virtuais como o KVM, Xen e OpenVZ veio sendo realizado em trabalhos anteriores, (VASCONCELOS; FREITAS, 2013), no decorrer do curso de mestrado. Foram realizadas avaliações de desempenho do KVM e Xen como plataformas de virtualização completa e observou-se que o KVM utilizando drivers paravirtualizados de acesso a dispositivos possui um melhor desempenho geral de virtualização. Os micro testes realizados na seção 5.9 demonstram o desempenho superior das plataformas OpenVZ e KVM em comparação ao Xen.

A virtualização completa oferece um meio cômodo de se realizar virtualização pois provê um ambiente completamente abstraído e sem a necessidade de adequações nos sistemas visitantes. Já a virtualização de sistema operacional oferece uma alternativa leve de virtualização na qual o sistema operacional hospedeiro é particionado em vários contêineres que alcançam taxas de execução e acesso a dispositivos próximos das alcançadas pelo sistema nativo, entretanto, apenas sistemas visitantes Linux podem ser virtualizados nessa plataforma. São duas abordagens bastantes utilizadas de virtualização e com vantagens distintas. A escolha das plataformas KVM e OpenVZ de virtualização completa e de sistema operacional deu-se pelo seus bons desempenhos, relatados na literatura (XAVIER *et al.*, 2013), (CHE *et al.*, 2010), (XAVIER *et al.*, 2014) e observados durantes os testes realizados no decorrer da pesquisa e por não necessitarem de mudanças profundas nos sistemas visitantes para que o mecanismo de virtualização funcione.

4 TRABALHOS RELACIONADOS

O desempenho da camada de virtualização dos MMVs é tema de vários trabalhos devido a importância da constante necessidade de diminuição da sobrecarga de virtualização e desenvolvimento de novas técnicas de virtualização e hipervisores. Os trabalhos relacionados são categorizados a seguir:

4.1 Virtualização completa e paravirtualização

Em (CHE *et al.*, 2010) foi discutido o desempenho das plataformas de virtualização Xen, KVM e OpenVZ através de micro testes de recursos e avaliações com aplicações servidoras *web*, Java, banco de dados e compilação do *kernel* Linux a fim de medir o desempenho da virtualização de processador, sistema de arquivos, rede e prover uma comparação quantitativa e qualitativa entre os três MMVs. O desempenho de virtualização de processador medido através dos softwares LINPACK, SPEC*CPU* e compilação do Linux mostrou-se bem próximo do nativo para as plataformas OpenVZ e Xen. A plataforma KVM apresentou menor poder de computação para alguns testes de operações de ponto flutuante. O estudo da virtualização de memória foi realizado com os softwares RAMSPEED e LMBench e sugeriu que a camada de virtualização tem pouco impacto em operações de acesso à memória, especialmente na largura de banda de acesso. A plataforma OpenVZ mostrou melhores taxas nos testes de *stress* de servidores web e bancos de dados, seguido de Xen e KVM. Para inspecionar a capacidade das plataformas de atuarem como servidores Java, os autores utilizaram o *software* SPEC JBB2005. Os resultados mostraram que Xen alcançou as maiores taxas, seguido de OpenVZ e KVM, principalmente devido à capacidade do Xen de paravirtualizar várias transações Java. Através da execução desses testes os autores observaram que OpenVZ alcançou melhor desempenho geral, seguido de Xen e KVM. O KVM obteve taxas muito inferiores às outras plataformas. Os autores não citam se foram utilizados *drivers* paravirtualizados nas máquinas virtuais KVM, o que pode vir a ser um dos motivos das baixas taxas alcançadas nessa plataforma.

Em outro estudo, (PADALA *et al.*, 2007), foi analisado o desempenho de virtualização Xen e OpenVZ quando utilizados para consolidação de servidores para uma aplicação multi-camada desenvolvida. Seus experimentos mostraram que Xen apresenta maior sobrecarga de virtualização e que o tempo de resposta médio da aplicação aumentou em até 400% no Xen e apenas 100% no OpenVZ à medida em que aumentava-se a quantidade de instâncias da aplicação.

Os autores reportaram que esse comportamento se deve ao Xen apresentar uma maior taxa de erros de busca de cache L2 em comparação ao OpenVZ.

4.2 Virtualização de sistemas operacionais

Um estudo realizado por (XAVIER *et al.*, 2014) conduziu vários experimentos com o objetivo de realizar um avaliação das plataformas de virtualização de sistema operacional baseadas em contêiners para execução de cargas de trabalho de Computação de Alto Desempenho (CAD). Os autores escolheram as plataformas Vserver, OpenVZ e *Linux Containers* (LXC) e executaram uma série de testes como Linpack, Stream, NetPipe e uma suíte de testes de isolamento de sistemas. As aplicações de CAD somente conseguirão obter vantagens dos sistemas de virtualização se a sobrecarga desta for reduzida. Nesse sentido, foi descoberto que todos os sistemas baseados em contêiners possuíam desempenho próximo ao nativo de *CPU*, memória, disco e rede. As principais diferenças residiam na implementação de gerenciamento de recursos e isolamento.

4.3 Desempenho de *clusters* Hadoop virtualizados

Outros trabalho analisam o desempenho do Hadoop em plataformas em nuvem. Em (JACOB; BASU, 2013) é analisado o desempenho da execução da suíte de desempenho *K-Means Clustering Algorithm* em um cluster Hadoop implantado em nuvens Eucalyptus com ferramentas como Ganglia e TestDFSIO. O trabalho discute como o desempenho do algoritmo escala com os números de nós na nuvem Eucalyptus. Foi notado que o aumento na quantidade de nós incrementou significativamente o desempenho do *cluster*, concluindo que a nuvem Eucalyptus é uma plataforma muito conveniente para execução de cargas de trabalho MapReduce.

No trabalho (IBRAHIM *et al.*, 2009) é discutido o desempenho entre *clusters* Hadoop físicos e virtuais. Os autores listam vários questionamento com relação ao problema e alertam para condições de concorrência de E/S quando várias MVs estão implantadas em um mesmo nó.

Como um esforço para minimizar a degradação de desempenho para cargas de trabalho virtualizadas Hadoop , os autores em (KANG *et al.*, 2011) apresentaram um agendador para o MMV que implementa três mecanismos para melhorar a eficiência e o balanceamento do Xen. As características das cargas de trabalho MapReduce facilitam a conglomeração de requisições de E/S de MVs que trabalham no mesma tarefa MapReduce, o que reduz o

número de mudança de contextos. Outra característica observada foi que a maioria das tarefas MapReduce possuem uma quantidade significante de eventos bloqueadores de E/S, acarretando que a finalização de uma tarefa depende de outros nós.

4.4 Conclusão

Desempenho de virtualização é um campo de estudo bastante pesquisado devido a constante necessidade da diminuição da sobrecarga para melhor execução de sistemas. A pesquisa na literatura revelou esforços como desenvolvimento de novos algoritmos agendadores de MVs e comparativos de desempenhos entre aplicações.

A utilização de virtualização como plataforma para aplicações de alto desempenho (*big data*, CAD) apresenta-se como uma alternativa eficiente, flexível e econômica aos *clusters* físicos. Mas essa eficiência só pode ser alcançada com uma baixa sobrecarga de virtualização, fator que destaca a importância de estudos de análise de desempenho das plataformas virtuais e em nuvem. Este trabalho contribui com a análise de desempenho da execução de cargas de trabalho MapReduce em clusters Hadoop virtualizados sob diferentes plataformas e técnicas de virtualização, pois até a data de publicação desta dissertação não haviam trabalhos que abordassem testes de desempenho entre essa combinação de *softwares*. Administradores de sistemas, arquitetos de soluções e pesquisadores de linhas de pesquisas relacionadas podem utilizar os resultados deste trabalho como parâmetros de avaliação durante o planejamento, seleção ou implantação de sistemas virtualizados.

5 PROPOSTA E RESULTADOS

A proposta deste trabalho é avaliar o desempenho de *clusters* Hadoop virtualizados na execução de tarefas MapReduce. Tais *clusters* estão hospedados nas nuvens OpenNebula que utilizam camadas de virtualização distintas: OpenVZ e KVM. Para realizar essa análise foram executadas duas baterias de avaliações: micro e macro testes.

Os macro-testes compreendem os testes MapReduce. São programas que utilizam grande quantidade de variados recursos computacionais como acesso a disco, processamento, memória e comunicação via rede simultaneamente. Estes macro-testes são exemplos de aplicações da vida real e seus resultados permitem analisar o desempenho do sistema como um todo. Os micro-testes são programas de avaliação de desempenho de recursos isolados do sistema, como rede e processamento. Tais programas são úteis para analisar o desempenho isolado da virtualização dos recursos computacionais.

5.1 Cenários de avaliação

Os ambientes experimentais consistem de dois servidores idênticos IBM BladeCenter HS21 com dois processadores Intel Xeon E5-2620 de 2GHz, 6 núcleos de processamento cada e tecnologia *Hyper-threading*, 48 GB de RAM DDR3 ECC, discos rígidos *Serial Attached Scsi* (SAS) de 300 GB de armazenamento com taxa de transmissão de 6 Gbps, interconectados através de uma rede *Gigabit Ethernet*. O sistema operacional utilizado nas máquinas físicas foi o Ubuntu GNU/Linux 14.04.1 LTS amd64. A versão do OpenNebula utilizada foi a 4.8.0.

Cada *cluster* Hadoop consiste de seis MVs, uma mestre e cinco escravas com a mesma configuração: sistema operacional Ubuntu GNU/Linux amd64 LTS 12.04, 2 vCPUs, 2 GB de vRAM, 1 GB de memória *swap* e 10 GB de disco. Foi utilizado o sistema de arquivos EXT4 em todas as MVs.

A versão do QEMU/KVM 2.0.0 utilizada é a 2.0.0, os drivers paravirtualizados VirtIO estão inclusos no *kernel* Linux no módulo virtio.ko. O *kernel* OpenVZ utilizado foi o 2.6.32-openvz-amd64-042stab093.5. Foi utilizado o Citrix XenServer 6.5 para a execução dos microtestes no ambiente Xen.

Nos *clusters* MapReduce foram utilizados Hadoop 1.2.1 e Java Oracle versão 1.7.0u45. O sistema de arquivos distribuído HDFS possui 60 GB (6x 10 GB por MV), o tamanho de bloco HDFS foi configurado em 64 MB e o fator de replicação configurado em 3. O

tempo de expiração das tarefas foi configurado em 30 minutos.

5.2 Metodologia

Todas as ferramentas descritas nesta seção são parte da distribuição Apache Hadoop, portanto, já disponíveis na instalação padrão de um cluster Hadoop. Os resultados das medições destas ferramentas foram redirecionados para arquivos de textos para serem posteriormente organizados e processados, para finalmente serem gerados os respectivos gráficos. Os testes foram executados 5 vezes e os gráficos abaixo demonstram a média e o erro padrão dos valores alcançados.

5.3 Wordcount

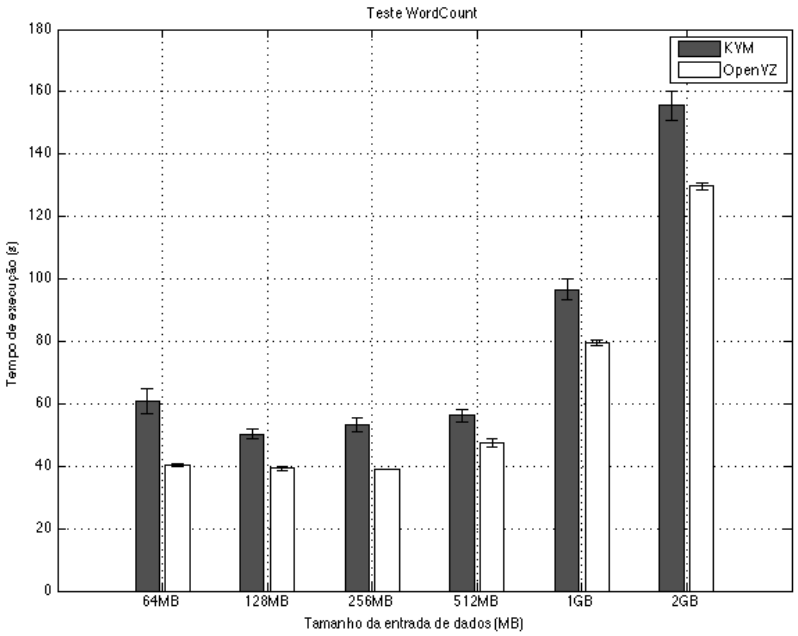
O teste de desempenho WordCount é uma aplicação que recebe arquivos de texto como entrada e computa o número de ocorrências de cada palavra nos arquivos. A saída do programa é um arquivo de texto em que cada linha contém uma palavra e o seu número de ocorrências separados por uma tabulação.

Cada tarefa *Map* recebe uma linha como entrada e a quebra em palavras e gera um par chave-valor contendo a palavra e o valor 1 (um). Os processos *Reduce* somam as ocorrências de cada palavra, realizam combinações e retornam um par chave-valor único contendo a sua soma. Os processos *Reduce* são utilizados para realizar combinações na saída dos processos map. A combinação de cada palavra em um único registro contribui para a diminuição da quantidade de dados a serem enviados pela rede (APACHE, 2015).

O WordCount já vem incluso na instalação padrão do Hadoop e é largamente utilizado como um método para comparação de desempenho entre *clusters* Hadoop. Os arquivos de entrada utilizados nesta avaliação foram gerados a partir da concatenação de arquivos de textos baixados do Projeto Gutenberg (GUTENBERG, 2015), que oferece vários livros eletrônicos em diferentes formatos e de forma gratuita. Os tamanhos de arquivos de texto utilizados foram: 64 MB, 128 MB, 256 MB, 512 MB, 1 GB e 2 GB.

Como pode-se observer na Figura 10, o OpenVZ alcançou os menores tempos de processamento do WordCount. Os resultados das execuções OpenVZ obtiveram valores mais constantes que as execuções KVM, que obtiveram maiores variações nos tempos de execução do WordCount.

Figura 10 – Resultados do teste WordCount



Fonte: Elaborado pelo autor.

5.4 TeraSort

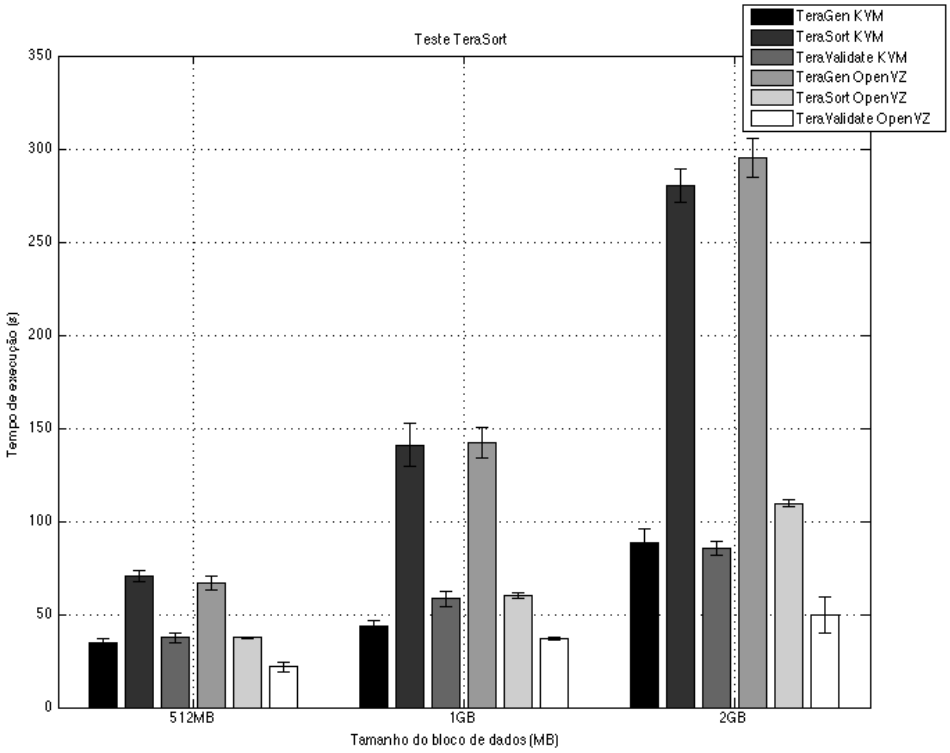
O objetivo do aplicativo TeraSort é ordenar certo volume de dados o mais rapidamente possível. É um teste que combina o uso das camadas HDFS e MapReduce. É composto de três aplicações MapReduce:

- TeraGen: responsável pela geração dos dados.
- TeraSort: replica os dados de entrada e realiza a ordenação dos dados.
- TeraValidate: valida se a saída do TeraSort está ordenada.

O usuário especifica a quantidade de linhas e o TeraGen gera uma quantidade de caracteres aleatórios de 'A' a 'Z'. O TeraSort é um programa MapReduce que realiza a ordenação dos dados gerados, e faz uso de intenso poder de computação e tráfego de rede na comunicação entre os nós.

O TeraValidate garante que a saída das operações acima está globalmente ordenada. Ele cria um processo *Map* por arquivo no diretório de saída e cada *Map* garante que cada chave é menor ou igual a anterior. Os processos *Map* também geram registros com a primeira e última chaves do arquivo e os processos *Reduce* garantem que a primeira chave de um arquivo *i* é maior do que a última chave *i*-1.

Figura 11 – Resultados do teste TeraSort



Fonte: Elaborado pelo autor.

Foram utilizados tamanhos variados para a geração dos dados de entrada pelo TeraGen: 512 MB, 1GB e 2GB.

A Figura 11 ilustra o desempenho das duas nuvens na execução do TeraSort com entradas de vários tamanhos. Observa-se que o KVM obtem menores tempos de execução nos testes de geração de dados com o TeraGen, entretanto, o OpenVZ alcança valores menores de tempo de execução e erro padrão nos testes TeraSort e TeraValidate.

5.5 TestDFSIO

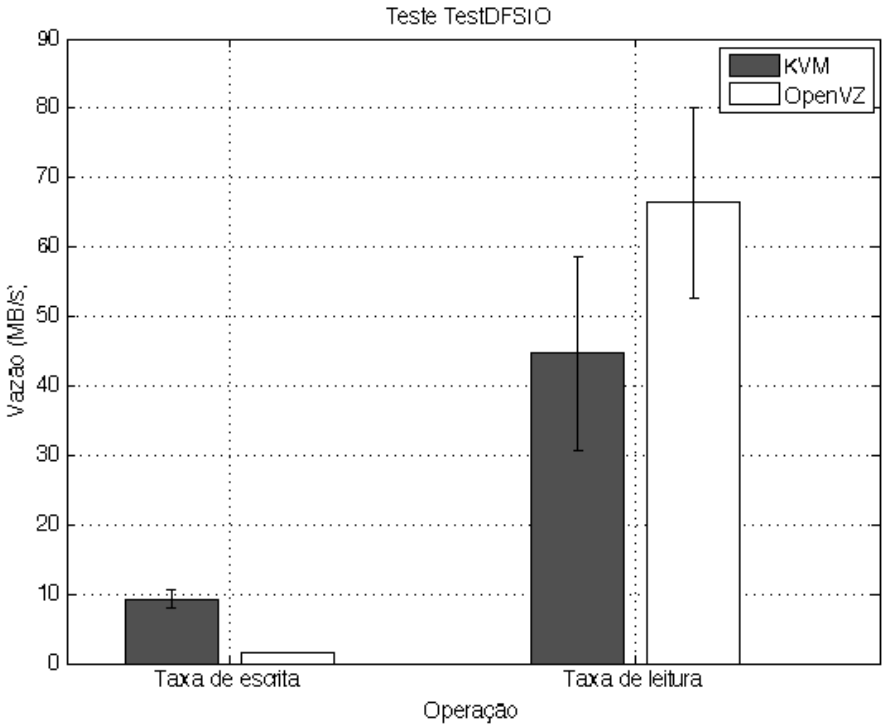
O aplicativo TestDFSIO é um teste de escrita e leitura do HDFS. É útil para realizar testes de *stress* no HDFS, avaliar parâmetros nos *NameNodes* e *DataNodes*, descobrir gargalos e avaliar a taxa de E/S do cluster. O TestDFSIO consiste em dois testes: o primeiro, gera e escreve os arquivos em um diretório HDFS; o segundo lê os arquivos criados pelo primeiro e realiza as medições.

Cada arquivo é lido ou escrito em uma tarefa *Map* separada e a saída desse processo é usada para coletar estatísticas relacionadas ao arquivo processado. As estatísticas são acumuladas

na fase *Reduce* para produzir o relatório de desempenho.

Nas execuções realizadas, foi ordenado ao TestDFSIO criar 10 arquivos de 100 MB no HDFS. A Figura 12 descreve o desempenho das duas plataformas durante o teste. Neste teste o OpenVZ obteve taxas menores que o KVM nos testes de escrita, entretanto, nos testes de leitura o desempenho do OpenVZ foi aproximadamente 50% superior ao KVM e suas amostras apresentaram erros padrão ligeiramente inferiores.

Figura 12 – Resultados do teste DFSIO

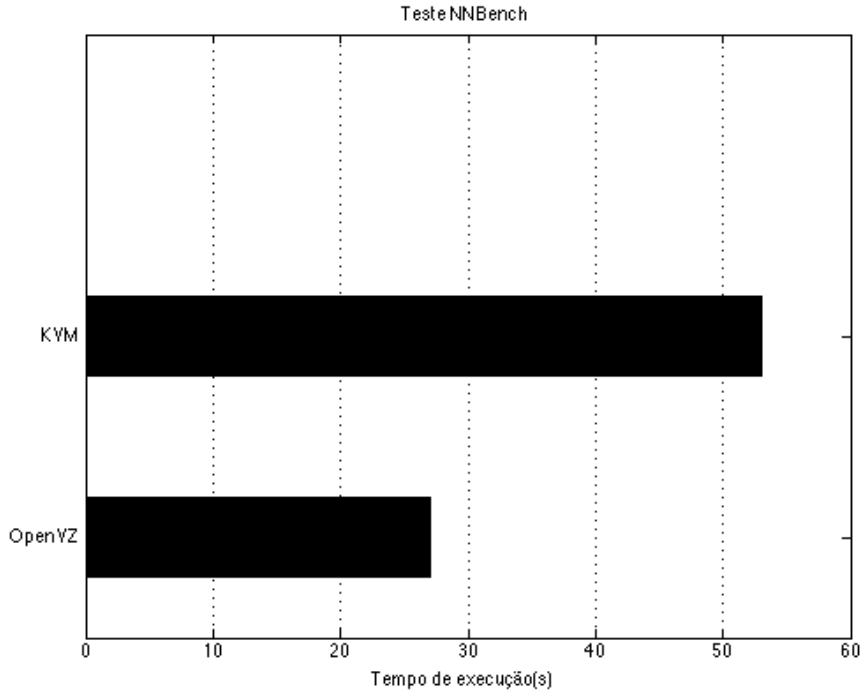


Fonte: Elaborado pelo autor.

5.6 NNBench

O *software* NNBench é útil para avaliar o desempenho e configurações do *NameNode*. Esse teste gera várias requisições de pequeno tamanho ao HDFS objetivando um teste de valor suportado de requisições. O NNBench pode simular requisições para criação, leitura, renomeação e deleção de arquivos no HDFS. Nos testes executados, NNBench criou 1000 arquivos utilizando 6 processos maps e 2 reduces. A Figura 13 descreve o tempo de execução do NNBench para cada nuvem.

Figura 13 – Resultados do teste NNBench



Fonte: Elaborado pelo autor.

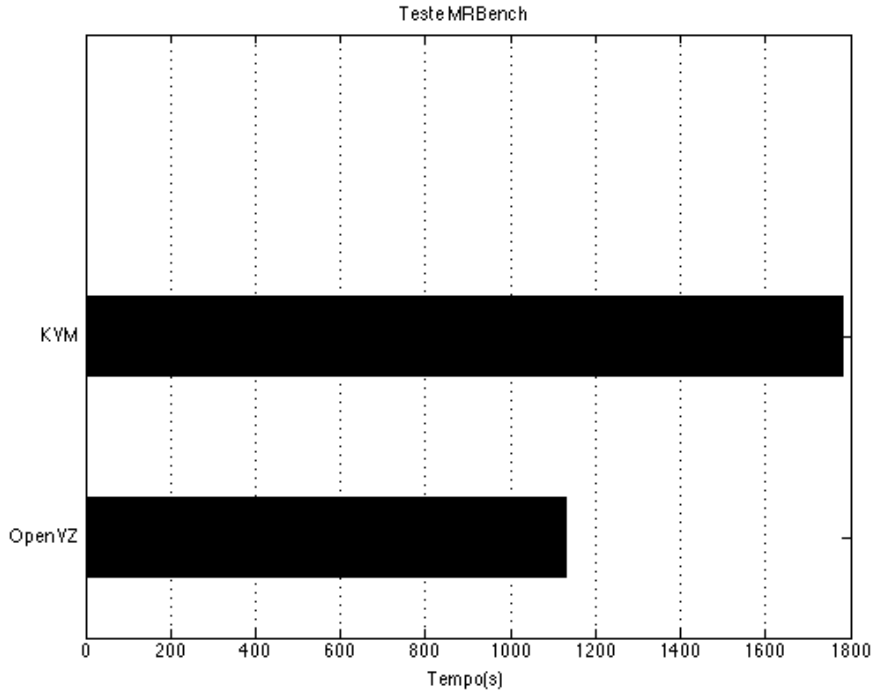
5.7 MRBench

O utilitário MRBench executa em *loop* uma pequena tarefa MapReduce. Esse utilitário age como um teste de desempenho complementar ao TeraSort pois o MRBench verifica tanto se os processos estão respondendo e executando de forma eficiente no cluster. O objetivo desse teste de desempenho é avaliar a camada MapReduce, pois o seu impacto no HDFS é muito limitado. A Figura 14 ilustra o tempo de execução do utilitário MRBench nos *clusters*.

5.8 Pi

O utilitário Hadoop Pi é um programa MapReduce que estima o valor do número Pi utilizando o método estatístico quasi-Monte Carlo, em que pontos colocados aleatoriamente em um quadrado de unidade também caem dentro de um círculo inscrito nesse quadrado com uma probabilidade igual à área do círculo, Pi/4. O valor de Pi pode ser estimado do valor de $4R$, no qual R é a proporção do número de pontos que estão dentro do círculo em relação ao número total de pontos que estão dentro do quadrado (LIMA, 2014). Quanto maior a amostra de pontos usados, melhor será a estimativa. Este teste possui foco na capacidade do *cluster*, envolvendo

Figura 14 – Resultados do teste MRBench



Fonte: Elaborado pelo autor.

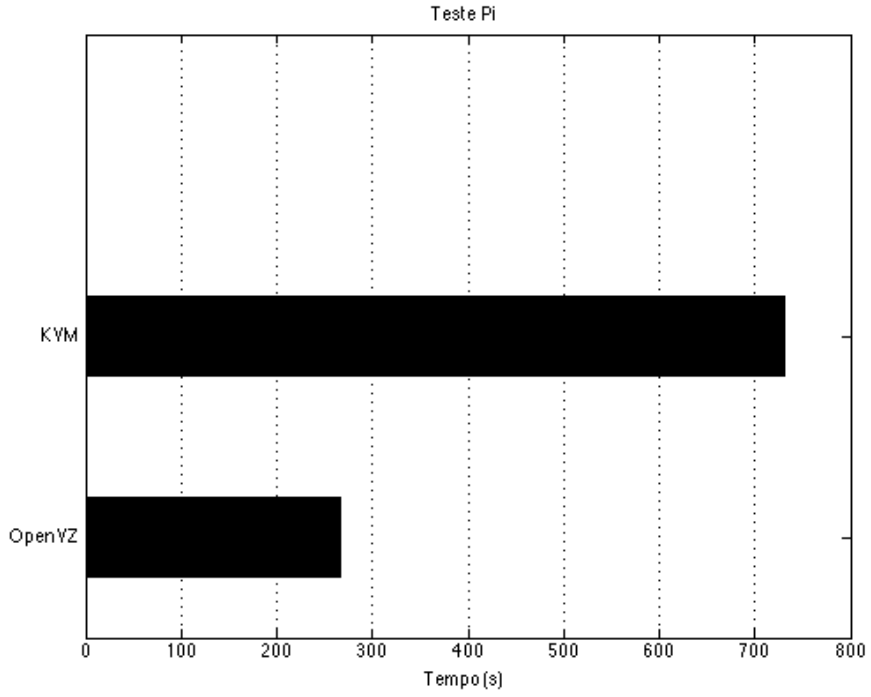
pouca utilização de E/S de disco e tráfego de rede. A Figura 15 ilustra o tempo de execução do utilitário Pi com seis terefas *Map* e dez bilhões de amostras por tarefa, sob os dois tipos de virtualização.

5.9 Avaliação dos resultados

Pode-se observar pelos testes executados que a nuvem executada sob a plataforma OpenVZ possui menor sobrecarga de virtualização de *CPU*, alcançando tempos de execução menores em testes que fazem uso intensivo de *CPU* como WordCount, TeraSort, TeraValidate, MRBench e Pi. O desempenho do OpenVZ também é superior para a maioria dos testes de E/S de disco, principalmente, para os testes de leitura em disco do TestDFSIO. O KVM, por sua vez, obteve taxas superiores ao OpenVZ nos testes de criação de grandes arquivos no TeraGen e no teste de escrita do TestDFSIO. O tempo de execução do KVM na criação sequencial de pequenos arquivos no NNBench foi quase o dobro que o alcançado pelo OpenVZ.

Com o objetivo de investigar a sobrecarga de virtualização e comprovar os resultados, foram realizados micro testes de desempenho isolados, e divididos por recurso, diretamente nas plataformas de virtualização, fora do *cluster* Hadoop. Os gráficos abaixo descrevem a média de

Figura 15 – Resultados do teste Pi



Fonte: Elaborado pelo autor.

cinco execuções dos testes.

5.9.1 E/S de disco

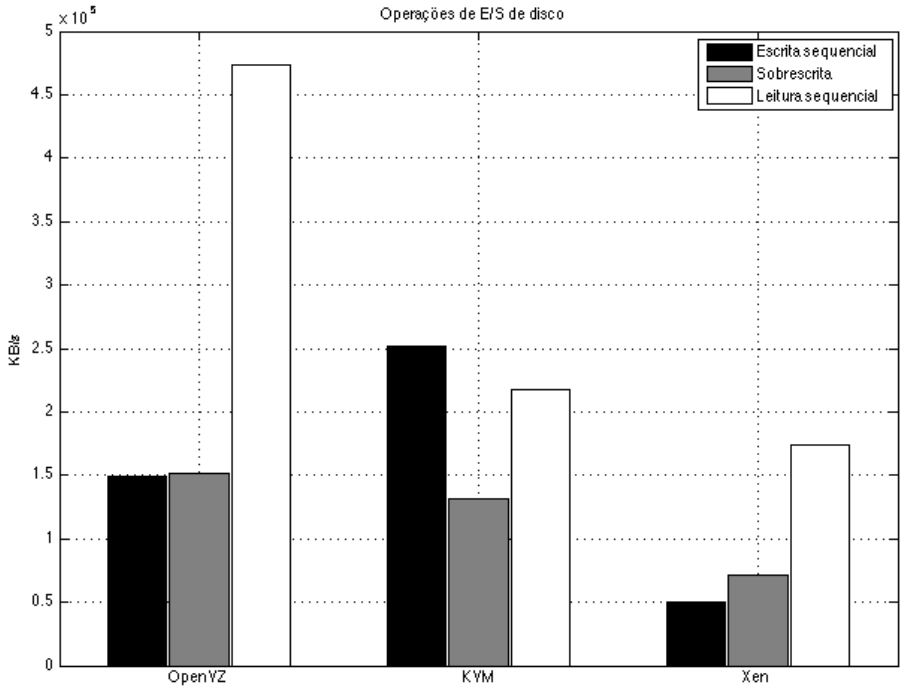
Foi executado o utilitário bonnie++ v1.96 em MVs KVM, Xen HVM e OpenVZ. Bonnie++ é um programa de avaliação de desempenho para discos rígidos e sistemas de arquivos. Existem vários tipos de operações com arquivos utilizadas por várias aplicações e em diferentes intensidades. Bonnie++ testa algumas destas operações e para cada teste exibe a taxa alcançada e o percentual de utilização da *CPU*.

Foram avaliadas três operações de E/S de disco: leitura, escrita e reescrita de dados. A Figura 16 descreve os resultados obtidos.

Pode-se notar que o OpenVZ obteve desempenho muito superior ao KVM e Xen no teste de leitura sequencial em disco e o KVM alcançou melhores taxas em escrita sequencial. Estes resultados comprovam o melhor desempenho alcançado pelo OpenVZ nos testes de leitura em disco durante o TestDFSIO e TeraValidate, e as taxas superiores alcançadas pelo KVM no teste de escrita durante a execução do TestDFSIO.

Foram calculadas com bonnie++ as taxas de operações de metadados de arquivos.

Figura 16 – Resultados do teste de operações de E/S em disco com Bonnie++



Fonte: Elaborado pelo autor.

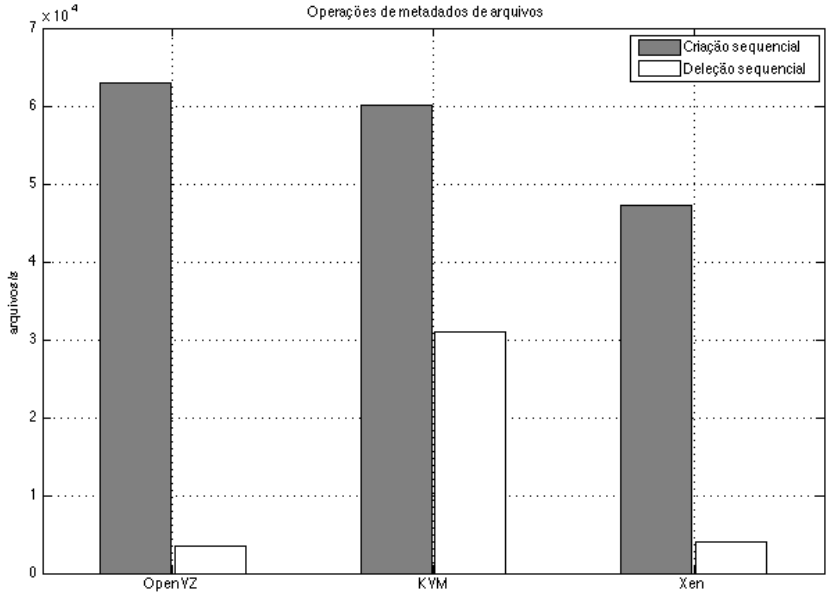
Essas operações incluem a criação e deleção, de forma sequencial e aleatória, de pequenos arquivos no sistema de arquivos. Os resultados são ilustrados na Figura 17.

O desempenho do OpenVZ obteve melhores taxas de criação sequencial de vários arquivos, esse teste ajuda a compreender o menor tempo de execução dessa plataforma no teste NNBench, em que são criados vários arquivos no HDFS.

Observamos que as taxas de deleção de arquivos do KVM foi maior que a alcançada pelo OpenVZ, entretanto essa operação foi pouco utilizada pelos testes de desempenho MapReduce.

A Figura 18 demonstra a utilização do processador durante os testes de disco. OpenVZ e sua implementação de virtualização de sistema operacional, obteve uma menor sobrecarga de *CPU* em quase todos os testes como criação, reescrita e busca, exceto o teste de leitura em disco em que o KVM obteve menor uso de *CPU*. A plataforma Xen alcançou pequenas taxas de utilização de *CPU*, entretanto, seu desempenho nos mesmos testes foi bastante inferior em comparação às outras plataformas.

Figura 17 – Resultado do teste de operações com metadados de arquivos com Bonnie++



Fonte: Elaborado pelo autor.

5.9.2 Desempenho de rede

A rede foi intensamente utilizada durante a avaliação, tanto *jobs* MapReduce quanto pelo HDFS. Com o objetivo de verificar isoladamente o desempenho das duas nuvens foi utilizada a ferramenta Iperf para gerar um fluxo de dados e medir a largura de banda máxima alcançada na interface de rede virtual. A Figura 19 descreve os resultados da execução do teste Iperf durante 60 segundos.

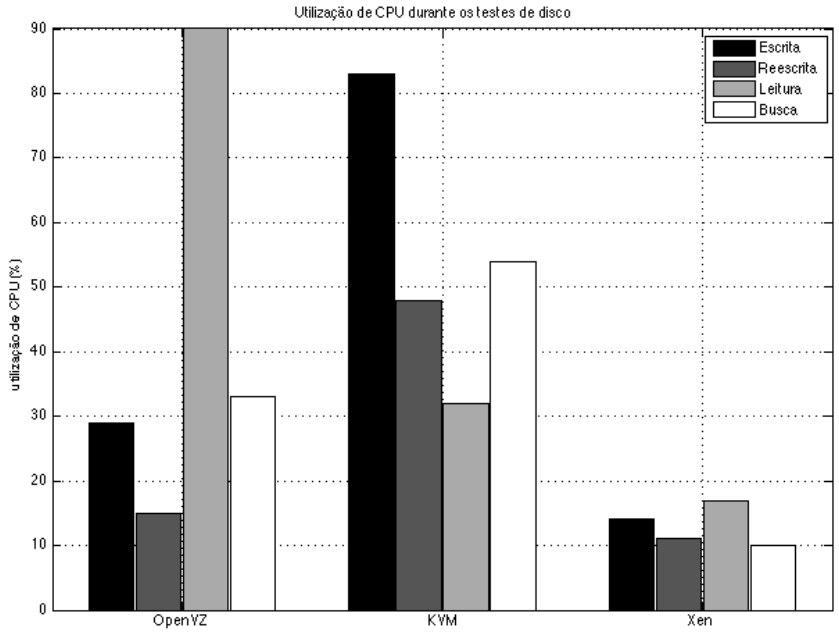
Através dos resultados observamos que ambas as nuvens alcançaram taxas de rede muito próximas das alcançadas pelo sistema nativo, o que mostra que as plataformas de virtualização possuem bom desempenho de rede.

5.9.3 Processamento

A capacidade de computação das plataformas foi avaliada utilizando os softwares LINPACK 11.3 e Systester 1.5.1. LINPACK mede a capacidade de processamento de instruções de ponto flutuante pelo cálculo de sistemas lineares utilizando fatoração LU. LINPACK foi executado com entradas de diferentes tamanhos: matrizes de ordem 1000 a 30000. A Figura 20 ilustra os resultados do teste.

O OpenVZ obteve uma taxa de *Giga FLoating-point Operations Per Second*/Bilhões

Figura 18 – Utilização de *CPU* durante os testes de E/S em disco



Fonte: Elaborado pelo autor.

de Operações de ponto-flutuante por segundo (GFLOPS) maior que KVM e Xen, e muito próxima das taxas alcançadas pelo sistema nativo.

O utilitário Systester é um teste de *stress* para medir o desempenho e estabilidade do sistema através do cálculo de várias casas decimais do número Pi. Nas execuções realizadas, o Systester foi informado a calcular o valor de Pi com 128 milhões de casas decimais.

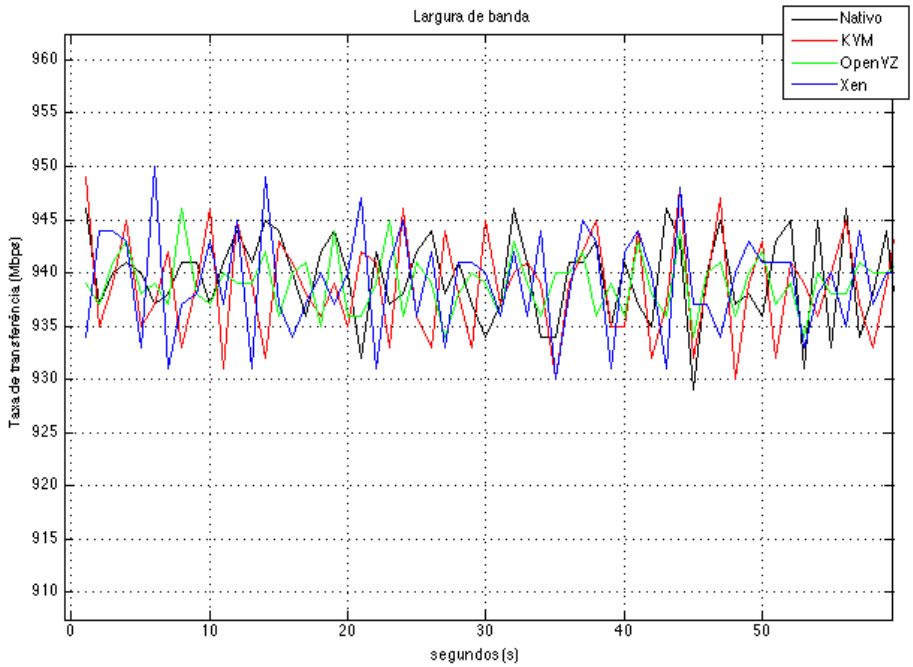
Através dos resultados ilustrados na Figura 21 podemos observar que o tempo de execução do teste para o OpenVZ foi menor, seguido de perto pelo Xen. A plataforma KVM obteve o maior tempo de execução do teste.

Estes testes ajudam a compreender e comprovar o melhor desempenho do *cluster* Hadoop implantado na nuvem que utiliza OpenVZ como virtualização durante o processamento de tarefas que exigiam intensamente os processadores das máquinas, como o TeraSort, MRBench e Pi.

5.9.4 Memória

A virtualização do acesso à memória foi medida com o software STREAM, que calcula a largura de banda estável da memória RAM durante a cópia de um conjunto de dados e realizações de sistemas lineares muito maiores que o cache de memória do sistema.

Figura 19 – Largura de banda alcançada em cada plataforma



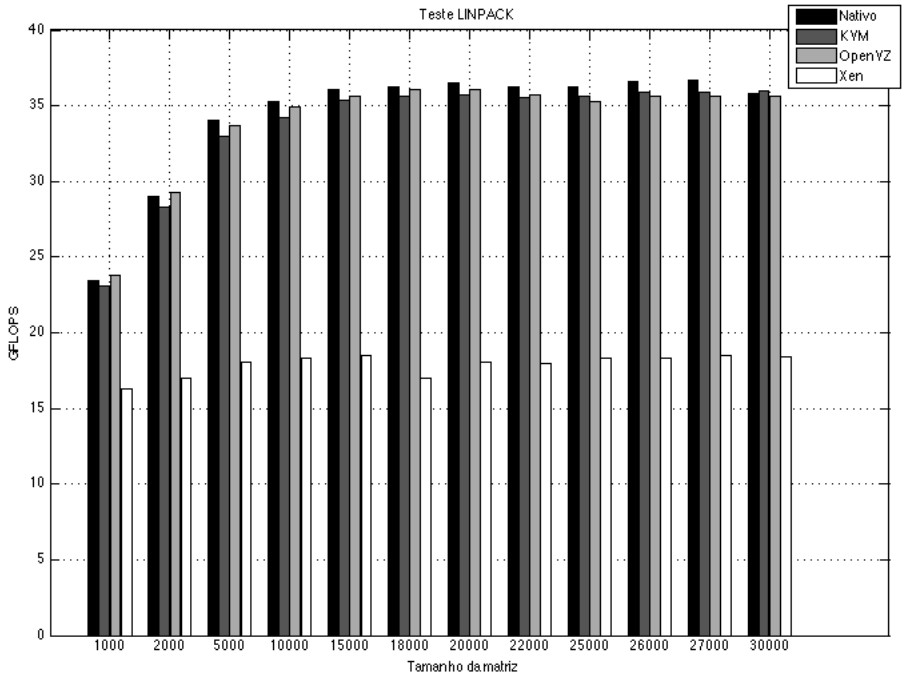
Fonte: Elaborado pelo autor.

A Figura 22 descreve o resultado dos testes STREAM e através dele pode-se observar que a largura de banda de memória no OpenVZ é bastante similar àquela alcançada na máquina nativa. Ao contrário dos testes anteriores, Xen alcançou boas taxas de acesso à memória com resultados próximos aos alcançados pelo OpenVZ. O KVM alcançou taxas de largura de banda 30% menores em comparação à máquina física.

5.10 Conclusão

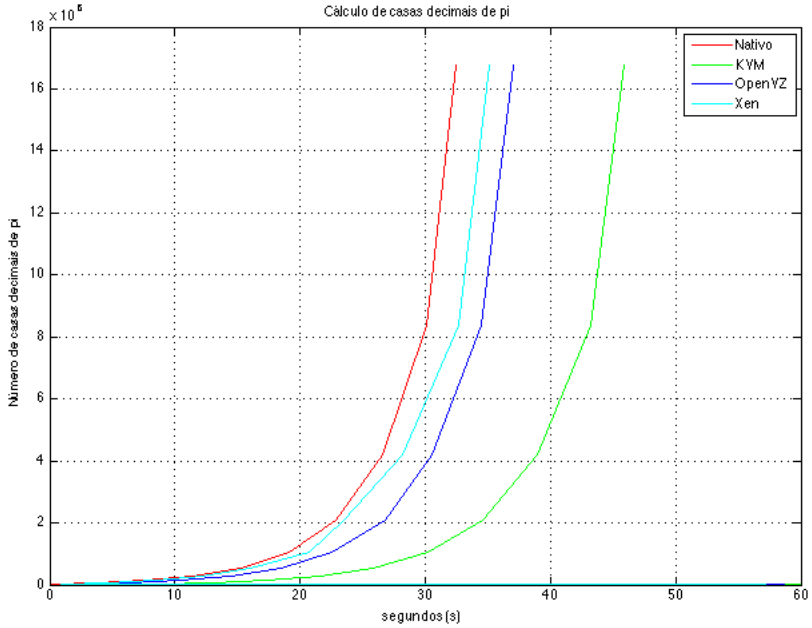
Este capítulo descreve o cenário experimental, metodologia utilizada, os tipos de testes realizados e os resultados obtidos. Tais resultados foram agrupados em duas categorias: macro-testes e micro-testes. O objetivo dessa metodologia foi analisar o desempenho das plataformas virtualizadas durante a intensa utilização de vários recursos simultâneos. Isso permitiu uma melhor análise dos resultados pois através dos micro-testes foi possível determinar que o baixo rendimento de alguns testes que utilizam vários recursos, como o TestDFSIO, foi ocasionado pelo baixo desempenho da virtualização de recursos específicos, como o de escrita de grandes arquivos em disco do OpenVZ.

Figura 20 – Poder de computação alcançado no teste LINPACK



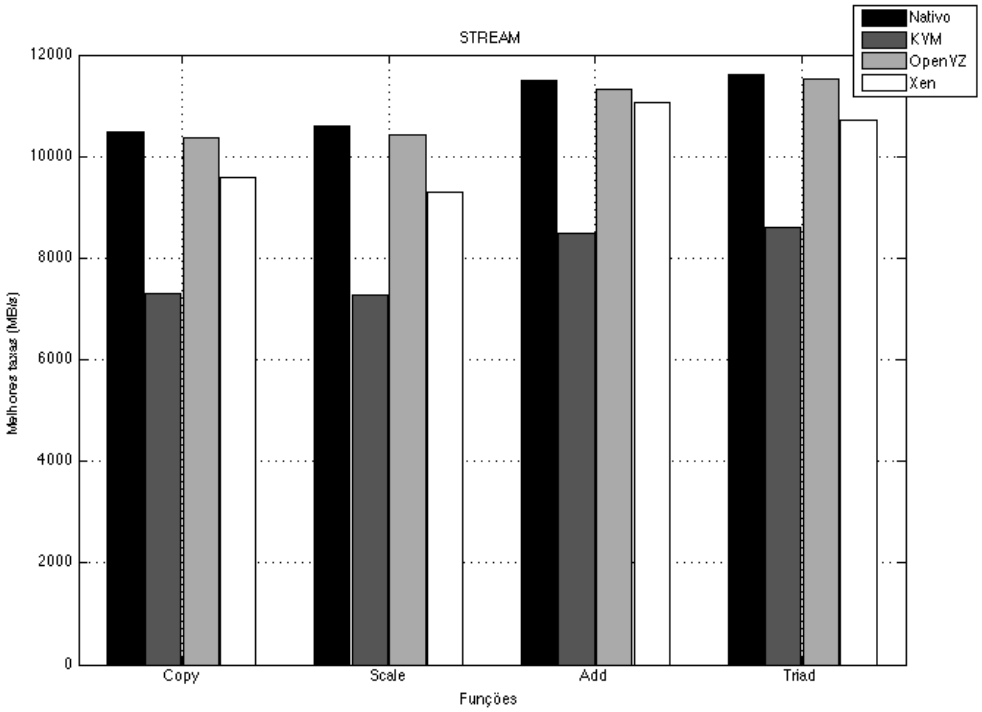
Fonte: Elaborado pelo autor.

Figura 21 – Tempo de execução do cálculo de casas de Pi no utilitário Systester



Fonte: Elaborado pelo autor.

Figura 22 – Largura de banda de acesso à memória no teste STREAM



Fonte: Elaborado pelo autor.

6 CONCLUSÃO E PERSPECTIVAS

As plataformas de virtualização KVM e OpenVZ são soluções que utilizam diferentes abordagens de virtualização e aplicações que realizam uso intensivo de recursos, como o Hadoop, necessitam de baixa sobrecarga de virtualização para serem capazes de tirar proveito das vantagens que sistemas virtualizados e em nuvem propiciam, como melhor utilização do *hardware*, consolidação de servidores, melhor gerência e facilidades de implantação de novos sistemas.

De acordo com nossos testes, o *cluster* Hadoop implantado na nuvem OpenNebula que utiliza OpenVZ para virtualização de recursos obteve melhores médias que a nuvem utilizando o KVM, demonstrando a menor sobrecarga de virtualização de sistemas operacionais. OpenVZ alcançou menores tempos de execução e maior poder de computação nos testes de stress de *CPU*, melhores taxas de acesso à memória e leitura de arquivos em disco. O KVM , por sua vez, obteve melhor desempenho em testes de escrita de grandes arquivos em disco.

Através da utilização do OpenVZ, um *cluster* virtualizado Hadoop pode atingir um melhor desempenho na execução de tarefas que realizem intensa utilização de *CPU*, rede e leitura em disco. Tarefas que realizem intensa escrita em disco devem ser analisadas e utilizadas com cautela, ou mesmo executadas nativamente.

O bom desempenho geral do OpenVZ aliado aos recursos trazidos pela nuvem OpenNebula tornam essa suíte de aplicativos uma plataforma de computação eficiente para *clusters* Hadoop MapReduce.

Em trabalhos futuros, planeja-se estudar o isolamento das plataformas de virtualização e o desempenho das mesmas para sistemas de videoconferência virtualizados.

REFERÊNCIAS

APACHE. **Hadoop Wiki**. 2015. Disponível em: <http://wiki.apache.org/hadoop/WordCount>. Acesso em: 15 maio 2015.

ARMBRUST, M.; FOX, A.; GRIFFITH, R.; JOSEPH, A. D.; KATZ, R. H.; KONWINSKI, A.; LEE, G.; PATTERSON, D. A.; RABKIN, A.; STOICA, I.; ZAHARIA, M. **Above the Clouds: A Berkeley View of Cloud Computing**. [S.l.], 2009. Disponível em: <http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html>.

BAUN, C.; KUNZE, M.; NIMIS, J.; TAI, S. **Cloud Computing - Web-Based Dynamic IT Services.** Springer, 2011. I-IX, 1-97 p. ISBN 978-3-642-20916-1. Disponível em: <http://dx.doi.org/10.1007/978-3-642-20917-8>.

C12LABS. **OpenNebula Extends Support to Deltacloud APIs**. 2015. Disponível em: <http://virtualization.info/en/news/2010/06/opennebula-extends-support-to-deltacloud-apis.html>. Acesso em: 15 maio 2015.

CHE, J.; YU, Y.; SHI, C.; LIN, W. A synthetical performance evaluation of openvz, xen and kvm. In: **Services Computing Conference (APSCC), 2010 IEEE Asia-Pacific**. [s.n.], 2010. p. 587–594. Disponível em: <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5708625>.

COELHO, F. A.; CALZAVARA, G. S.; LUCIA, R. D. **Virtualizacao - VMWare e Xen**. 2009. Disponível em: <www.gta.ufrj.br/grad/08_1/virtual/artigo.pdf>.

COUTINHO, E. F. **Fole: Um Framework Conceitual para Avaliação de Desempenho da Elasticidade em Ambientes de Computação em Nuvem**. Tese (Doutorado) — Universidade Federal do Ceará, 2014.

DESAI, A.; OZA, R.; SHARMA, P.; PATEL, B. Hypervisor: A survey on concepts and taxonomy. **International Journal of Innovative Technology and Exploring Engineering (IJITEE)**, v. 2, n. 3, p. 222–225, 2013.

EUCALYPTUS. **Eucalyptus Cloud Computing Architecture**. 2014. Disponível em: <https://www.eucalyptus.com/eucalyptus-cloud/iaas/architecture>. Acesso em: 15 maio 2015.

FOLCH, A. **Interface development for Eucalyptus based cloud**. Dissertação (Mestrado) — Vilnius Gediminas Techinical University, 2011.

GALVIN, P. B. **VMware vSphere Vs. Microsoft Hyper-V: A Technical Analysis**. 2009. 32 p. White Paper.

GIACOMO, D.; BRUNZEL, T. **Evaluating Cloud Computing - How it Differs to Traditional IT Outsourcing**. Dissertação (Mestrado) — Jönköping International Business School, 2010.

GOLDBERG, R. P. **Architetural Principles for Virtual Computer Systems**. Tese (Doutorado) — Harvard University, 1973.

GUTENBERG. **Projeto Gutenberg**. 2015. Disponível em: <https://www.gutenberg.org/wiki/PT_Principal>. Acesso em: 15 maio 2015.

IBM. **Extend your secure development process to the cloud and big data**. 2015. Disponível em: <http://www.ibm.com/developerworks/cloud/library/cl-extenddevtocloudbigdata/>. Acesso em: 15 maio 2015.

IBRAHIM, S.; JIN, H.; LU, L.; QI, L.; WU, S.; SHI, X. Evaluating mapreduce on virtual machines: The hadoop case. In: **Cloud Computing**. Springer Berlin Heidelberg, 2009. v. 5931, p. 519–528. ISBN 978-3-642-10664-4. Disponível em: <http://dx.doi.org/10.1007/978-3-642-10665-1_47>.

JACOB, J. P.; BASU, A. Performance analysis of hadoop map reduce on eucalyptus private cloud. **International Journal of Computer Applications**, v. 79, n. 17, p. 10–13, Outubro 2013. Full text available.

KANG, H.; CHEN, Y.; WONG, J. L.; SION, R.; WU, J. Enhancement of xen's scheduler for mapreduce workloads. In: **Proceedings of the 20th International Symposium on High Performance Distributed Computing**. New York, NY, USA: ACM, 2011. (HPDC '11), p. 251–262. ISBN 978-1-4503-0552-5. Disponível em: <http://doi.acm.org/10.1145/1996130.1996164>.

KLOSS, F. C. **Motor de Transformações Baseado em MapReduce**. Dissertação (Mestrado) — Universidade Federal do Paraná, 2013.

KVM. **Main Page - KVM**. Disponível em: <http://www.linux-kvm.org/page/Main_Page>. Acesso em: 15 maio 2015.

LIMA, A. **Exemplo do Hadoop Vistoriador de Pi no HDInsight**. 2014. Disponível em: <https://azure.microsoft.com/pt-br/documentation/articles/hdinsight-sample-pi-estimator/>. Acesso em: 15 maio 2015.

MAGNUS, J.; OPSAHL, G. **Open-Source Virtualization. Functionality and Performance of Qemu/KVM, Xen, Libvirt and VirtualBox**. Dissertação (Mestrado) — University of Oslo, 2013. Disponível em: <https://www.duo.uio.no/bitstream/handle/10852/37427/1/Opsahl_Master.pdf>.

MATHER, T.; KUMARASWAMY, S.; LATIF, S. **Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance**. [S.l.]: O'Reilly Media, Inc., 2009. ISBN 0596802765, 9780596802769.

MELL, P.; GRANCE, T. **The NIST Definition of Cloud Computing**. [S.l.], 2011.

MELO, C. A.; ARCOVERDE, D. F.; MORAES Éfrem R. A.; PIMENTEL, J. H. C.; FREITAS, R. Q. Software como serviço: Um modelo de negócio emergente. **Centro de Informática – Universidade Federal de Pernambuco (UFPE)**, p. 9, 2007. Disponível em: <http://www.cin.ufpe.br/~jhcp/publica/jhcp-saas.pdf>.

MUDA, K. **A Strategy for Virtual Machine Migration Based on Resource Utilization**. Dissertação (Mestrado) — Keio University, 2010.

NILSSON, J. **Hadoop MapReduce in Eucalyptus Private Cloud**. 2011. Bachelor Thesis, Umea University.

OWOPETU, O. O. **Private Cloud Implementation and Security Using Eucalyptus and Xen Frameworks**. 2013. Bachelor Thesis, Turku University of Applied Sciences.

PADALA, P.; ZHU, X.; WANG, Z.; SINGHAL, S.; SHIN, K. G.; PADALA, P.; ZHU, X.; WANG, Z.; SINGHAL, S.; SHIN, K. G. **Performance evaluation of virtualization technologies for server consolidation**. [S.l.], 2007.

PROTTI, D. J. **Linux KVM as a Learning Tool**. 2009. Disponível em: <http://www.linuxjournal.com/magazine/linux-kvm-learning-tool>. Acesso em: 15 maio 2015.

REGOLA, N.; DUCOM, J. C. Recommendations for virtualization technologies in high performance computing. In: **Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on**. [s.n.], 2010. p. 409–416. Disponível em: <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5708479>.

RUITER, T. A. **A Workload Model for MapReduce**. Dissertação (Mestrado) — Delft University of Technology, 2012.

SEMPOLINSKI, P.; THAIN, D. A comparison and critique of eucalyptus, opennebula and nimbus. In: **2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom)**. [s.n.]. Disponível em: <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5708480>.

VASCONCELOS, P. R. M.; FREITAS, G. A. A. Análise de desempenho de um ambiente virtualizado com kvm. In: **XXXII Encontro de Iniciação Científica - UFC**. Sobral, Brasil: [s.n.], 2013.

VASCONCELOS, P. R. M.; FREITAS, G. A. A. Performance analysis of hadoop mapreduce on an opennebula cloud with kvm and openvz virtualizations. In: **Internet Technology and Secured Transactions (ICITST), 2014 9th International Conference for**. Londres, Reino Unido: [s.n.], 2014a. p. 471–476. Disponível em: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=7038858>.

VASCONCELOS, P. R. M.; FREITAS, G. A. A. Evaluating virtualization for hadoop mapreduce on an opennebula cloud. **International Journal Multimedia and Image Processing (IJMIP)**, v. 4, n. 1/2/3/4, p. 50–60, March/June/September/December 2014b. ISSN 2042 4647. Disponível em: <http://infonomics-society.org/wp-content/uploads/ijmip/published-papers/volume-4-2014/Evaluating-Virtualization-for-Hadoop-MapReduce-on-an-OpenNebula-Cloud.pdf>.

VASCONCELOS, P. R. M.; FREITAS, G. A. A. Uma arquitetura de balanceamento de carga web escalável para nuvens eucalyptus. In: **Anais do VII ENUCOMP 2014**. Parnaíba, Piauí: [s.n.], 2014c. p. 107–119. ISBN 978-85-8320-073-4. Disponível em: <http://www.enucomp.com.br/2014/editais/AnaisVIIENUCOMP2014.pdf>.

VIRTUALBOX. **Oracle VM VirtualBox**. Disponível em: <https://www.virtualbox.org/>. Acesso em: 15 maio 2015.

VMWARE. **VMware Virtualization for Desktop & Server, Application, Public & Hybrid Clouds**. Disponível em: <http://www.vmware.com/>. Acesso em: 15 maio 2015.

WHITE, T. **Hadoop: The Definitive Guide**. 1. ed. [S.l.]: O'Reilly, 2012.

WOTTRICH, R.; GENEZ, T.; PEREIRA, W. Uma visão geral sobral sistemas virtualizados. **Minicurso de arquitetura de computadores - UNICAMP**, p. 8, 2012. Disponível em: <http://www.ic.unicamp.br/~ducatte/mo401/1s2012/T2/G04-115168-t2.pdf>. Acesso em: 15 maio 2015.

XAVIER, M.; NEVES, M.; ROSSI, F.; FERRETO, T.; LANGE, T.; ROSE, C. D. Performance evaluation of container-based virtualization for high performance computing environments.

In: **Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on**. [s.n.], 2013. p. 233–240. ISSN 1066-6192. Disponível em: <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6787290>.

XAVIER, M. G.; NEVES, M. V.; ROSE, C. A. F. A performance comparison of container-based virtualization systems for mapreduce clusters. In: **Parallel, Distributed and Network-Based Processing (PDP), 2014 22nd Euromicro International Conference on**. [s.n.], 2014. p. 299–306. ISSN 1066-6192. Disponível em: <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6787290>.

XEN. **The Xen Project, the powerful open source industry standard for virtualization**. Disponível em: <http://www.xenproject.org/>. Acesso em: 15 maio 2015.

XEN. **How Does Xen Work?** 2009. 10 p. White Paper. Disponível em: <http://www.xen.org/files/Marketing/HowDoesXenWork.pdf>.

YU, Y. **OS-Level Virtualization and Its Applications**. Tese (Doutorado) — Stony Brook University, 2007.

ZAHARIA, M.; BORTHAKUR, D.; SARMA, J. S.; ELMELEEGY, K.; SHENKER, S.; STOICA, I. Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling. In: **Proceedings of the 5th European Conference on Computer Systems**. New York, NY, USA: ACM, 2010. (EuroSys '10), p. 265–278. ISBN 978-1-60558-577-2. Disponível em: <http://doi.acm.org/10.1145/1755913.1755940>.

APÊNDICE A – PRODUÇÃO CIENTÍFICA RESULTANTE DESTE TRABALHO

A.1 Artigos publicados em Conferências Internacionais

VASCONCELOS, P. R. M.; FREITAS, G. A. A. Performance analysis of hadoop mapreduce on an opennebula cloud with kvm and openvz virtualizations. In: **Internet Technology and Secured Transactions (ICITST), 2014 9th International Conference for**. Londres, Reino Unido: [s.n.], 2014a. p. 471–476. Disponível em: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=7038858>.

A.2 Artigos publicados em Periódicos Internacionais

VASCONCELOS, P. R. M.; FREITAS, G. A. A. Evaluating virtualization for hadoop mapreduce on an opennebula cloud. **International Journal Multimedia and Image Processing (IJMIP)**, v. 4, n. 1/2/3/4, p. 50–60, March/June/September/December 2014b. ISSN 2042 4647. Disponível em: <http://infonomics-society.org/wp-content/uploads/ijmip/published-papers/volume-4-2014/Evaluating-Virtualization-for-Hadoop-MapReduce-on-an-OpenNebula-Cloud.pdf>.

A.3 Artigos publicados em Conferências Regionais

VASCONCELOS, P. R. M.; FREITAS, G. A. A. Uma arquitetura de balanceamento de carga web escalável para nuvens eucalyptus. In: **Anais do VII ENUCOMP 2014**. Parnaíba, Piauí: [s.n.], 2014c. p. 107–119. ISBN 978-85-8320-073-4. Disponível em: <http://www.enucomp.com.br/2014/editais/AnaisVIIENUCOMP2014.pdf>.

A.4 Artigos publicados em Eventos Locais

VASCONCELOS, P. R. M.; FREITAS, G. A. A. Análise de desempenho de um ambiente virtualizado com kvm. In: **XXXII Encontro de Iniciação Científica - UFC**. Sobral, Brasil: [s.n.], 2013.