Reference: Maron, C. A. F; Griebler, D. *Avaliando o Desempenho das Ferramentas de Nuvem Privada OpenStack e OpenNebula.* Laboratory of Advanced Researches on Cloud Computing (LARCC), Technical Report, 2015.

Relatório Técnico de Pesquisa (Atividades de 2014)

Nome do Projeto:

High Performance in Cloud (HiPerfCloud)

Avaliação do Desempenho de
Ambientes de Computação em Nuvem

RT1: Avaliando o Desempenho das Ferramentas de Nuvem Privada OpenStack e OpenNebula

ID do Documento:	LARCC-HiPerfCloud-RT1
Versão:	1.2
Autores:	Dalvan Griebler, Carlos A. F. Maron
Objetivo:	Avaliar o Desempenho das Ferramentas OpenStack e OpenNebula
Tarefa:	Medir o Desempenho no Isolamento de Recursos, Aplicações Paralelas e Aplicações Corporativas
Hardware:	2 clusters com 4 nodos cada, onde cada máquina possui um processador (Intel Core i5 650 3.20 GHz) com 4 GB de memória (DDR3 1333 MHz) e 500 GB de disco (Sata II), operando em um rede Gigabit
Ambiente:	Sistema Operacional (Ubuntu Server 14.04), Virtualizador (KVM), OpenStack (vers. Icehouse), OpenNebula (vers. 4.8.0), Benchmarks de Isolamento (Iperf, IOzone, STREAM e LINPACK), Aplicações Paralelas (NPB-MPI e NPB-OMP) e Aplicações Corporativas (FileBench)
Softwares:	SPSS (Análise Estatística), GNUPlot (Gráficos), Latex (Documentos)

Tarefa	Responsável	Instituição	Papel	Data
Criado por:	Dalvan Griebler	SETREM	Coordenador	27/12/2014
Editado por:	Carlos A. F. Maron	SETREM	Pesquisador	09/02/2015
Revisado por:	Leonardo Moerschberger	ABASE	Colaborador	20/02/2015
	Vera Lúcia Benedetti	SETREM	Colaboradora	20/02/2015
	Fauzi Shubeita	SETREM	Colaborador	20/02/2015
Aprovado por:	Dalvan Griebler	SETREM	Coordenador	06/03/2015
	Ildo Corso	ABASE	Colaborador	06/03/2015
Publicado:	LARCC	SETREM	Laboratório de Pesquisa	20/05/2015
Editado por:	Dalvan Griebler	SETREM	Coordenador	09/01/2016

Log de Mudanças do Documento

Versão	Autores	Instituição	Mudança	Data
1	Dalvan Griebler	SETREM	Versão Inicial	27/12/2014
1	Carlos A. F. Maron	SETREM	Envio Para Revisão	09/02/2015
1	Carlos A. F. Maron	SETREM	Envio Para Aprovação	27/02/2015
1	Carlos A. F. Maron, Dalvan Griebler, Ildo Corso	SETREM	Versão Final	05/05/2015
1.2	Dalvan Griebler	SETREM	Atualização do Layout e Informações do Projeto	09/01/2016

Lista de colaboradores internos e externos

A baixo é listado (em ordem alfabética) as pessoas que fizeram contribuições para este relatório técnico:

- Adriano Vogel (SETREM)
- Carlos A. F. Maron (SETREM)
- Claudio Schepke (UNIPAMPA)
- Dalvan Griebler (SETREM)

Resumo Geral

O objetivo do **Projeto HiPerfCloud (*High Performance in Cloud*)** é essencialmente a avaliação de desempenho das ferramentas de administração de IaaS (*Infrastructure as a Service*) e analisar o impacto que as mesmas podem causar em diferentes cenários (isolamento de recursos e, aplicações corporativas e de alto desempenho/paralelas). Visto que, as ferramentas possuem uma série de características particulares [33, 17], surgiu a hipótese de que podem ter diferentes desempenhos e também impactar sobre as aplicações [20, 21]. Este documento relata os experimentos e resultados iniciais em direção à avaliação de desempenho nas ferramentas OpenStack [29] e OpenNebula [28].

Contexto do Relatório

Este documento é o primeiro Relatório Técnico *RT1: Avaliando o Desempenho das Ferramentas de Nuvem Privada OpenStack e OpenNebula* referente ao **Projeto HiPerfCloud**, para apresentar os resultados iniciais dos testes nos cenários de: isolamento de recursos (memória RAM, processador, armazenamento e rede), aplicações de alto desempenho e aplicações corporativas. Para tal avaliação, foram utilizados *benchmarks* que simulam cargas de trabalho reais e específicas, permitindo uma avaliação precisa do desempenho nos cenários objetivados.

Estrutura do Relatório

Neste documento realiza-se inicialmente, uma apresentação geral do relatório técnico. Posteriormente, contextualiza-se a computação em nuvem, as ferramentas de administração OpenStack e OpenNebula e os cenários de avaliação. Em seguida, relata-se brevemente sobre a implantação dos ambientes de testes. Ao final, são discutidos os resultados alcançados.

Sumário

1. Introdução

Neste capítulo, será abordado uma visão geral deste documento, mostrando um breve resumo sobre os capítulos seguintes e terminologias utilizadas durante o trabalho.

1.1 Visão Geral

O presente documento descreve a avaliação de desempenho das ferramentas OpenStack e OpenNebula realizado no Projeto HiPerfCloud. Do mesmo, o foco está em:

- Realizar experimentos nos cenários de isolamento de recursos, aplicações paralelas e aplicações corporativas.
- Avaliar e comparar o desempenho entre as duas ferramentas e discutir os resultados iniciais.
- Realizar testes estatísticos para verificar se os desempenhos foram significativamente diferentes.
- Apontar características das ferramentas que influenciam no desempenho.

1.2 Terminologia

- **Isolamento de Recursos:** Representa os recursos de processamento: Memória RAM, armazenamento, rede e processador.
- **Aplicações Paralelas:** Área da computação de alto desempenho.
- **Aplicações Corporativas:** *Softwares* ou serviços comumente utilizado por empresas.
- **Benchmark:** Programa para teste específico de determinado recurso ou serviço.
- **Cluster**: Conjunto de computadores interligados por uma rede somando recursos.
- **OpenStack**: Ferramenta de administração de nuven IaaS.
- **OpenNebula**: Ferramenta de administração de nuven IaaS
- **Iperf**: *Benchmark* para avaliação de rede.
- **IOzone**: *Benchmark* para avaliação de unidades de armazenamento e sistemas de arquivos.
- **STREAM**: *Benchmark* para avaliação da memória RAM.
- **LINPACK**:*Benchmark* para avaliação do processador.
- **NPB-MPI**: *Benchmark* composto por diferentes *kernels* de simulação de aplicações paralelas com bibliotecas MPI.
- **NPB-OMP**: *Benchmark* composto por diferentes *kernels* de simulação de aplicações paralelas com bibliotecas OMP
- **Filebench**: *Benchmark* composto por diferentes *kernels* de simulação de aplicações corporativas.

1.3 Estrutura deste Documento

Este documento está organizado em cinco capítulos:

- Capítulo 1: Introduz as informações sobre este documento.
- Capítulo 2: Nesta seção, encontra-se o referencial sobre a computação em nuvem, os tipos e os ambientes de implantação, alguns estudos de caso da computação em nuvem. O capítulo tem como abordagem principal relatos sobre as ferramentas OpenStack e OpenNebula utilizadas na pesquisa, as características dos *benchmarks* de isolamento de recursos e aplicações paralelas e corporativas.
- Capítulo 3: Descreve detalhes para a preparação do ambiente computacional, descrevendo a configuração das máquinas físicas, das ferramentas OpenStack e OpenNebula, e do ambiente na nuvem.
- Capítulo 4: Apresenta a metodologia utilizada para realizar os testes, os resultados das execuções dos *benchmarks* em formas de gráficos como também, tabelas contendo a análise estatística destes resultados.
- Capítulo 5: Conclusão dos resultados e trabalhos futuros.

2. Contextualização

O presente capítulo contextualiza a pesquisa realizada. Trazendo os conceitos sobre a computação em nuvem, um *background* sobre as tecnologias. Como também descreve as diversas ferramentas utilizadas as ferramentas utilizadas, tanto para montagem da infraestrutura de nuvem e também para a avaliação do mesmo. Conta com um breve relato sobre a motivação da escolha deste tema.

2.1 Computação em Nuvem: *Background* e Motivação

A computação em nuvem é uma tema em destaque no meio acadêmico e comercial. Pois oferece um conjunto de tecnologias formando infraestruturas de nuvem que possibilita aos usuários utilizar diversos tipos de recursos, como: *softwares* (SaaS - *Software as a Service*), plataformas de desenvolvimentos (PaaS - *Plataform as a Service*), e também infraestruturas de processamento (IaaS - *Infraesctuture as a Service*) [19].

Em geral, nuvens podem ser uma grande infraestrutura computacional. Portante são encontradas em ambientes público (provedores de serviços), privadas (empresas particulares ou instituições), híbridas e comunitárias. Os tipos de infraestruturas e seus ambientes são algumas definições básicas da computação nuvem [19].

As características de uma nuvem tornam-á um facilitador de resultados [1]. Todo o seu conjunto computacional está diretamente ligado a tecnologias de virtualização e permite que serviços de infraestrutura sejam alocados em nuvem. Desta forma, contribuí para uma otimização na utilização de recursos, pois várias instânicas podem ser alocadas em um mesmo *host* físico, permitindo que diversos benefícios sejam repassados aos usuários (economia de energia, escalabilidade e flexibilidade).

A nuvem como um todo, necessita de uma ferramenta que gerencie de forma única diversos dos seus recursos. Podemos observar no trabalho de [33], que as ferramentas utilizadas para administração de nuvem, possuem diferentes características de implantação e de estrutura. Nesse sentido existe uma dúvida sobre as influências destas ferramentas no processamento em nuvem. Além disso, tem-se a camada de virtualização, onde trabalhos de [36, 31, 16, 27, 9, 32, 25], no geral avaliam o impacto do desempenho nas aplicações paralelas e isolamento de recursos, levando em conta a tecnologia de virtualização em ambientes de nuvem, alguns avaliam somente os virtualizadores de forma individual.

Contudo, podemos dizer que a nuvem será o futuro de infraestruturas e *softwares*, pois o mercado se encontra favorável para isso. Estima-se que até 2019, a nuvem privada será adotada por 79% das empresas brasileiras, as quais buscam montar sua própria infraestrutura [35]. Segundo [7], justifica o fato de que a demanda de *datacenter* no Brasil cresce na ordem dos 70% ao ano.

Em uma visão comercial, temos várias empresas provedores de serviços de computação em nuvem. Alguns exemplos são: Google, Amazon, Microsoft [8, 2, 26]. Atualmente empresas particulares estão buscando alocar parte de seus serviços em infraestruturas de nuvens públicas, como podemos perceber em alguns estudos de caso a seguir.

A empresa Google é famosa por seus serviços, como Gmail, Youtube, entre outros. Para oferecer tais serviços aos usuários, a empresa necessita de uma grande capacidade computacional para atender a demanda de utilização destes, e tal capacidade que permite à empresa inclusive a possibilidade de oferecer serviços como IaaS, PaaS.

É o caso da empresa Rovio, criadora do jogo "Angry Birds", que utilizou a plataforma de desenvolvimento oferecida pela Google (PaaS), e através dela desenvolveu e hospedou a

aplicação nos servidores da Google [14].

A plataforma IaaS do Google é voltada também para alto desempenho, e neste caso, vemos o exemplo [14], um centro de pesquisa especializado em mapeamento das alterações genéticas de diversos tipos de câncer. Uma pesquisa que necessita uma grande capacidade de servidores e processamento, e assim o centro de pesquisa hospedou em nuvem suas aplicações, para realizar um mapeamento genético de células cancerígenas.

No Brasil, um exemplo da utilização e implantação de nuvens públicas, é a Universidade de São Paulo - USP. Com um investimento de mais de 200 milhões de reais, para montar uma estrutura completa de *data center* e substituição de máquinas dos usuários. O principal objetivo deste investimento foi a resolução de alguns problemas que a universidade vinha enfrentando, tais como: a falta de agilidade e flexibilidade, gastos desnecessários com infraestrutura, ausência e falhas no alinhamento entre as atualizações em *softwares* do parque de máquinas, e a inexistência de escalabilidade. Foram os principais fatores que determinaram esse tipo de escolha, permitindo à USP um avanço em diversas linhas de pesquisas [12].

Com alguns estudos de caso, nos reflete que o modelo (IaaS) está na base de todos os tipos de nuvens, pois é ele que provê todos os recursos computacionais para as camadas acima deste modelo. E comercialmente falando, a nuvem se torna sim uma infraestrutura atraente.

Porém, é importante observar que a camada de virtualização presente nas infraestruturas de nuvens deixaram-nas em meio a um paradigma sobre seu desempenho. Acredita-se que devido as diferentes características das ferramentas de administração de nuvens IaaS, o desempenho das aplicações pode se sujeitar à alguma influência, positiva ou negativa.

Podemos observar no trabalho [23, 21, 22] que possui resultados iniciais, do desempenho em nuvem utilizando as mesmas tecnologias de virtualização em diferentes ferramentas de administração de nuvem. Mostrando que desempenho pode sofrer diferenças significativas entre ambas.

2.2 Ferramentas de Administração de Nuvem

Uma infraestrutura de nuvem pode ser um aglomerado de recursos computacionais. As ferramentas de administração de nuvem são usadas para tornar disponível ao usuário de forma dinâmica e de fácil gerenciamento todos esses recursos. Contudo, a quantidade de ferramentas existentes para este fim, torna difícil a escolha da ferramenta correta, e que melhor irá atender as necessidades.

Um estudo realizado [33], teve como objetivo principal analisar e comparar as funcionalidades descritas na literatura das principais ferramentas de administração de nuvem IaaS existentes. Através deste estudo, o resultado do trabalho foi a classificação das ferramentas com as características mais completas.

O estudo mostrou que as ferramentas OpenStack e OpenNebula foram as melhores na análise. E a partir disto, o trabalho de conclusão de curso [20] implantou estas duas ferramentas, e realizou uma análise voltada para o desempenho destes ambientes.

A partir do estudo, com a prática da implantação, que são ferramentas bem diferentes no que diz respeito à estrutura de componentes. OpenStack é composto por um conjunto complexo de componentes que desempenham funções distintas e específicas na infraestrutura. Já o OpenNebula se mantém ao contrário, com uma configuração mais simples e direta, sem muitos arquivos para editar durante sua configuração.

Nas seções seguintes, serão abordados algumas características destas ferramentas.

2.2.1 OpenNebula

OpenNebula [28] é uma ferramenta desenvolvida para administração de nuvens públicas ou privadas, oferecendo serviços de infraestrutura. Permite aos usuários, um ambiente para alocações de servidores, redes, e roteadores virtuais. Na implantação da ferramenta OpenNebula, é necessário definir um *host* como gestor principal dos serviços e recursos. Nele, são alocados os principais componente da ferramenta, como: Oned, Sunstone e mm_sched [34].

- Oned: É o principal componente da estrutura. Sua função está diretamente ligada ao controle dos nodos, redes virtuais, instâncias virtuais, usuários, grupos de usuários e *datastorage*.

- mm_sched: *Match-making Scheduler*, responsável em criar políticas para a alocação das instâncias virtuais na nuvem. É um serviço isolado do ONED. No momento da requisição de uma instância, o serviço realiza uma análise dos recursos e dos *hosts* com menos carga, após isso, inicia uma conexão com o ONED, para que a instância seja devidamente alocada.

- Sunstone: É interface de gerenciamento principal dos serviços OpenNebula. Com ela, administradores e usuários gerenciam através de uma interface WEB, todos os recursos existentes e alocados em uma nuvem.

Como a computação em nuvem está diretamente associada a tecnologias de virtualização. OpenNebula trabalha em conjunto com os virtualizadores Xen Server, KVM e VMware. Para criação das redes e interfaces virtuais, tem compatibilidade com os serviços, OpenvSwitch, Ebtables, 802.1Q, e *dummy* [34].

A ferramenta OpenNebula tem uma abordagem diferenciada das redes e interfaces virtuais. No caso do OpenvSwitch, tem a função independente de criar/gerenciar redes e interfaces das instâncias. Ebtables, aplica apenas um isolamento de rede por meio de máscara 255.255.255.0, usando regras ebtables aplicadas às interfaces *bridges*. 802.1Q, usa os padrões para criação de VLANs, que é demarcação por identificadores específicos criando uma camada de ligação entre cada rede virtual. *Dummy*, é um *driver* básico, desenvolvido pelo próprio OpenNebula, aonde não exerce nenhum tipo específico de operação de rede, e durante seu funcionamento, as regras de *firewall* são ignoradas [34]

No quesito armazenamento, a ferramenta possui os métodos de armazenamento local ou distribuído. Neste caso, ao instanciar uma máquina virtual (VM), a unidade de armazenamento pode estar no *host* em que a VM está executando, ou em uma unidade compartilhada pelo *front end*, usando o serviço NFS.

No virtualizador KVM, os formatos de discos utilizado podem ser o QCOW2, ISO ou RAM. O OpenNebula ainda é compatível com os seguintes formatos de discos: Ceph, Gluster, FS LVM, LVM, VMware VMFS [28].

2.2.2 OpenStack

A ferramenta OpenStack [29] é uma ferramenta *open source* voltada para oferecer serviços de infraestrutura em nuvem, projetada para ambientes privados e públicos. A versão com o codinome IceHouse (2014.1.3), lançada em outubro de 2014 possui os principais componentes em sua estrutura:

- **Horizon** - Painel de utilização da ferramenta. Possui uma interface gráfica e pode ser acessado através do navegador.

- **Nova** - Serviço de computação. Gerencia as instâncias virtuais. Tem comunicação direta com as ferramentas de virtualização.
- **Neutron** - Serviço voltado para redes na nuvem. Através dele, é possível criar redes, sub-redes, roteadores, e aplicar regras para controle específico no tráfego de rede.
- **Swift** - Serviço de armazenamento. Voltado para controle de armazenamento desestruturado em nuvem.
- **Cinder** - Serviço de armazenamento. Controle de armazenamento persistente. Permite criar discos em forma de blocos.
- **Keystone** - Serviço de segurança. Com ele, a ferramenta autentica e autoriza todos os serviços do OpenStack. Além ainda de exercer um controle de usuário.
- **Ceilometer** - Serviço de monitoramento. Monitora e gerencia a utilização dos recursos em nuvem para cada usuário.
- **Orchestration** - Serviço de aplicações de multi-camadas. Serviço que controla o ciclo de vida de infraestrutura e de aplicações dentro da nuvem.
- **Trove** - Serviço de banco de dados. Voltado para banco de dados relacional e não relacional em nuvem, podendo criar bancos de dados MySQL.

No OpenStack, existe essa divisão bem específica dos componentes que controlam parte dos serviços e os recursos da nuvem. Na ferramenta, são 9 os componentes macros, e alguns deles como Neutron, Nova, Cinder possui outros sub-componentes. Estes componentes são os principais responsáveis em gerenciar os recursos de uma nuvem (Rede, computação, armazenamento), pois são os componentes que estão em contato direto com as ferramentas de virtualização (Ex: KVM, LVM, OpenvSwitch).

O Nova, por meio dos sub componentes *nova-compute, nova-scheduler*, repassa ao *hypervisor*, todos os comandados que exercem as funções de gerenciamento das máquinas virtuais (criação, remoção, migração, agendamento), e *nova-conductor*, tem a função de intermediar as comunicação entre os serviços citados anteriormente e um banco de dados, salvando informações específicas das VMs. Além desses, os subcomponentes do Nova são: *nova-api, nova-api-metadata, nova-consoleauth, nova-novncproxy, nova-cert*. Através do Nova, OpenStack tem suporte aos seguintes virtualizadores: Xen Server, KVM, LXC, VMware, Hyper-V [29].

O componente Neutron, permite que em uma infraestrutura de nuvem seja implantada com serviços de *firewalls* e balanceadores de carga. Estes serviços possuem um tipo de implementação específica utilizando os recursos internos da ferramenta. Contudo, em sua implantação original, o serviço de rede é assumido através de VLANs e *iptables*, e utilizando ainda APIs para *plugins* de terceiros, como o OpenvSwitch, que permite um tunelamento mais avançado nas comunicações [29].

OpenStack tem uma abordagem complexa para implementar armazenamento em nuvem, (*Ephemeral Storage* e *Persistent Storage*). Seus dois componentes Cinder e Swift, tratam de maneira diferente os métodos de armazenamento e são responsáveis em prover o *Persisten Storage*. Em conjunto à estes componentes, estão tecnologias do tipo LVM, Ceph, Gluster, NFS, ZFS, Sheepdog [29].

2.3 Aplicações Paralelas

Como definição básica, são aplicações que são executadas em um ambiente distribuído ou paralelo. É uma aplicação que consegue tirar vantagem do paralelismo existente nas arquiteturas.

Boa parte do grande processamento envolvido em aplicações paralelas, está alocado para o meio científico. Pesquisas das mais diversas áreas necessitam de um intenso processamento de dados, e que precisam ser resolvidos em pouco tempo. Um estudo aplicado do uso de aplicações paralelas é a simulação de correntes oceânicas. Onde alguns métodos específicos como discretização de equações em grades e discretização de domínio sobre grade, são muito comuns na computação de alto desempenho. A computação gráfica, é uma área que também tira proveito de aplicações de alto desempenho. E recentemente a área comercial vem se destacando neste meio, com um processamento direcionado a altos volumes de dados, que exercem um intenso tráfego de I/O (Ex: BigDate). O que para a computação de alto desempenho, é alguns dos principais desafios [5].

A execução de aplicações paralelas são destinadas para infraestruturas com um grande poder computacional. A computação em nuvem na maioria dos casos é um aglomerado de recursos que podem ser oferecidos em formas de serviços (visto nas seções anteriores), e que também podem apresentar uma grande capacidade de processamento, semelhantes a *clusters* de alto desempenho. Pesquisadores e usuários estão com um olhar positivo para a execução de aplicações paralelas em um ambiente de nuvem, mesmo que paradigmas vinham sendo criados sobre tipos de execuções em nuvem, devido as camadas de abstrações existentes em suas infraestruturas (virtualização) e algumas possíveis interferências externas (compartilhamento de recursos). Pois, a nuvem torna seus recursos dinâmicos, e suas alocações podem ser através de demanda. Para a aplicações paralelas, é fator que pode fazer grande diferença nos tempos de execução [5].

Algumas bibliotecas de programação paralela são muito utilizadas no meio da computação de alto desempenho, como MPI e OMP.

MPI (*Message Passing Interface*) é um padrão de comunicação entre processos, é implementado em linguagem C ou Fortram. Lançado no início dos anos 90, é uma linguagem que apresenta uma grande produtividade e eficiência, pois permite que todo o processamento seja distribuído, executando aplicações em memória distribuída. ([15])

OMP (*Open Multi-Processing*) é um padrão de comunicação paralelo que permite a criação de processos compartilhando a mesma memória. Através deste padrão, processos são criados nos diversos núcleos disponíveis no ambiente [4].

Um *benchmark* que representa a utilização destas bibliotecas é o NAS (*Nasa Parallel Benchmark*). Este conjunto de pequenos *kernels* lançado pela NASA em 1992 tinha como objetivo suprir a falta de *benchmarks* para computadores altamente paralelos. Os *benchmarks* que compõem o NAS, consistem em cinco núcleos e três pseudo-aplicações, e ainda com benchmarks multi-zona, I/O, e de malhas adaptativas de grades computacionais paralelas, que se utilizam de uma dinâmica de fluídos computacionais [3]. A forma de aplicar as intensidades das cargas de trabalho são definidas por classes no momento de sua compilação. As classes disponíveis são:

- Classe S: cargas pequenas, usado para testes rápidos.
- Classe W: tamanho de cargas usados para estações de trabalhos.
- Classes A, B, C: tamanhos de testes padrões. Aumenta em torno de 4 vezes entre uma classe e outra.
- Classes D, E, F: Grandes cargas no trabalho. Aumenta em torno de 16 vezes entre cada classe.

Os 5 (cinco) *kernels* usados para os cálculos de movimentação da dinâmica de fluídos computacionais são:

- **IS** – *Integer Sort*: Ordena números inteiros usando bucket sort.

- **EP** – *Embarrassingly Parallel*: Geração independente de valores Gaussian e variáveis randômicas usando o método Polar Marsaglia.
- **CG** – *Conjugate Gradiant*: Cálculo de valores de matrizes.
- **MG** – *Multi-Grid*: Comunicação intensiva de curta e longa distância com a memória.
- **FT** – *Fast Fourier Transform*: método Transformada de Fourier, usando a comunicação todos para todos.

Complementando a suíte do NAS, as 3 pseudo-aplicações aplicam a resolução de um sistema de equações diferenciais parciais não-lineares, sendo elas:

- **BT** – *Block Tri-diagonal*: Algoritmo Bloqueio Tridiagonal.
- **SP** – *Scalar Penta-diagonal*: Algoritmo Penta-diagonal.
- **LU** – *Lower-Upper* – Algoritmo de Gauss.

A suíte do NAS também pode ser utilizada para avaliação de I/O paralelo e de movimentos de dados e computação não estruturada. Os *kernels* são:

- **UA** – *Unstructured Adaptive*: Resolução de equações em malhas adaptativas não estruturadas, acesso à memória dinamicamente e irregularmente.
- **BT-IO** – testes de diferentes técnicas de I/O paralelo.
- **DC** – *Data Cube*: Cálculos com valores comumente usados para série temporal.
- **DT** – *Data Traffic*.

2.4 Isolamento de Recursos

O conjunto que integra a capacidade computacional, basicamente é composto por memória RAM, processador, unidade de armazenamento e rede de comunicação. Estes são os princípios mais básicos para se alcançar uma capacidade de processamento. Devido a constante evolução nessa área, algumas aplicações chamadas de *benchmarks*, tem a função específica de avaliar a capacidade de processamento. Não somente de recursos físicos, como também os mais diversos tipos de serviços podem ser avaliados.

A maneira que os *benchmarks* testam a capacidade é através da aplicação de cargas de trabalhos sintéticas, criadas durante a execução, simulando algum tipo de serviço e assim utilizando os recursos necessários. Os resultados destes testes, são usados como referência da capacidade de processamento.

2.4.1 Processador (LINPACK)

LINPACK é um *benchmark* escrito em linguagem Fortran, que possui sub-rotinas que analisam e resolvem em tempo de execução simultâneo diversos sistemas de equações algébricas lineares. Atualmente o código pode ser encontrado em linguagem C.

As sub-rotinas são utilizas para resolução de matrizes quadradas de coeficientes. Entre as diversas utilizadas pelo algoritmo, algumas são usadas em cálculos estatísticos, onde são desenvolvidas para tirar proveito de tempo de execução e armazenamento.

Geralmente, qualquer matriz de coeficiente será armazenada em memória. Para o processamento destas matrizes, o LINPACK comumente utiliza duas sub-rotinas para processar os

dados contido nas matrizes [6]. Sendo assim, o o valor de entrada para a execução do LINPACK é o valor da matriz, que deverá ser compatível com o tamanho da memória RAM do equipamento.

2.4.2 Memória (STREAM)

STREAM é um benchmark simples, utilizado para medir a largura de banda de memória RAM (*Random Access Memory*). Este que foi projetado para trabalhar com um fluxo de dados maiores que o espaço disponível em memória cache. Desta forma, cria conjunto de dados que não possam ser armazenados em cache L1, L2 ou L3, e utiliza somente processamento da memória RAM. Este *benchmark* pode ser compilado com Fortram e OMP [24].

2.4.3 Rede (IPerf)

Utilizado para medir o desempenho de uma comunicação de rede, o IPerf avalia a largura de banda de uma rede utilizando o protocolo TCP (*Transmission Control Protocol*), e algumas outras características do UDP. As principais características para os testes TCP são:

- Testes da largura de banda.
- Especifíciar tamanhos de MSS/MTU (*Maximum Segment Size*/*Maximum Transmission Unit*).
- Suporte para tamanho de janelas TCP via *buffers* de *sockets*.
- Recurso multi-thread com a biblioteca pthreads.

Para o protocolo UDP (*User Datagram Protocol*), as características são:

- Criação de fluxos específicos do pacote UDP para medir a largura de banda.
- Medida de perda de pacotes.
- Medida do atraso do jitter.

2.4.4 Armazenamento (IOzone)

O *benchmark* IOzone [18] é utilizado para avaliar o desempenho de sistemas de arquivos e em unidades de armazenamento. Utiliza uma ampla variedade de testes, em diversas plataformas de sistemas operacionais. Suas principais operações são: *Read, write, re-read, re-write, read backwards, read strided, fread, fwrite, random readwrite, preadpwrite variants, aio_read, aio_write, mmap.*

Em grande parte das funções do IOzone são necessárias bibliotecas para realização das operações de armazenamento. Porém, as funções básicas como, *read, write, re-read, re-write*, são testadas de forma simples pelo *benchmark*, e utilizam os próprios recursos do sistema operacional. A função *write* testa o desempenho de gravação de um novo arquivo. *Re-write* testa a gravação de um mesmo arquivo já criado. *Read* avalia a leitura de um arquivo armazenado, e *Re-read* a releitura de um arquivo lido recentemente.

2.5 Aplicações Corporativas (Filebench)

Podemos definir aplicações corporativas como um estrutura de *software* projetada para contribuir com as atividades das empresas. Porém, uma aplicação corporativa não se faz somente através de um *software*, mas sim, deve ser formado por um conjunto de outros serviços. Por exemplo, um *software* de controle gerencial, necessita de serviços de armazenamento (Banco de Dados), serviços de redes (compartilhamento), controles de acessos (Samba, *firewall).* Pois um ambiente corporativo, muitas vezes se faz de um grande número de pessoas.

Devido ao aumento da quantidade de empresas, e consequentemente a utilização de tecnologias da informação para favorecer os seus processos, a procura se tornou crescente. Como citamos anteriormente na seção 2.1, empresas tem um olhar positivo para as infraestruturas de nuvens, com isso, existe outra perspectiva para uma aplicação corporativa rodar em nuvem.

Nas seções anteriores, foram citados *benchmarks* para avaliação do isolamento de recursos do ambiente computacional. Cada um deles testa em específico um componente. Porém, alguns *benchmarks* realizam testes utilizando o conjunto de recursos de uma forma mais intensa, como é o caso de testes com o Filebench. O *benchmark* simula as operações em determinados serviços, de forma muito semelhante a de um ambiente em produção. Mas, seu desempenho está ligado principalmente para a latência (m/s) nas operações de armazenamento. Porém, ainda apresenta resultados dos tempos de processamento para os testes realizados.

Os principais tipos de fluxos realizados pelo Filebench são: *Read, write, create, delete, append, getattr, setattr, readdir, semaphore block/post, rate limit, throughput limit.* E os principais atributos para os testes são: *Sync_ Data, Sync_ Metada, I/O Size, I/O Pattern, Working set size.*

Utilizando esse conjunto de fluxos, e os atributos que são repassados para cada serviço simulado, o Filebench faz com que os *kernels* Filebench simulem diversos serviços, alguns deles: Servidor de e-mail (varmail), servidor de arquivos (fileserver) e servidor WEB (webserver).

3. Implantação

Nesta seção serão abordados detalhes de implementação física e das ferramentas utilizadas durante a pesquisa.

3.1 Ambiente Nativo

O ambiente físico foi organizado em dois *clusters* com 4 *desktops* cada. Nesse tipo de organização, foi definido um *front end*, no qual estiveram alocados os principais serviços das ferramentas e sendo também *gateway* de acesso à internet para o restante dos nodos de cada *cluster*.

Cada *desktop* tem as seguintes caracteristícias: Processador *Core* i5 650 - 3.2 GHz. Memória RAM de 4 GB, placa de rede 10/100/1000, operando em padrões *gigabit*. Disco de 500 GB operando em sata II, particionado em 2 unidades e dedicando 4 GB para partição de *swap*.

Figura 3.1: Ambiente Nativo - *Cluster* OpenStack e OpenNebula

Como sistema operacional, utilizou-se a versão Ubuntu Server 14.04, por ser uma distribuição comumente utilizada em meios acadêmicos, e por possuir uma vasta documentação sobre as ferramentas de administração de nuvem e outros recursos e serviços para o ambiente. Por padrão, o sistema operacional foi devidamente atualizados em todos os hosts. É importante evidenciar que nos sistemas operacionais alocados na nuvem (ambas as ferramentas) não houveram customizações. A mesma imagem usada para instalação do ambiente Nativo fora usada para criação do ambiente da Nuvem.

Durante os testes foi necessário a utilização de uma ferramenta de monitoramento. Para esta função, utilizou-se a ferramenta Ganglia [13], voltada para monitoramento de recursos em ambientes de alto desempenho, pois exerce baixa intrusão no ambiente.

Em cada um dos *clusters* das ferramentas, foi necessário a configuração de uma unidade compartilhada com o serviço NFS. Isto foi necessário para instalação das aplicações paralelas, e também para utilização dos outros *benchmarks*. Pois desta forma, utilizou-se um diretório único em todos os ambientes.

3.2 Implantação da Ferramenta OpenNebula

Para implantação da ferramenta OpenNebula, utilizou-se a versão 4.8.0, lançada em agosto de 2014, no momento da configuração, era a versão mais estável. Nesta implantação, adotou-se o padrão recomendado pelo manual de instalação do OpenNebula no Ubuntu Server 14.04, utilizado com o virtualizador KVM.

Na Figura 3.2, podemos ver a disposição das máquinas virtuais na nuvem OpenNebula, utilizando 1 (uma) VM para cada *host*.

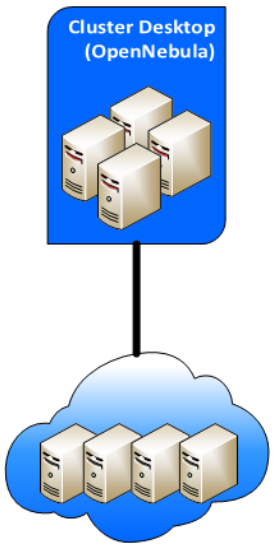

Figura 3.2: Nuvem OpenNebula

Após concluída a atualização do sistema operacional, configura-se os repositórios oficiais do OpenNebula na versão 4.8. A instalação de certa forma é simples, utilizando o próprio utilitário de instalação de pacotes do Ubuntu (apt-get) instala-se o principal pacote da ferramenta no *front end*, descrito nos manuais como "opennebula". Juntamente com este pacote, várias outras dependências são instaladas, como as bibliotecas de linguagens, gerenciador banco de dados, entre outros.

A instalação padrão recomenda a configuração de um interface *bridge*, para ser utilizada exclusivamente para comunicação entre o restante dos nodos. Esta configuração é feita de maneira simples, após a instalação do pacote para implementar as *bridges*, configura-se o arquivo de interfaces no sistema operacional.

O manual básico recomenda ainda a configuração de uma unidade compartilhada pelo *front end*, utilizando o serviço NFS do Linux. Esta unidade será montada em todos os *hosts*, e nela é armazenado os discos das instâncias, e outros arquivos essenciais.

No *front end* é importante instalar o Sunstone. Com ele, é possível gerenciar todos os recursos virtuais da nuvem. Pois através de uma interface gráfica acessível pelo navegador evita-se a digitação de comandos complexos via terminal.

Em cada nodo do *cluster*, adiciona-se os repositórios oficiais do OpenNebula, e instala-se os pacotes destinados somente aos nodos. No manual, é descrito como "opennebula-node", que com ele vários outros pacotes, bibliotecas e serviços são instalados. Da mesma forma que foi configurada as interfaces *bridges* no *front end*, também é necessário este tipo de configuração nos nodos, instalando o utilitário e editando o arquivo interfaces no sistema operacional. E a finalização da configuração dos nodes é feita com a instalação do serviços NFS, necessário para montar a unidade compartilhada.

Para adicionar nodos à infraestrutura do OpenNebula, existem duas maneiras, pela *dashboard*, ou por linha de comando. É importante que todos os serviços estejam rodando, e com os

arquivos devidamente configurados, como o serviço NFS e a unidade compartilhada montada. Outro fato que deve ser observado durante a configuração, é a partição de *swap* no Linux. Caso ela não esteja ativa, podem ocorrer inconsistências no momento da criação das VMs.

Os passos descritos anteriormente, são relatados com mais detalhes no manual básico de configuração [30], e é o ponto de partida para uma implantação da ferramenta OpenNebula.

Para explorar as diversas possibilidades da ferramenta, é necessário estudar diversas documentações existentes em seu acervo, e com isso, implementar recursos mais específicos.

3.3 Implantação da Ferramenta OpenStack

A versão utilizada para implementar a ferramenta OpenStack leva o codinome IceHouse (2014.1.3). Como foi descrito na contextualização, é possível perceber que a ferramenta possui diversos componentes, e consequentemente vários arquivos que precisam ser editados. Ferramenta esta que mais demandou atenção em relação a configuração devido aos problemas que surgiam durante todo o processo.

Na Figura 3.3, podemos ver a disposição das máquinas virtuais na nuvem OpenNebula, utilizando 1 (uma) VM para cada *host*.

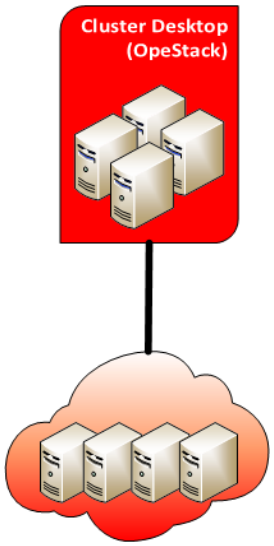

Figura 3.3: Nuvem OpenStack

A versão do Ubuntu Server 14.04 já contém os repositórios para instalação de serviços de nuvem. Portanto, após o sistema estar atualizado, iniciam-se as configurações pelo *front end*. Neste nodo, é necessário duas placas de redes, as quais devem ser configuradas de maneira que uma seja dedicada para o acesso a internet, e outra para comunicação exclusiva com o restante do *cluster*.

Inicialmente instala-se serviços como o *middleware* RabbitMQ (Troca de mensagens) e gerenciador de banco de dados MySQL. O gerenciador de banco de dados é necessário para todos os componentes desenvolvidos para o OpenStack grave informações de suas configurações e outros detalhes específicos. Em todos os serviços, o padrão para criação do banco de dados e configuração das permissões de acesso seguirá desta forma:

Acessando o gerenciador de banco de dados (SGBD):

Criando o banco

- CREATE DATABASE [serviço/componente];

Definindo as permições

- GRANT ALL PRIVILEGES ON [serviço/componente].* TO '[serviço/componente]'@'IP_do_FRONT_END' IDENTIFIED BY '[password]';
- GRANT ALL PRIVILEGES ON [serviço/componente].* TO '[serviço/componente]'@'%' IDENTIFIED BY '[password]';

Após os serviços iniciais instalados, o próximo passo é instalar o *Keystone*, o principal serviço de autenticação dos serviços da infraestrutura. Para isso, edita-se o arquivo principal com as informações para ter acesso ao banco de dados, e então esta etapa estar concluída, executa-se um comando de sincronização das tabelas MySQL. Após esta etapa, executa-se os comandos específicos do Keystone para criação dos dados principais de cada componente, são eles: (*user-create* - adicionar usuário, *tenant-create* - grupo isolado de serviços ou usuários, *role-create* - usuário master para o serviço, *service-create* - criação do serviço).

O processo de criação dos usuários e serviços se repetem para os outros componentes do OpenStack. Porém, algumas comandos não são necessários. Então, para os outros componentes, é utilizado os seguintes comandos do *keystone*:

Criação do usuário para o serviço e grupo de permissões para o usuário:

- keystone user-create –name=[serviço/componente] –pass=[password] –email=[serviço/componente]@domain.com
- keystone user-role-add –user=[serviço/componente] –tenant=service –role=admin

Criação do usuário do serviço e *endpoint* (endereço de URL usada pelo serviço)

- keystone service-create –name=[serviço/componente] –type=[Tipo_do_serviço] –description="[descrição]"
- keystone endpoint-create –service-id=$(keystone service-list | awk '/ image / print $2') –publicurl=http://[IP_externo]:[Porta_do_serviço] –internalurl=http://[IP_interno]:[Porta_do_serviço] –adminurl=http://[IP_interno]:[Porta_do_serviço]

Após realizar as configurações básicas, é importante a criação um arquivo contendo variáveis de ambiente que irão autenticar os serviços no decorrer da instalação, e também após a infraestrutura pronta. É um arquivo básico contendo as credenciais de autenticação do *keystone* e endereços para autenticação.

Seguindo com a instalação dos componentes do OpenStack, instala-se os pacotes do Glance e executa o comando para sincronização do banco de dados. Glance é o serviço para gerenciamento das imagens da nuvem, após sua instalação é necessário adicionar as imagens através de comandos específicos.

O próximo componente a ser instalado é responsável pelo controle das máquinas virtuais, o serviço Nova. Para isto, instala-se os pacotes padrões, cria-se os dados para autenticação no *keystone*, altera os arquivos principais de configuração e por fim sincroniza as tabelas no BD (Banco de Dados).

Após o Nova, é a vez do componente responsável pela rede do ambiente, o Neutron. Seguindo o mesmo padrão, instala-se os pacotes padrões, cria-se os dados para autenticação no *keystone*, altera os arquivos de configuração e por fim sincroniza as tabelas no BD. De acordo com o manual seguido, recomenda-se o uso de um node específico para gerenciar a rede do *cluster*. Porém, com a infraestrutura limitada, optou-se em configurar os serviços específicos de rede, no próprio *front end*. Nesta etapa de configuração, deve-se habilitar alguns recursos do sistema operacional para realizar o roteamento de pacotes. Ainda é necessário configurar o *plugin*

OpenvSwitch, e sub-componentes do Neutron, para comunicar-se com camadas de redes mais baixas.

O componente Cinder foi instalado para criar o armazenamento em bloco. Após a instalação dos pacotes padrões, e editado os arquivos de configuração e, criado as tabelas no banco MySQL, é necessário criar a partição LVM aonde serão armazenados os blocos de armazenamento. Para este procedimento, é necessário executar comandos que não fazem parte do componente Cinder, e são específicos da tecnologia LVM.

Na contextualização foi relatado os principais formatos de disco que a ferramenta OpenStack suporta. A configuração utilizando a tecnologia LVM é bastante comum nas infraestruturas com o OpenStack. Porém, tivemos que manter semelhanças ao ambiente com a ferramenta OpenNebula. Sendo assim, o diretório (/var/lib/nova/instances) aonde o componente Nova armazena os discos e as informações de cada VM, foi compartilhado para ser montado no restante dos nodos utilizando o serviço NFS.

E por fim, instala-se a *dashboard* (Horizon), para o *front end* estar acessível.

Para configuração dos nodos, é necessário instalação dos sub componentes do Nova, Neutron e pacotes do *hypervisor* (KVM). Após a instalação dos pacotes, edita-se os arquivos correspondentes a cada componente. É necessário uma atenção especial para os arquivos do Neutron, pois deve ser editado respeitando as redes e os tipos de protocolos que serão usados pela ferramenta.

É importante considerar que na implantação da versão IceHouse, o maior parte das inconsistências da ferramenta se concentraram nos componentes Cinder e Neutron. Para o Cinder e as tecnologias que integram a ferramenta (Ex: LVM) deve ser melhor aprofundado o estudo, pois alguns problemas surgem após o desligamento do *front end*. No Neutron, o problema esteve na comunicação da nuvem (VMs) com a internet. Problema que provavelmente acontece nos sub componentes que integram as camadas de mais baixo nível da nuvem e a rede física.

Após todos os ajustes na ferramenta, é necessário a criação as redes que serão usados na nuvem. Esse é um processo bem crítico, e é necessário muita atenção. É importante seguir uma ordem de comandos, onde primeiro cria-se a rede, seguindo pela sub-rede em comandos distintos e repassando parâmetros que serão as características das redes. O próximo passo é a criação dos roteadores e a conexão destes ao serviços do próprio Neutron, e especificando quais redes os roteadores irão se conectar. Durante a criação do roteador virtual, é necessário repassar qual rede deverá ser usada para acesso externo da nuvem, e qual o seu endereço de *gateway*.

3.4 Preparação do Ambiente

Com o estudo de diversos artigos publicados [36, 31, 16, 27, 9, 32, 25], pode-se perceber uma dimensão sobre a realização de testes, como também a utilização das ferramentas e a forma de avaliação. Estes *benchmarks* foram relatados na seção 2.

A suíte NPB OMP e MPI foram compiladas usando a classe B. Chegou-se a esta definição devido aos testes realizados com as outras classes, onde algumas praticamente não exerciam nenhum estresse significativo no ambiente, e outras não foram usadas devido a grande carga que aplicavam no momento de sua execução.

Portanto, os seguintes *kernels* utilizados foram: BT, CG, EP, FT, IS, LU MG e SP. Como o padrão MPI implementa seu *kernels* instanciando processos pelos *hosts* do *cluster*, no momento da compilação o algoritmo já verifica a quantidade de processos compatíveis com cada *kernel*. Portanto, os *kernels* do MPI ficaram assim: LU, FT, EP, CG, IS e MG: 1,4,8,16 processos. SP e BT: 1,4,9,16 processos. Na suíte OMP, uma variável de ambiente foi definida para indicar a quantidade de *threads* durante a execução. Como a arquitetura física do processador possuí 4 *threads*, esse foi o valor máximo definido nesta variável.

Para a execução completa das suítes, foi criado um *shell script* para automatizar a execução delas. Cada *kernel* foi executado 40 vezes, afim de cálculo de média. Durante a execução, foi respeitado um ordem aleatória e contínua, aonde cada *kernel* era executado após o outro, até completar as 40 vezes em cada processo ou em cada *thread*.

Para avaliação do processador, foi utilizado o *benchmark* LINPACK. Utilizou-se o código original realizando apenas uma alteração para definir dentro código o valor da matriz utilizada durante o teste (4000 X 4000). Foi o maior valor encontrado para que durante o teste, o código não quebrasse em partes as execuções. Sendo assim, utilizando esta matriz, o *benchmark* executava em apenas um ciclo do código.

Para os testes com a unidade de armazenamento, utilizou-se o IOzone [18]. Este é um *benchmark* que pode ser instalado pelo próprio utilitário do Linux (*apt-get*). O IOzone, permite uma diversidade de testes, mas inicialmente, usou-se alguns atributos e funções básicas para testes específico das operações de armazenamento. Para executa-lo, foi utilizado o parâmetro aonde o IOzone cria um arquivo de 100 MB para realizar as funções de escrita, reescrita, leitura e releitura (*write, rewrite, read e reread*).

Para testes de rede foi utilizado Iperf, instalado pelos repositórios oficiais do Ubuntu. Sua execução é simples, e não foi utilizado nenhum padrão específico para os testes, apenas os parâmetros de inicialização do cliente e do servidor. Caso nada seja declarado no momento da execução, o processo cliente realiza as requisições ao servidor utilizando um tempo padrão de 10 segundos.

O *benchmark* STREAM foi usado para avaliação da largura de banda da memória RAM. Para execução deste, foi utilizado o código original escrito em linguagem C. Durante a execução, não é necessário nenhum parâmetro específico, mas internamente o *benchmark* realiza algumas rotinas, e utiliza parâmetros estáticos como o tamanho da matriz endereçada na memória, sendo de 2.000.000 posições. O tamanho de memória requerida para o armazenamento desta matriz, sendo o valor de 45,8 MB. E cada chamada de execução do STREAM, o código é executado internamente 10 vezes no sistema, e destas 10 execuções, apenas o melhor resultado é repassado. Com estes parâmetros implementados pelo código do STREAM, ele executa os testes de cópia, adição, escala e tríade dos valores na memória RAM.

Sendo assim, para avaliação do ambiente Nativo e em nuvem, foram utilizados os *benchmarks* IOzone para teste da unidade de armazenamento. LINPACK para avaliar o desempenho do processador. STREAM para teste de memória RAM, e IPERF para avaliar o desempenho da rede. Todos estes benchmarks seguiram um ciclo de 40 execuções, aonde foram implementadas através de um *Shell Script*, que automatizou todo este processo de execução e coleta dos resultados.

O *benchmark* Filebench [11], foi utilizado para testes com aplicações corporativas. Para a instalação é necessário os *sources* do *benchmark*, e algumas bibliotecas em especifico, ao qual sempre são verificadas durante a instalação inicial. Para a compilação e instalação, utiliza-se o utilitário *make* do próprio sistema operacional. Com a compilação concluída, o *benchmark* irá criar os binários para execução de cada *kernel*. São estes que irão simular as cargas de trabalhos reais em cada serviços testado pelo Filebench. Para o teste inicial, utilizou-se os *kernels* varmail, webserver e fileserver, que respectivamente simulam um servidor de e-mails, servidor WEB e servidor de arquivos. Cada *kernels* foi executado 40x, e cada execução durou 1 (um) minuto.

4. Resultados

Nesta Capítulo, serão apresentados a metodologia e os resultados dos testes realizados durante a pesquisa. Os resultados por sua vez, serão plotados em gráficos relativos ao desempenho dos *benchmarks* e as métricas coletadas são descritas em tabelas para a análise estatística.

4.1 Metodologia dos testes

Ao se tratar de avaliação de recursos, deve-se observar os métodos que são empregados, e consequentemente quais os tipos de métricas que serão coletadas nos testes. Alguns *benchmarks* nos trazem métricas mais completas sobre determinados testes. Contudo, na pesquisa, deve-se observar o objetivo das avaliações (Desempenho, Eficiência, Disponibilidade, *Speed-up*).

Para execução dos testes, procurou-se manter o ambiente mais homogêneo possível em questões de metodologia e tecnologia usada pelas ferramentas. Isto para garantir uma análise justa dos resultados em cada ambiente de nuvem. Portanto, partindo do princípio, cada *benchmark* foi executado 40x, para fim de cálculo de média. Durante a execução, sempre procurou-se observar o comportamento de cada *benchmark*, afim de evitar algum viés ou perda durante a execução. Então, teve-se o cuidado de em cada execução limpar os dados da unidade de *swap*, caso alguma aplicações viesse a fazer algum reuso de algum dado armazenado, oriundo de outras execuções (cache). Procurou-se manter o mínimo de aplicações rodando em cada ambiente, dedicando o máximo dos recursos para cada determinada execução.

E para tornar os resultados legítimos, a análise estatística serviu para encontrar diferenças significativas em cada execução. Deste modo, a análise foi realizada com SPSS [10], de maneira que todos os resultados fossem pareados e com uma margem de confiança de 95%. Métrica comumente utilizada em experimentos de *softwares*. A variável Sig. retornada pelo *software* de análise estatística, define os resultados significativamente diferentes quando seu valor for menor que 0,05. Para representar toda a análise estatística, montamos as tabelas no decorrer dos resultados com uma simbologia para melhor representar as informações. Para o resultado que não é significativamente diferente, o valor Sig. estará na cor verde. As médias aritméticas nas cores vermelha e azul, respectivamente representam superioridade de OpenStack ou OpenNebula.

As informações e suas simbologias são:

- Média Aritmética: $\bar{x}$
- Desvio Padrão: σ
- Variável Sig.: $Sig.$
- Ambiente OpenStack: β
- Ambiente OpenNebula: Ω
- Ambiente Nativo π

4.2 Isolamento de recursos

Nesta seção será detalhado os resultados numéricos dos *benchmarks* para avaliação do isolamento de recursos, informações que podem ser visualizadas na Figura 4.1.

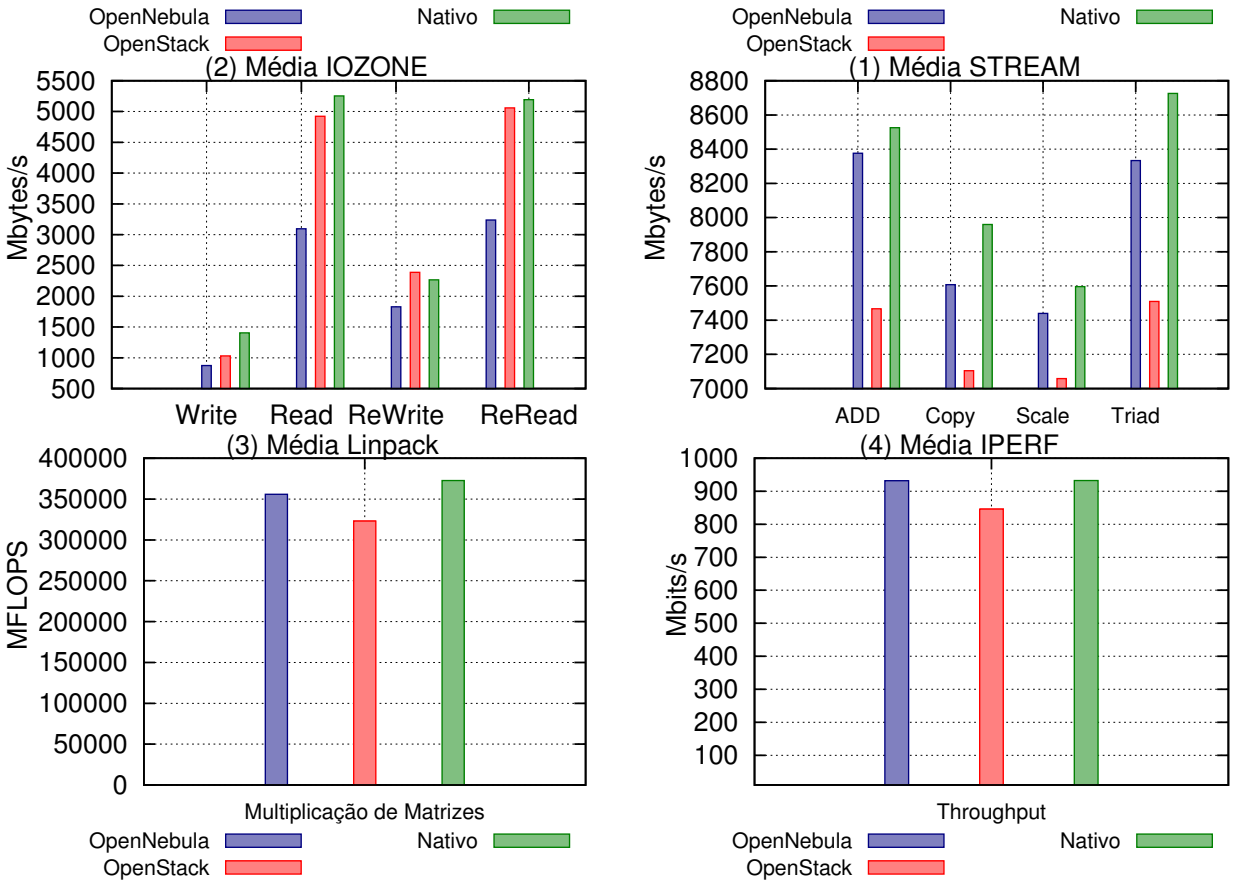

Figura 4.1: Isolamento de Recursos

- STREAM (1)

  No gráfico dos resultados, percebemos o bom desempenho do OpenNebula em relação ao OpenStack nos testes de largura de banda da memória RAM, se aproximando dos resultados do ambiente Nativo. De acordo com as características das ferramentas, ao qual percebemos na seção 2, OpenStack possui um conjunto bem específico de componentes. Acredita-se que o componente *nova-compute-KVM*, juntamente com o *middleware* RabbitMQ, de alguma forma tenha influenciado os resultados. A justificativa para o RabbitMQ, é que devido ele ser o responsável pela comunicação entre os serviços da infraestrutura OpenStack, em algum momento do teste isso tenha prejudicado a sua execução.

  O *nova-compute-KVM* é o principal componente que realiza algumas chamadas específicas ao virtualizador. Durante a execução da VM, informações da mesma são alocadas em tempo real em um banco de dados SQL. Porém, na versão IceHouse, a subcomponente *nova-conductor* faz a intermediação desta comunicação entre o componente e o banco de dados.

  No OpenNebula, existe apenas o componente ONED, que exerce uma comunicação direta com a VM. Inclusive, em toda a documentação da ferramenta, não foi encontrado nenhum vestígio de que o componente interferisse, a ponto de ser semelhante à ferramenta OpenStack.

  Outro fato que deve ser melhor investigado, está em algumas particularidades do virtualizador. A ferramenta OpenStack, implementa alguns *drivers* (virtIO) automaticamente durante a criação da instância. No OpenNebula, acontece o contrário, é necessário a con-

figuração na *template* de criação das VMs, para que isto seja repassado ao virtualizador. Acredita-se que o modulo que trabalha em conjunto com o virtualizador, chamado EPT, não está sendo habilitado para as instancias do OpenStack. Este módulo é responsável em tornar a tabela de endereçamento de memória, acessível pela própria VM, evitando neste caso, alguma sobrecarga de *hardware*, e assim disponibilizar esses endereços diretamente na VM. É Neste sentido, pode haver alguma configuração específica em que se deve habilitar esse módulo para ser usado nas instâncias OpenStack. Isso é algo sugestivo. Não encontramos evidências suficientes para validar essa informação.

Nos resultados estatísticos, OpenNebula foi positivamente melhor neste tipo de teste, sendo todos os resultados significativamente diferentes ao OpenStack.

- IOzone (2)

  Nos testes de desempenho com a unidade de armazenamento, a ferramenta OpenStack se destacou nos resultados.

  Como foi descrito na Seção 3, a implantação da ferramenta OpenNebula, utiliza a unidade compartilhada pelo *front end*, fazendo uso do protocolo NFS e formato de disco QCOW2. No OpenStack, o ambiente foi semelhante, onde o mesmo formato de disco e o mesmo protocolo de compartilhamento (NFS) foram utilizados.

  O bom desempenho é evidente para a ferramenta OpenStack, conseguindo alcançar melhores resultados em relação ao OpenNebula. Este é um fato que precisa ser melhor investigado, pois não foi encontrado documentação suficiente que detalhasse o funcionamento das tecnologias envolvidas a mais baixo nível. Sendo assim, acreditamos que o trabalho envolvendo o componente Cinder, juntamente com o *middleware* RabbitMQ, permite que OpenStack tenha um bom desempenho quando se trata deste tipo de teste.

  Ao analisar a ferramenta OpenNebula, percebemos que se faz o uso das tecnologias em seu estado natural de uso, e que não possui nenhum componente aparente que permite um ganho ou perda de desempenho. Desta forma, para o resultado do OpenStack, as evidências ficam fortes para os componentes Cinder e *RabbitMQ*

- Iperf(4)

  O Iperf foi utilizado para medir a largura de banda. Nos resultados ambiente do Nativo e OpenNebula alcançaram desempenhos muito próximos.

  O virtualizador KVM, possui um *driver* chamado virtIO, que ao ser implementado nas instâncias virtuais, permite que as interfaces da instância trabalhem em um modo de paravirtualização. Isso permite que I/O de rede e algumas funções de discos são privilegiadas por este *driver*.

  Contudo, devido a forma de implementação e a utilização de determinados recursos, faz com que o OpenStack tenha seu desempenho prejudicado pelo *plugin* OpenvSwitch, que implementa interfaces virtuais no sistema operacional nativo, e o componente Neutron, que cria redes, sub-redes e roteadores virtuais. O OpenvSwitch, além de criar um tunelamento exclusivo para comunicação interna de alguns componentes da ferramenta, faz com que mais camadas de *software* sejam adicionadas durante a comunicação.

  Para implantar as redes na ferramenta OpenNebula, foi utilizado um recurso básico desenvolvido pela própria OpenNebula *Systems*, o *driver* Dummy. Este *driver*, implementa a comunicação de rede na nuvem, de uma forma livre, sem restrições e regras de segurança. Devido a estes recursos, o desempenho de rede na ferramenta OpenNebula permitiu competir de igual ao ambiente Nativo, onde o resultado estatístico na Tabela 4.3 mostra que não foi possível definir qual o melhor ambiente.

- LINPACK (3)

  Utilizou-se o *benchmark* LINPACK para avaliar a capacidade do processador, que avalia a quantidade de pontos flutuantes (FLOPS) que o processador utiliza para concluir determinada tarefa.

  A lista de *deamons* em execução nos nodos do *cluster* OpenNebula, estão somente processos dos serviços de virtualização. Na ferramenta OpenStack, os processos são os componentes *nova-compute*, OpenvSwitch, Neutron, e também processos das ferramentas de virtualização. Acreditamos que ao executar uma instância, os processos da ferramenta OpenStack estão competindo recursos com o ambiente Nativo, degradando o desempenho das instâncias virtuais.

As Tabelas 4.1, 4.2 e 4.3 possuem os resultados estatísticos de isolamento de recursos.

Tabela 4.1: Resultados estatísticos OpenStack vs OpenNebula (isolamento de recursos)

		OpenStack (β)		OpenNebula (Ω)		β *vs* Ω
Benchmark	Medida	$\bar{x}$ (40x)	σ (40x)	$\bar{x}$ (40x)	σ (40x)	*Sig.*
LINPACK	CPU	330723142,10	102237375,27	364495563,02	729840,02	,044
IPERF	Throughput	846,20	62,96	932,77	2,46	,000
STREAM	ADD	7466,36	30,55	8376,53	209,08	,000
	COPY	7104,69	44,99	7607,84	206,33	,000
	SCALE	7058,23	51,86	7439,58	208,41	,000
	TRIAD	7509,13	36,51	8332,84	245,49	,000
IOzone	Write	1030,26	100.96	873,04	131,57	,000
	ReWrite	2387,39	131,57	1827,29	116,42	,000
	Read	4924,05	313,14	3094,43	145,63	,000
	ReRead	5058,30	260,48	3236,96	90,11	,000

Tabela 4.2: Resultados estatísticos OpenStack vs Nativo (Isolamento de recursos)

		OpenStack (β)		Nativo (π)		β *vs* π
Bench.	Medida	$\bar{x}$ (40x)	σ (40x)	$\bar{x}$ (40x)	σ (40x)	*Sig.*
LINPACK	CPU	330723142,10	102237375,27	372782,72	304,40	,000
Iperf	Throughput	846,20	62,96	932,35	,83	,000
STREAM	ADD	7466,36	30,55	8525,76	55,42	,000
	COPY	7104,69	44,99	7959,16	104,60	,000
	SCALE	7058,23	51,86	7595,65	40,76	,000
	TRIAD	7509,13	36,51	8725,63	72,46	,000
IOzone	Write	1030,26	1406,18	1406,18	61,04	,000
	ReWrite	2387,39	131,57	2265,48	49,05	,000
	Read	4924.05	313,14	4924,05	129,20	,000
	ReRead	5058.30	260,48	5192,15	260,48	,000

4.2.1 Visão Geral

Nesta seção, foram apresentados os testes de isolamento de recursos (Memória RAM, processador, armazenamento e rede), respectivamente utilizando os *benchmarks* STREAM, LINPACK, IOzone, Iperf.

Neste ambiente controlado, os resultados de memória RAM, processador e rede, foram favoráveis ao OpenNebula, se mostrando significativamente diferente que a ferramenta OpenStack. Porém, OpenStack teve um resultado surpreendente no desempenho de disco, sendo o ambiente de nuvem na ferramenta OpenStack significativamente melhor que o ambinte nativo, e consequentemente melhor que o OpenNebula.

Tabela 4.3: Resultados estatísticos OpenNebula vs Nativo (isolamento de recursos)

		OpenNebula (Ω)		Nativo (π)		Ω *vs* π
Bench.	Medida	$\bar{x}$ (40x)	σ (40x)	$\bar{x}$ (40x)	σ (40x)	*Sig.*
LINPACK	CPU	364495563,02	729840,02	372782,72	304,40	,000
Iperf	Throughput	932,77	2,46	932,35	,83	,322
STREAM	ADD	8376,53	209,08	8525,76	55,42	,000
	COPY	7607,84	206,33	7959,16	104,60	,000
	SCALE	7439,58	208,41	7595,65	40,76	,000
	TRIAD	8332,84	245,49	8725,63	72,46	,000
IOzone	Write	873,04	131,57	1406,18	27,98	,000
	ReWrite	1827,29	116,42	2265,48	49,05	,000
	Read	3094,43	145,63	4924,05	129,20	,000
	ReRead	3236,96	90,11	5192,15	260,48	,000

É necessário uma pesquisa mais aprofundada na tecnologia LVM para descobrir características de mais baixo nível. A definição é escassa, e alguns resultados ainda se tornam uma incógnita.

4.3 Aplicações Paralelas

Nesta seção, serão detalhados os resultados das execuções do *benchmark* NAS. Na Figua 4.2 estão os gráficos dos resultados do tempo de execução utilizando as bibliotecas OMP. Na Figura 4.3, estão os gráficos de tempo de execução usando a biblioteca MPI.

A utilização do recurso *Hyper Threading* (HT) afeta em alguns aspectos algumas execuções a partir de 3 *threads* (OMP) e de 9 processos (MPI). Esse é um processo natural, o recurso HT ao ser ativado, não possui um desempenho similar aos núcleos físicos do processador e, ainda compete pela utilização da largura de banda entre processador e memória RAM.

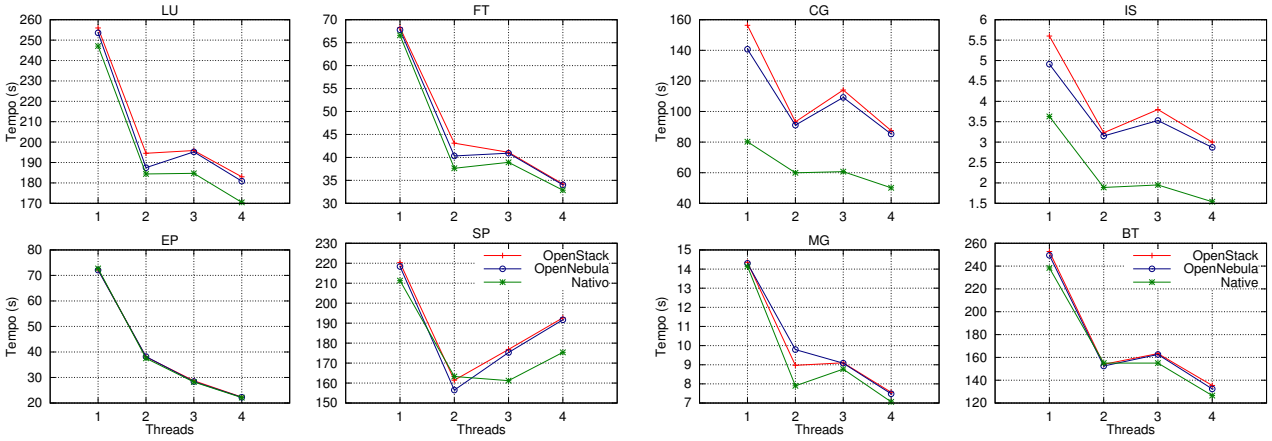

Figura 4.2: Análise dos resultados numéricos das aplicações usando OpenMP

Os resultados das execuções do isolamento de recursos, servem como referência para execução das aplicações paralelas. Durante a execução, cada *kernel* carrega do disco o código de execução e aloca na memória RAM, após isso, quase todo o tempo da execução é utilizando recursos de memória, processador e rede (para MPI). Sendo assim, as aplicações do NAS utilizam o disco para um eventual registro em log, ou para gravar algumas informações do algoritmo.

A Tabela 4.4 mostra resultados estatísticos com os resultados do NPB-OMP.

Nas execuções com *kernel* EP (OMP e MPI), os resultados foram próximos ao Nativo. A sua execução é por meio de cargas paralelas independentes, executando de forma contínua, sem

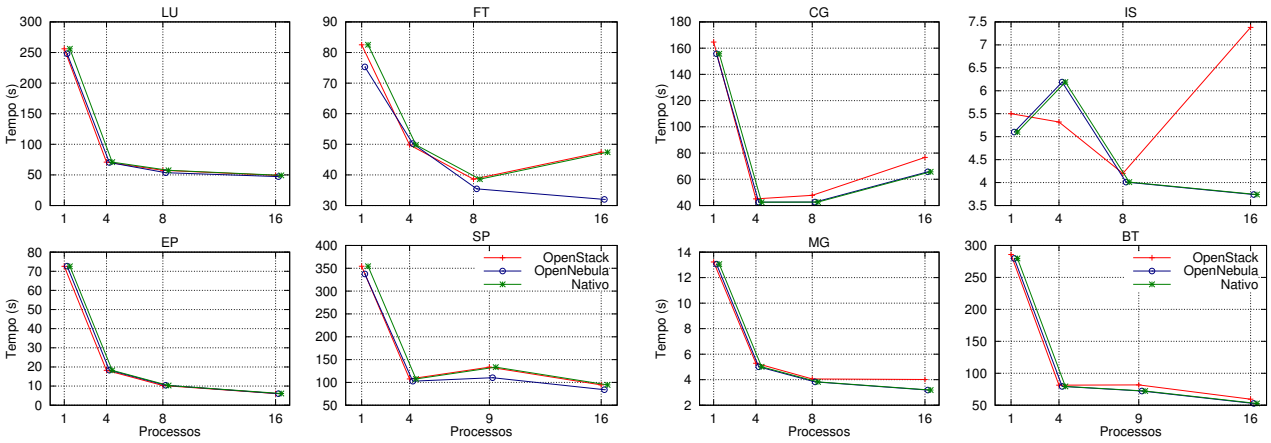

Figura 4.3: Análise dos resultados numéricos das aplicações usando MPI

Tabela 4.4: Resultados estatísticos de Nuvem vs Nativo (NPB-OMP)

		OpenStack-OMP (β)		Nativo-OMP (π)		β *vs* π
Bench.	Th.	$\bar{x}$ (40x)	σ (40x)	$\bar{x}$ (40x)	σ (40x)	*Sig.*
BT	1	252,74	1,07	238,39	1,90	,000
	2	153,92	17,58	155,00	25,76	,822
	3	163,33	,49	154,97	,31	,000
	4	135,38	,17	126,57	,13	,000
CG	1	156,32	20,53	80,24	,78	,000
	2	93,27	16,47	59,96	13,99	,000
	3	114,05	,13	60,71	,08	,000
	4	87,48	7,09	50,14	,05	,000
EP	1	72,27	,35	72,74	,61	,000
	2	38,05	1,29	37,52	,60	,030
	3	28,73	,35	28,25	,39	,000
	4	22,33	,13	22,13	,15	,000
FT	1	68,25	,11	66,59	,52	,000
	2	43,11	6,22	37,64	3,04	,000
	3	41,11	,29	38,94	,15	,000
	4	34,22	,09	32,87	,36	,000
IS	1	5,60	,11	3,63	,03	,000
	2	3,23	,76	1,89	,01	,000
	3	3,80	,07	1,95	,01	,000
	4	3,00	,04	1,54	,01	,000
LU	1	255,94	,59	247,08	1,86	,000
	2	194,55	12,19	184,41	9,84	,000
	3	195,90	,74	184,71	,60	,000
	4	183,02	,44	170,47	,24	,000
MG	1	14,35	,01	14,14	,12	,000
	2	8,97	1,52	7,90	,75	,000
	3	9,10	,02	8,77	,03	,000
	4	7,56	,04	7,06	,05	,000
SP	1	220,30	,27	211,27	1,63	,000
	2	161,40	13,10	163,29	18,05	,000
	3	176,85	,84	161,20	,58	,000
	4	192,66	,62	175,39	,37	,000

	OpenNebula-OMP (Ω)		Nativo-OMP (π)		Ω *vs* π
Proc.	$\bar{x}$ (40x)	σ (40x)	$\bar{x}$ (40x)	σ (40x)	*Sig.*
1	249,47	3,02	238,39	1,90	,595
2	152,46	13,53	155,00	25,76	,000
3	162,51	,68	154,97	,31	,000
4	132,33	,45	126,57	,13	,000
1	140,68	34,07	80,24	,78	,000
2	91,11	21,67	59,96	13,99	,000
3	109,21	13,97	60,71	,08	,000
4	85,30	9,82	50,14	0,5	,000
1	72,09	,30	72,74	,61	,000
2	38,18	1,25	37,52	,60	,002
3	28,41	,38	28,25	,39	,53
4	22,25	,13	22,13	,15	,001
1	67,80	,36	66,59	,52	,000
2	40,33	,3,38	37,64	3,04	,002
3	40,92	,24	38,94	,15	,000
4	34,02	,10	32,87	,36	,000
1	4,91	,66	3,63	,03	,000
2	3,15	1,09	1,89	,01	,000
3	3,53	,24	1,95	,01	,000
4	2,87	,11	1,54	,01	,000
1	253,56	1,71	247,08	1,86	,000
2	187,48	6,40	184,41	9,84	,127
3	195,21	1,21	184,71	,60	,000
4	180,89	,96	170,47	,24	,000
1	14,30	,03	14,14	,12	,000
2	9,79	2,02	7,90	,75	,000
3	9,08	,02	8,77	,03	,000
4	7,48	,06	7,06	,05	,000
1	218,39	,98	211,27	1,63	,000
2	156,48	,10,67	163,29	18,05	,041
3	175,20	1,22	161,20	,58	,000
4	191,68	,97	175,39	,37	,000

consulta a outros processos durante a execução. O MG é um algoritmo *multigrid* simplificado que exige uma comunicação estruturada entre seus processos (OMP e MPI), neste caso, percebe-se que nas execuções MPI, ao aumentar o fluxo das comunicação, o baixo desempenho da rede no OpenStack, faz o tempo de execução se elevar.

A Tabela 4.5 mostra resultados estatísticos com os resultados do NAS utilizando bibliotecas MPI comparando o ambiente de nuvem (OpenStack e OpenNebula) com o ambiente Nativo.

A Tabela 4.6 mostra resultados estatísticos com os resultados do NAS, aonde em um ambiente de nuvem, as execuções do OMP BT, EP, IS, SP, e CG com 2 *threads* e CG 4, não foi possível definir as melhores ferramentas e apenas na execução do MG 2 e FT 1 que a ferramenta OpenStack foi significativamente melhor.

As execuções do CG, são envolvidas por grandes problemas matemáticos, e necessitam de um grande comunicação entre processos e endereços alocados em memória e o *kernel* IS, também está relacionado, porém com uma ordenação de valores em memória RAM. Com isso, o baixo

Tabela 4.5: Resultados estatísticos de Nuvem vs Nativo (NPB-MPI)

		OpenStack-MPI (β)		Nativo-MPI (Ω)		π *vs* Ω
Bench.	Th.	$\bar{x}$ (40x)	σ (40x)	$\bar{x}$ (40x)	σ (40x)	*Sig.*
BT	1	285,83	1,11	260,51	,63	,000
	4	81,43	,31	75,42	,47	,000
	9	81,87	1,77	68,02	,99	,000
	16	59,46	,68	47,79	,29	,000
CG	1	164,64	,86	88,30	,49	,000
	4	45,09	,61	37,72	,18	,000
	8	47,81	2,02	37,40	4,62	,000
	16	76,65	1,65	53,77	,11	,000
EP	1	72,58	,31	72,10	,44	,000
	4	18,24	,08	75,42	,47	,000
	8	10,30	,59	10,37	,78	,619
	16	6,11	,10	5,94	,04	,000
FT	1	82,51	26,28	77,44	,34	,228
	4	49,81	,70	47,78	,36	,000
	8	38,60	,67	34,28	2,94	,000
	16	47,44	1,04	29,51	,05	,000
IS	1	5,50	,05	3,33	,02	,000
	4	5,32	,20	4,32	,04	,000
	8	4,20	,95	3,09	,27	,000
	16	7,38	,45	3,12	,14	,000
LU	1	256,12	,94	236,42	,42	,000
	4	70,88	1,72	68,12	,52	,000
	8	57,55	3,78	47,86	6,49	,000
	16	49,36	,65	35,81	,41	,000
MG	1	13,23	,09	12,48	,07	,000
	4	5,29	,10	4,71	,03	,000
	8	4,06	,40	3,74	,52	,000
	16	4,02	,30	2,65	,01	,000
SP	1	354,58	1,60	299,28	,48	,000
	4	107,59	,60	93,95	2,34	,000
	9	133,49	4,11	97,17	,95	,000
	16	94,49	,71	72,55	,35	,000

	OpenNebula-MPI (Ω)		Nativo-MPI (π)		Ω *vs* π
Proc.	$\bar{x}$ (40x)	σ (40x)	$\bar{x}$ (40x)	σ (40x)	*Sig.*
1	279,74	1,83	260,51	,63	,000
4	79,47	,31	75,42	,47	,000
9	72,28	,63	68,02	,99	,000
16	52,83	,49	47,79	,29	,000
1	155,71	21,54	88,30	,49	,000
4	42,59	,76	37,72	,18	,000
8	42,63	3,37	37,40	4,62	,000
16	65,73	2,86	53,77	,11	,000
1	72,68	,37	72,10	,44	,000
4	18,30	,10	18,20	,12	,000
8	10,43	,69	10,37	,78	,723
16	6,05	,05	5,94	,04	,000
1	75,29	8,43	77,44	,44	,112
4	50,24	,31	47,78	,36	,000
8	35,45	1,88	34,28	2,94	,033
16	31,99	,62	29,51	,05	,000
1	5,10	,30	3,33	,02	,000
4	6,19	,45	4,32	,04	,000
8	4,01	,32	3,09	,27	,000
16	3,74	,74	3,12	,14	,000
1	248,07	1,14	236,42	,42	,000
4	70,27	1,37	68,12	,52	,000
8	53,63	3,57	47,86	6,49	,000
16	47,34	,87	35,81	,41	,000
1	13,06	,09	12,48	,07	,000
4	5,01	,04	4,71	,03	,000
8	3,83	,40	3,74	,52	,415
16	3,19	,24	2,65	,01	,000
1	337,58	2,75	299,28	,48	,000
4	102,70	,55	93,95	2,34	,000
9	110,19	1,21	97,17	,95	,000
16	84,10	,74	72,55	,35	,000

Tabela 4.6: Resultados estatísticos Nuvem vs Nuvem (NPB-MPI e OMP)

		OpenStack-OMP (β)		OpenNebula-OMP (Ω)		β *vs* Ω
Bench.	Th.	$\bar{x}$ (40x)	σ (40x)	$\bar{x}$ (40x)	σ (40x)	*Sig.*
BT	1	252,74	1,07	249,47	3,02	,000
	2	153,92	17,58	152,46	13,53	,679
	3	163,33	,49	162,51	,68	,000
	4	135,38	,17	132,33	,45	,000
CG	1	156,32	20,53	140,68	34,07	,026
	2	93,27	16,47	91,11	21,67	,645
	3	114,05	,13	109,21	13,97	,035
	4	87,48	7,09	85,30	9,82	,279
EP	1	72,27	,35	72,09	,30	,021
	2	38,05	1,29	38,18	1,25	,682
	3	28,73	,35	28,41	,38	,001
	4	22,33	,13	22,25	,13	,010
FT	1	68,25	,11	67,80	,36	,000
	2	43,11	6,22	40,33	3,38	,014
	3	41,11	,29	40,92	,24	,005
	4	34,22	,09	34,02	,07	,000
IS	1	5,60	,11	4,91	,66	,000
	2	3,23	,76	3,15	1,09	,722
	3	3,80	,07	3,53	,24	,000
	4	3,00	,04	2,87	,11	,000
LU	1	255,94	,59	253,56	1,71	,000
	2	194,55	12,19	187,48	6,40	,002
	3	195,90	,74	195,21	1,21	,004
	4	183,02	,44	180,89	,96	,000
MG	1	14,35	,01	14,30	,03	,000
	2	8,97	1,52	9,79	2,02	,048
	3	9,10	,02	9,08	,04	,006
	4	7,56	,04	7,48	,06	,000
SP	1	220,30	,27	218,39	,98	,000
	2	161,40	13,10	156,48	10,67	,082
	3	176,85	,84	175,28	1,22	,000
	4	192,66	,62	191,68	,97	,000

	OpenStack-MPI (β)		OpenNebula-MPI (Ω)		β *vs* Ω
Proc.	$\bar{x}$ (40x)	σ (40x)	$\bar{x}$ (40x)	σ (40x)	*Sig.*
1	285,83	1,11	279,74	1,83	,000
4	81,43	,31	79,47	,31	,000
9	81,87	1,77	72,28	,63	,000
16	59,46	,68	52,83	,49	,000
1	164,64	,86	155,71	,21	,014
4	45,09	,61	42,59	,76	,000
8	47,81	2,02	42,63	3,37	,000
16	76,65	1,65	65,73	2,86	,000
1	72,58	,31	72,69	,37	,183
4	18,24	,08	18,30	,10	,004
8	10,30	,59	10,43	,69	,341
16	6,11	,10	6,05	,05	,003
1	82,51	26,28	75,29	8,43	,057
4	49,81	,70	50,24	,31	,000
8	38,60	,67	35,45	1,88	,000
16	47,44	1,04	31,99	,62	,000
1	5,50	,05	5,10	,30	,000
4	5,32	,20	6,19	,45	,000
8	4,20	,95	4,01	,32	,232
16	7,38	,45	3,74	,74	,000
1	256,12	,94	248,07	1,14	,000
4	70,88	1,72	70,27	1,37	,087
8	57,55	3,78	53,63	3,57	,000
16	49,36	,65	47,34	,87	,000
1	13,23	,09	13,06	,09	,000
4	5,29	,10	5,01	,04	,000
8	4,06	,40	3,83	,40	,020
16	4,02	,30	3,19	,24	,000
1	354,58	1,60	337,58	2,75	,000
4	107,59	,60	102,70	,55	,000
9	133,49	4,11	110,19	,71	,000
16	94,49	,71	84,10	,74	,000

desempenho de rede na ferramenta OpenStack, faz o tempo de execução CG aumentar. E como a ferramenta OpenNebula apresentou melhores resultados em processamento envolvendo memória RAM e rede, *benchmark* teve seu tempo de execução menor em IS e CG.

4.3.1 Visão Geral

Nas seções anteriores, foram apresentados os resultados dos testes com aplicações paralelas, utilizando a suíte NBP com as bibliotecas MPI e OMP.

Utilizando o OMP, OpenStack alcançou melhores resultados nas execuções com o SP 2, sendo significativamente melhor que o Nativo. E nas execuções com BT 2, não foi possível definir a melhor ferramenta. No OpenNebula, os resultados foram melhores que o Nativo nos testes com o BT 2 e EP 1. E nos testes com o BT 1, EP 3, LU 2, não foi possível definir qual o melhor ambiente.

Nos testes com o NAS utilizando a biblioteca MPI, nos testes com EP 8, FT 1 não foi possível definir qual o melhor ambiente. No restante dos testes o Nativo foi significativamente melhor.

Comparando o ambinte Nativo com a ferramenta OpenNebula, apenas nas execuções EP 8, FT 1 e MG 8, não foi possível definir o melhor ambiente para as execuções.

Isso mostra que alguns tipos de execuções de aplicações paralelas, o ambiente da Nuvem exerce uma influência positiva. É necessário um estudo mais específico nos virtualizadores, nas ferramentas de nuvem e nos códigos dos *kernels* para apontar algo específico que tenha realmente influenciado os testes.

Os testes comparando as execuções no OpenNebula e OpenStack com as bibliotecas OMP, os *kernels* BT 3, CG 2 e 4, EP 2, IS 2 e SP 2, não foi possível definir a melhor ferramenta para este tipo de execução. E apenas em teste com o MG 2 e FT 1 do padrão OMP, que o OpenStack foi melhor que OpenNebula.

Nas execuções com o MPI, EP 1 e 8, FT 1, IS 8, os resultados não foram significativamente diferentes entre as ferramentas. E nas execuções com EP 4, FT 4 e IS 4, OpenStack foi melhor.

Como se é entendido, MPI utiliza suas execuções em memória distribuída, e consequentemente utiliza a rede para isto. Em contraste com os resultados de isolamento de recursos, OpenNebula teve o melhor desempenho em rede, e principalemnte em memória e processador. OMP é executa em memória compartilhada, e os resultados de isolamento de recursos favorecem à ferramenta OpenNebula. Contudo, alguns resultados podem estar relacionados ao tipo de execução aplicado por cada *kernel*.

4.4 Aplicações Corporativas

Nesta seção, serão descritos os resultados numéricos dos *benchmarks* de simulação de aplicações corporativas, Filebench e a análise estatísticas destes resultados.

Na Figura 4.4, consta os resultados da simulação de um servidor de e-mails. A Figura 4.5 possui os resultados com a simulação de um servidor de arquivos. E por fim, a Figura 4.6, mostra os resultados com a simulação de um servidor WEB. Todos os *kernels* do Filenbench, seguiram os mesmos padrões relatados na metodologia dos testes, porém, com a diferença que a execução de cada *kernel* durou 1 minuto, parâmetro necessário durante a sua execução. O restante dos parâmetros, adotou-se o valor padrão que já era definido internamente no código do *kernel*.

Cada um dos *kernerls* do Filebench trabalha com a funções que estão diretamente ligadas ao uso de disco. Mas, cada *kernel* é simulado de maneira exclusiva. Por exemplo: Ao simular um servidor de e-mails, o *kernel* Cria, deleta, abre, encerra, Lê, sincroniza arquivos em disco como se fosse e-mails contendo anexos.

Sendo assim, o desempenho está relacionado aos argumentos citados nos testes de isolamento de recursos utilizando o IOzone, onde OpenStack teve um bom desempenho em disco. Porém, como este *benchmark* realiza a repetição destas funções durante o ciclo de execução, o mesmo

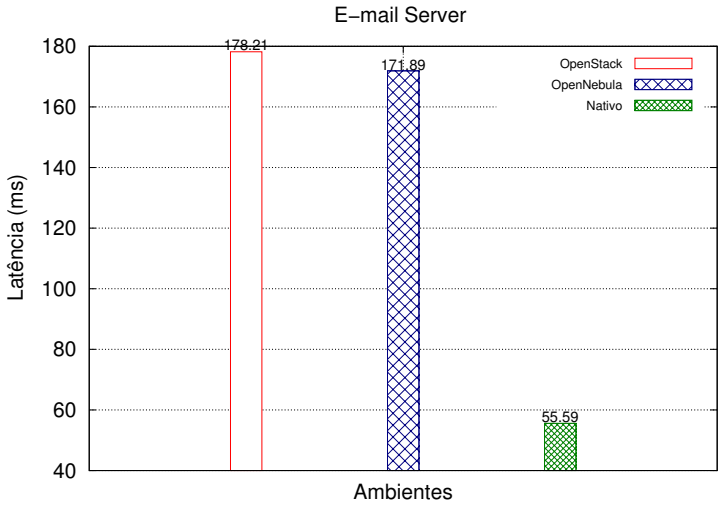

Figura 4.4: Testes numéricos com Servidor de e-mails

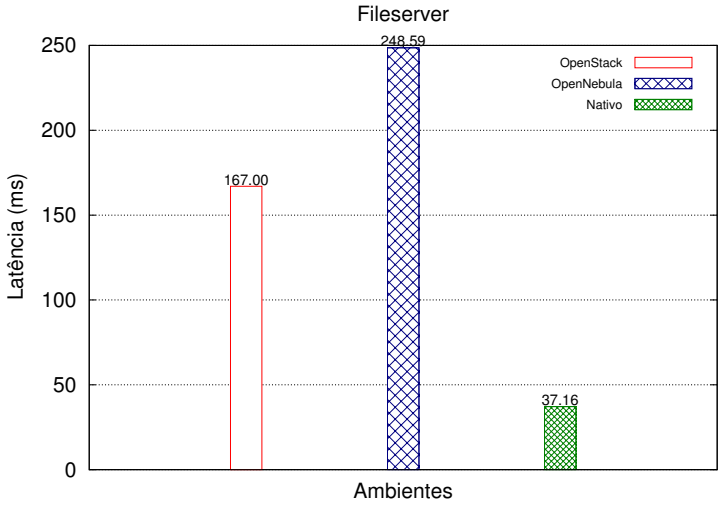

Figura 4.5: Resultados numéricos com servidor de arquivos

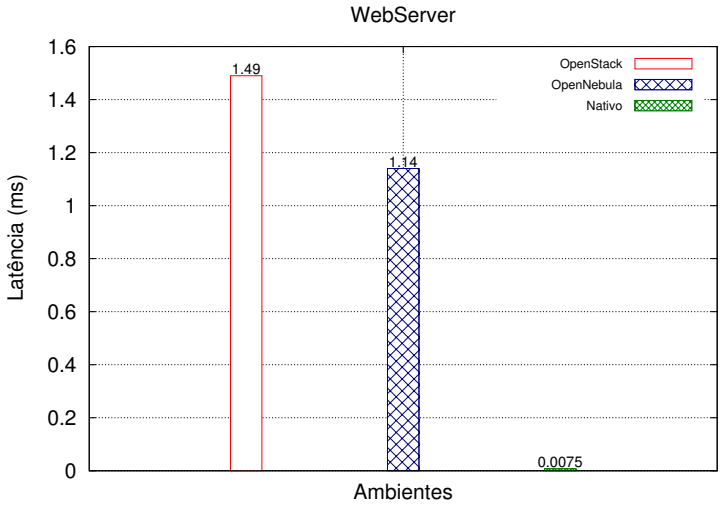

Figura 4.6: Resultados numéricos com servidor WEB

faz um reaproveitamento de dados em cache. Processo natural, implementado pelo próprio sistema operacional.

Nas Tabelas 4.7, 4.8 e 4.9 se encontram os resultados estatísticos.

Tabela 4.7: Resultados Estatísticos Nuvem vs Nuvem (Aplicações corporativas)

		OpenStack (β)		OpenNebula (Ω)		β *vs* Ω
Benchmark	Medida	$\bar{x}$ (40x)	σ (40x)	$\bar{x}$ (40x)	σ (40x)	*Sig.*
Fileserver	Latência	49,55	5,95	269,09	64,55	,000
Varmail	Latência	10,66	,74	141,48	14,46	,000
Webserver	Latência	,06	,11	,72	,50	,000

Tabela 4.8: Resultados estatísticos OpenStack vs Nativo (Aplicações Corporativas)

		OpenStack (β)		Nativo (π)		β *vs* π
Bench.	Medida	$\bar{x}$ (40x)	σ (40x)	$\bar{x}$ (40x)	σ (40x)	*Sig.*
Fileserver	Latência	49,55	5,95	40,53	2,44	,000
Varmail	Latência	10,66	,74	62,60	3,03	,000
Webserver	Latência	,06	,11	,0050	,02	,003

Tabela 4.9: Resultados estatísticos das aplicações corporativas (OpenNebula vs Nativo)

		OpenNebula (Ω)		Nativo (π)		Ω *vs* π
Bench.	Medida	$\bar{x}$ (40x)	σ (40x)	$\bar{x}$ (40x)	σ (40x)	*Sig.*
Fileserver	Latência	269,09	64,55	40,53	2,44	,000
Varmail	Latência	141,48	14,46	62,60	3,03	,000
Webserver	Latência	,72	,502	,0050	,02	,000

Os resultados estatísticos na Tabela 4.7, nos mostram que a ferramenta OpenStack alcançou os melhores resultados, sendo significativamente melhor. Podemos ver na Tabela 4.8, que o ambiente OpenStack foi significativamente melhor que o ambiente Nativo nos testes de simulação de um servidor e-mails, e em todos *kernels* melhor que OpenNebula. Já o OpenNebula, os resultados forma negativamente diferentes em relação ao Nativo.

4.4.1 Visão Geral

Nas seções anteriores foram apresentados resultados utilizando aplicações corporativas, simulando um servidor de e-mail, Web e de arquivos.

Nestes resultados, a latência de disco é um fator que predomina nos testes, e justamente por isso, OpenStack teve bons resultados na comparação, onde em alguns casos, sendo melhor que o ambiente Nativo como mostra os resultados do Varmail.

5. Conclusão

Este relatório apresentou resultados com testes específicos de isolamento de recursos, aplicações paralelas e corporativas nas ferramentas OpenStack e OpenNebula e ambiente Nativo, com o propósito de estudar a relação que cada ferramenta exerce sobre os resultados.

Com o estudo realizado, conclui-se que as ferramentas exercem influência nas execuções. Durante os testes buscou-se manter o ambiente homogêneo para não comprometer alguns resultados, porém, devido as características particulares de cada ferramenta, em alguns aspectos não foi possível manter esta semelhança, por exemplo: infraestrutura de rede (detalhes na seção 3)

O modo de implantação da ferramenta OpenStack, juntamente com a tecnologia de virtualização de disco, permitiu que a ferramenta alcançasse resultados positivos em relação ao OpenNebula e ao ambiente Nativo, favorecendo as aplicações corporativas. A média dos resultados de disco ficam entre 22% à 36% melhores. E nas aplicações corporativas de simulação de um servidor de e-mail OpenStack foi em média 487% melhor que o ambiente Nativo, um resultado surpreendente, que necessita de uma investigação mais afundo. E nos resultados de Fileserver e Webserver, ficaram a diferença favorecendo o ambiente Nativo, respectivamente ficam em 18% e 92%.

Contudo, os resultados de isolamento de recursos nos quesitos rede, memória e processador, e nas aplicações paralelas, mostram que a ferramenta OpenNebula é melhor que OpenStack. E podemos considerar que os 10% que OpenNebula foi melhor que OpenStack (memória, disco e processamento) são significativos.

Em relação as aplicações paralelas, o desempenho da rede apresentado no isolamento de recursos foi determinante para a vantagem da ferramenta OpenNebula nos testes com as aplicações MPI.

Entende-se que as aplicações que utilizam das bibliotecas MPI, criam processos em larga escala para comunicação de memória distribuída, e com bibliotecas OMP, o paralelismo é através de memória compartilhada. Sendo assim, os resultados numéricos mostram que as aplicações MPI em OpenNebula foram em média $8,6\%$ melhores e as aplicações usando OpenMP tiveram um benefício médio de 2%.

Os resultados aqui apresentados são o ponto de partida para uma melhor investigação sobre o desempenho de ambientes de nuvens IaaS, envolvendo as ferramentas usadas na pesquisa. Os testes foram iniciais, e é importante a partir do entendimento e experiência destes, aprofundar outros níveis de abordagens sobre as ferramentas, sendo que todos os passos seguidos para a implantação, levaram em conta o mínimo exigido para o funcionamento.

Os resultados são claros, as ferramentas exercem algum tipo de influência. Contudo, o ponto de partida para uma nova abordagem seria aprofundar o entendimento das APIs que envolvem cada ferramenta de nuvem. Com isso, poderia haver uma otimização das requisições das ferramenta aos virtualizadores para cargas de trabalhos específicas, e assim, conseguir alguns avanços nos tempos de execução.

Portanto, a escolha de uma ferramenta de administração apropriada, gera não somente uma capacidade de processamento maior ou tempo de processamento menor, mas interfere também em outros aspectos, permitindo que mais tarefas ou usuários possam executar suas aplicações. Além disso, podemos mencionar a diminuição do consumo energético, uma vez que há menos tempo de processamento/execução, e até mesmo redução dos gastos com a implantação da infraestrutura da nuvem, já que um número menor de unidades de processamento, usando OpenNebula, pode atingir uma mesma capacidade de processamento que uma infraestrutura maior, usando OpenStack.

A pesquisa até o momento teve um grande avanço no entendimento sobre as tecnologias

envolvidas, contribuindo para o meio acadêmico com 2 publicações aprovadas para eventos regionais (Ver Apêndice - ERAD e ERRC) e um submetido à um evento nacional (Ver Apêndice - SBRC).

Nossas intenções futuras é seguir com as mesmas metodologias de testes aplicadas neste trabalho, mas com outras ferramentas de administração de nuvem, e comparar os resultados com os aqui obtidos. Estamos buscando também mudar alguns aspectos no modo de implantação das ferramentas OpenNebula e Openstack, e assim buscar realizar os mesmos testes considerando o modo de implantação, principalmente com disco e rede. Utilizar outros tipos de *benchmarks* para análise do isolamento de recursos que utilizam bibliotecas de programação paralela. Avaliar outros *kernels* do NAS, que fazem um uso exaustivo de rede, se simulando I/O em disco em um ambiente de alto desempenho. E por fim, aprofundar outros testes com outros *benchmarks* de simulação de aplicações corporativas.

Referências Bibliográficas

[1] ABRAMS, R.: Bringing the cloud down to Earth. PlanningShop (2011)

[2] Amazon: Amazon EC2 <http://aws.amazon.com/pt/ec2//> (2015)

[3] Bailey, D.H., Barszcz, E., Barton, J.T., Browning, D.S., Carter, R.L., Dagum, L., Fatoohi, R.A., Frederickson, P.O., Lasinski, T.A., Schreiber, R.S., et al.: The nas parallel benchmarks. International Journal of High Performance Computing Applications 5(3), 63–73 (1991)

[4] Chapman, B., Jost, G., Van Der Pas, R.: Using OpenMP: portable shared memory parallel programming, vol. 10. MIT press (2008)

[5] Culler, D.E., Singh, J.P., Gupta, A.: Parallel computer architecture: a hardware/software approach. Gulf Professional Publishing (1999)

[6] Dongarra, J.J., Bunch, J.R., Moler, C.B., Stewart, G.W.: LINPACK users' guide, vol. 8. Siam (1979)

[7] Dynamics, D.: Demanda por serviços de virtualização cresce 70%, afirma pesquisa <http://www.datacenterdynamics.com.br/focus/archive/2015/01/demanda-por-servi%C3%A7os-de-virtualiza%C3%A7%C3%A3o-cresce-70-afirma-pesquisa> (January 2015), last access in January, 2015

[8] Engine, G.: Google Cloud Plataform <https://cloud.google.com/appengine/docs> (2015)

[9] Evangelinos, C., Hill, C.: Cloud Computing for Parallel Scientific HPC Applications: Feasibility of Running Coupled Atmosphere-Ocean Climate Models on Amazon's EC2. ratio 2(2.40), 2–34 (2008)

[10] Field, A.: Discovering Statistics Using SPSS. SAGE, Dubai, EAU (2009)

[11] Filebench: Filebench <http://filebench.sourceforge.net/wiki/index.php/Main_Page> (July 2015)

[12] Galisteu, R.: USP migra para a cloud computing privada (March 2013), http://www.itforum365.com.br/noticias/detalhe/3778/usp-migra-para-a-cloud-computing-privada

[13] Ganglia: What is Ganglia? <http://ganglia.sourceforge.net> (2015)

[14] Google: Google cloud platform (2013), disponível em: <https://cloud.google.com/>. Acessado em: 30 de Março de 2014.

[15] Gropp, W., Lusk, E., Thakur, R.: Using MPI-2: Advanced features of the message-passing interface. MIT press (1999)

[16] Gupta, A., Milojicic, D.: Evaluation of HPC Applications on Cloud. In: Open Cirrus Summit (OCS), 2011 Sixth. pp. 22–26. IEEE (October 2011)

[17] Hentges, E.L., Thomé, B.R.: Análise e Comparação de Ferramentas Open Source de Computação em Nuvem para o Modelo de Serviço IaaS. Master's thesis, Undergraduate Thesis, Sociedade Educacional Três de Maio (SETREM), Três de Maio, RS, Brazil (December 2013)

[18] Iozone: Iozone File System Benchmark (Official Page) <http://www.cs.virginia.edu/ mccalpin/papers/bandwidth/node1.html> (2006), last access in April, 2014

[19] Marinescu, D.: Cloud Computing: Theory and Practice. Elsevier, 225 Wyman Street, Waltham, 02451, USA (2013)

[20] Maron, C.A.F.: Avaliação e Comparação da Computação de Alto Desempenho em Ferramentas Opensource de Administração de Nuvem Usando Estações De Trabalho. Master's thesis, Undergraduate Thesis, Sociedade Educacional Três de Maio (SETREM), Três de Maio, RS, Brazil (August 2014)

[21] Maron, C.A.F., Griebler, D., Schepke, C.: Comparação das Ferramentas OpenNebula e OpenStack em Nuvem Composta de Estações de Trabalho. In: 14th Escola Regional de Alto Desempenho do Estado do Rio Grande do Sul (ERAD/RS). pp. 173–176. Sociedade Brasileira de Computação, Alegrete, RS, Brazil (March 2014)

[22] Maron, C.A.F., Griebler, D., Vogel, A., Schepke, C.: Avaliação e Comparação do Desempenho das Ferramentas OpenStack e OpenNebula. In: 12th Escola Regional de Redes de Computadores (ERRC). pp. 1–5. Sociedade Brasileira de Computação, Canoas (November 2014)

[23] Maron, C.A.F., Griebler, D., Vogel, A., Schepke, C.: Em Direção à Comparação do Desempenho das Aplicações Paralelas nas Ferramentas OpenStack e OpenNebula. In: 15th Escola Regional de Alto Desempenho do Estado do Rio Grande do Sul (ERAD/RS). Sociedade Brasileira de Computação, Gramado, RS, Brazil (March 2015)

[24] Mccalpin, J.: The Stream Benchmark) <http://iozone.org/> (1996), last access in April, 2014

[25] Mehrotra, P., Djomehri, J., Heistand, S., Hood, R., Jin, H., Lazanoff, A., Saini, S., Biswas, R.: Performance Evaluation of Amazon EC2 for NASA HPC Applications. In: Proceedings of the 3rd Workshop on Scientific Cloud Computing Date. pp. 41–50. ACM, Delft, The Netherlands (June 2012)

[26] Microsoft: Microsoft Azure <http://azure.microsoft.com/pt-br/> (2015)

[27] Navaux, P., Roloff, E., Diener, M., Carissimi, A.: High Performance Computing in the Cloud: Deployment, Performance and Cost Efficiency. In: Proceedings of the 2012 IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom). pp. 371–378. IEEE Computer Society, Washington, DC, USA (2012)

[28] OpenNebula: OpenNebula (Official Page) <http://opennebula.org/> (2014), last access in October, 2014

[29] OpenStack: OpenStack Roadmap <http://openStack.org/software/roadmap/> (2014), last access May, 2014

[30] Project, O.: Quickstart: OpenNebula 4.4 on Ubuntu 14.04 and KVM <http://docs.opennebula.org/4.6/design_and_installation/quick_starts/qs_ubuntu_kvm.html> (2015)

[31] Regola, N., Ducon, J.C.: Recommendations for Virtualization Technologies in High Performance Computing. pp. 409 – 416. IEEE, Indianapolis (November 2010)

[32] Strazdins, P.E., Cai, J., Atif, M., Antony, J.: Scientific Application Performance on HPC, Private and Public Cloud Resources: A Case Study Using Climate, Cardiac Model Codes and the NPB Benchmark Suite. In: Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International. pp. 1416–1424. IEEE (2012)

[33] Thome, B., Hentges, E., Griebler, D.: Computação em Nuvem: Análise Comparativa de Ferramentas Open Source para IaaS. In: 11th Escola Regional de Redes de Computadores (ERRC). p. 4. Sociedade Brasileira de Computação, Porto Alegre, RS, Brazil (November 2013)

[34] Toraldo, G.: OpenNebula 3: Cloud Computing. Packt, Birmingham (2013)

[35] World, C.: Nuvem privada deve ser opção de 76% de empresas no Brasil até 2019 <http://computerworld.com.br/tecnologia/2014/08/28/nuvem-privada-deve-ser-opcao-de-76-de-empresas-no-brasil-ate-2019/> (August 2014), last access in February, 2015

[36] Xavier, M., Neves, M., Rossi, F., Ferreto, T., Lange, T., Rose, C.D.: Performance Evalution of Container-based Virtualization for High Performance Computing Environments. In: 21st Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP). pp. 233–240. IEEE, Belfast, UK (March 2013)

Artigos Escritos

Este Capítulo apresenta as referências de todos os artigos publicados durante a realização do projeto HiPerfCloud no semestre de 2014.

Maron, C. A. F.; Griebler, D.; Vogel, A.; Schepke C.. Em Direção à Comparação do Desempenho das Aplicações Paralelas nas Ferramentas OpenStack e OpenNebula. 15th Escola Regional de Alto Desempenho do Estado do Rio Grande do Sul (ERAD/RS), 2015.

Maron, C. A. F.; Griebler, D.; Vogel, A.; Schepke C.. Avaliacão e Comparação do Desempenho das Ferramentas OpenStack e OpenNebula. 12th Escola Regional de Redes de Computadores (ERRC), 2014.

Maron, C. A. F.; Griebler, D.; Schepke C.. Comparação das Ferramentas OpenNebula e OpenStack em Nuvem Composta de Estações de Trabalho 14th Escola Regional de Alto Desempenho do Estado do Rio Grande do Sul (ERAD/RS), 2014.