UNIVERSIDADE ESTADUAL PAULISTA
“Júlio de Mesquita Filho”

Pós-Graduação em Ciência da Computação

Gustavo Cesar Bruschi

StackAct: Avaliação de Desempenho em uma Nuvem IaaS Multicamadas

SÃO JOSÉ DO RIO PRETO

2016

Gustavo Cesar Bruschi

StackAct: Avaliação de Desempenho em uma Nuvem IaaS Multicamadas

Orientador: Profa. Dra. Roberta Spolon

Dissertação de Mestrado elaborada junto ao Programa de Pós-Graduação em Ciência da Computação – Área de Concentração em Arquitetura de Computadores e Sistemas Distribuídos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

SÃO JOSÉ DO RIO PRETO

2016

Bruschi, Gustavo Cesar.
StackAct : avaliação de desempenho em uma nuvem IaaS multicamadas / Gustavo Cesar Bruschi. -- São José do Rio Preto, 2016
101 f. : il.

Orientador: Roberta Spolon
Dissertação (mestrado) – Universidade Estadual Paulista "Júlio de Mesquita Filho", Instituto de Biociências, Letras e Ciências Exatas

1. Computação. 2. Sistemas de computação virtual. 3. Computação em nuvem. 4. Desempenho - Medição. 5. Software gratuito. I. Spolon, Roberta. II. Universidade Estadual Paulista "Júlio de Mesquita Filho". Instituto de Biociências, Letras e Ciências Exatas. III. Título.

CDU – 681.3.025

Ficha catalográfica elaborada pela Biblioteca do IBILCE
UNESP - Câmpus de São José do Rio Preto

Gustavo Cesar Bruschi

StackAct: Avaliação de Desempenho em uma Nuvem IaaS Multicamadas

Área de Concentração: Arquitetura de Computadores e Sistemas Distribuídos
Linha de Pesquisa: Computação em Nuvem

Banca Examinadora:

_____

Profa. Dra. Roberta Spolon
Departamento de Computação
Faculdade de Ciências
Universidade Estadual Paulista – Bauru
(Presidente)

_____

Prof. Dr. José Remo Ferreira Brega
Departamento de Computação
Faculdade de Ciências
Universidade Estadual Paulista – Bauru

_____

Prof. Dr. Hélio Crestana Guardia
Departamento de Computação
Universidade Federal de São Carlos

Agradecimentos

Agradeço a Deus, pela vida, paz e tranquilidade em todos os momentos;

À minha esposa Francine, pelo apoio, paciência, compreensão e cumplicidade ao meu lado;

Ao meu filho Felipe e minha filha Júlia, por serem minhas fontes de energia;

Aos amigos Henrique Martins, Leandro Pauro, Luis Alexandre da Silva e Kelton Pontara da Costa, que me acompanharam e ajudaram nas dificuldades técnicas;

À professora Dra. Roberta Spolon pelo incentivo, pelos ensinamentos que me foram transmitidos, pela orientação desta monografia e por ter acreditado em mim.

Resumo

A Computação em Nuvem tornou-se sinônimo de qualidade e eficiência em investimento na área de Tecnologia de Informação, criando novos desafios para o processamento e integração de dados. O desempenho da solução adotada é um ponto chave para o sucesso de uma solução em Nuvem, assim como a maneira como as máquinas virtuais realizam a leitura e gravação no armazenamento, podem ser determinantes para uma melhor utilização desta solução. Este trabalho apresenta o StackAct, um mecanismo que permite realizar o monitoramento e obter dados, em uma Nuvem IaaS, relativos ao consumo de recursos computacionais de uma solução em três camadas utilizando orquestrador Apache CloudStack com hypervisor XenServer e armazenamento dos dados no sistema Openfiler. Foram realizados testes de desempenho utilizando três diferentes tipos de perfil de instâncias em uma nuvem computacional privada, possibilitando mensurar os consumos de CPU, E/S e Memória nas três camadas envolvidas, com diferentes tipos de ofertas de serviços. Os testes resultaram em um comparativo entre cada item analisado para cada camada individual, onde foi possível detectar uma pequena variação entre as diferentes configurações de testes devido a forma como a camada do hypervisor enfileira as requisições realizadas pela camada do orquestrador. Já no comparativo realizado entre as camadas, foi possível constatar o alto consumo de disco na camada de armazenamento de dados, em especial E/S de gravação de dados, que levaram a realização de outros testes utilizando disco de estado sólido na camada de armazenamento, tendo um grande impacto no desempenho da solução como um todo. Foi detectado também um alto consumo de memória na camada hypervisor, que é justificada pela alocação do próprio hypervisor além das VMs que estão sendo criadas e utilizadas no processo.

Palavras-chave: Computação em Nuvem, Desempenho, *opensource*, Apache *CloudStack, Xen Hypervisor.*

Abstract

Cloud Computing has become synonymous of quality, efficiency, and return of investment in Information Technology, creating new challenges for processing and data integrations. The performance of the adopted solution is a key to the success of a solution on Cloud, as well as the way that virtual machines use reading and writing to storage, which can be decisive for a better use of this solution. This work presents the StackAct, a mechanism that allows for monitoring and obtaining data on the consumption of computing resources of a solution in three layers using orchestrator IaaS Apache CloudStack with XenServer hypervisor and storage of data on the NAS OpenFiler system. Based on this mechanism, performance tests were conducted using three different instances of a private cloud. CPU, I/O, and memory usages in the three layers involved were measured with different types of loads. The tests resulted in a comparison between each item analyzed for each individual layer, and it was possible to detect a slight variation between the different configurations of tests because of the way that the hypervisor layer queues the requests made by the orchestrator layer. In the comparison made between the layers, it was possible to high consumption of disk in the data storage layer, in particular I/O data recording, which led to other tests using solid state disk in the storage layer, having an high impact on the performance of the solution as a whole. It was also detected high memory in the hypervisor layer, which is justified by the allocation of the hypervisor itself beyond the VMs being created and used in the process.

Palavras-chave: Cloud Computing, Performance, opensource, Apache CloudStack, Xen Hypervisor.

Lista de Figuras

Lista de Quadros

Lista de Abreviaturas

ACM - *Association for Computing Machinery*

AoE - *ATA over Ethernet*

API - *Application Programming Interface*

ASF - *Apache Software Foundation*

AWS - *Amazon Web Services*

CEO - *Chief executive officer*

CLC - *Cloud Controller*

CLI - *Command Line Interface*

CPU - *Central Processing Unit*

CSV - *Comma-separated Values*

E/S - Entrada/Saída

EC2 - *Elastic Compute Cloud*

GPL - *General Public License*

HPC - *High Power Computing*

IaaS – *Infrastructure as a Service*

IEEE - *Institute of Electrical and Electronics Engineers*

IP – *Internet Protocol*

iSCSI - *Internet Small Computer System Interface*

ISO - *International Organization for Standardization*

KSAR - *Graphical SAR Report Analyzer*

KVM - *Kernel-based Virtual Machine*

LAN - *Local Area Network*

LVM - *Logical Volume Manager*

MHZ - *Megahertz*

NC - *Node Controller*

NFS - *Network File System*

NIST - *National Institute of Standards and Technology*

PaaS - *Plataform as a Service*

PDF - *Portable Document Format*

QCOW2 - *QEMU Copy On Write*

RAM - *Random Access Memory*

SaaS - *Software as a Service*

SAN - *Storage Area Network*

SAR - *System Activity Reporter*

SAS - *Serial Attached SCSI*

SATA - *Serial ATA*

SC - *Storage Controller*

SCSI - *Short for Small Computer System Interface*

SLA - *Service Level Agreement*

SO - Sistema Operacional

SSD - *Solid-state Drive*

SSH - *Secure Shell*

TI - Tecnologia da Informação

TLP - *Top Level Project*

VM - *Virtual Machines*

VMDK - *Virtual Machine Disk*

VMFS - *Virtual Machine File System*

VPN - *Virtual Private Network*

Sumário

CAPÍTULO 1 – Introdução

1.1 Considerações Iniciais

A Computação em Nuvem, ou *Cloud Computing*, tornou-se um paradigma popular da computação durante a última década, aliando a ideia do provisionamento de recursos computacionais com a real necessidade para uma aplicação de software, sendo oferecido como serviço na Internet por terceiros que são chamados de provedores de nuvem. Assim como a água e a energia elétrica nas residências e empresas, onde o consumidor paga pelo que consome, a Computação em Nuvem é oferecida pelos provedores de nuvem no modelo do consumo sob demanda - o cliente paga pelo o que consome.

A Computação em Nuvem é um modelo para permitir, de forma conveniente, o acesso à rede sob demanda a um *pool* compartilhado de recursos configuráveis de computação, como redes, armazenamento, servidores e serviços, que podem ser rapidamente provisionados e liberados com um esforço mínimo de gerenciamento ou interação com o provedor de serviços (MELL; GRANCE, 2011). Entre as diversas vantagens, estas soluções oferecem *pools* de recursos virtualizados, pagos no formato "pague o que usar", reduzindo drasticamente o investimento inicial e custos de manutenção.

A Computação em Nuvem é frequentemente associada com a oferta de novos mecanismos que permitem aos fornecedores dar aos usuários acesso a um número praticamente ilimitado de recursos (*Resource Outsourcing*), conforme apresentado na Figura 1. Garantias são oferecidas pelo provedor de nuvem por meio de contrato de serviço sob medida: *Service Level Agreements* (SLA) ou Termo de Aceitação de Serviço. (KEITH; BURKHARD, 2010)

Figura 1 – Uso da Computação em Nuvem.

FONTE: VERAS, 2012 (adaptado pelo autor)

Basicamente, a Computação em Nuvem é uma combinação de tecnologias conhecidas que estavam disponíveis no início dos anos 90, como *grid computing* ou computação em grade, e a virtualização. Cada uma dessas tecnologias, aliada ao modelo comercial adotado pelos provedores de nuvem, formam o componente essencial que permite que o usuário final pague apenas pelos recursos que consumiram, além de conseguir uma elasticidade de recursos conforme sua demanda - ou seja, crescer ou diminuir seu ambiente tecnológico conforme sua necessidade. (SINGH et al., 2014)

Atualmente, muitas das grandes empresas do setor de Tecnologia de Informação oferecem soluções em nuvem, como por exemplo, a Amazon EC2[1], o Microsoft Azure[2], o Google Apps[3] e o IBM Blue Cloud[4]. Além disso, há um número crescente de soluções de código aberto para construir nuvens privadas, públicas e híbridas como OpenNebula[5], Eucalyptus[6],

[1] O Amazon Elastic Compute Cloud é um serviço da web que fornece capacidade de computação redimensionável na nuvem (http://aws.amazon.com/pt/ec2).
[2] O Microsoft Azure é uma plataforma especial para execução de aplicativos e serviços, baseada nos conceitos da computação em nuvem (http:// www.microsoft.com/Azure).
[3] Google Apps for Work é um serviço do Google que oferece versões de vários produtos Google que podem ser personalizados de forma independente (http:// apps.google.com/work/apps).
[4] IBM cloud computing é o serviço de nuvem oferecido pela IBM (http:// www.ibm.com/cloud-computing/br/pt/).
[5] OpenNebula é uma plataforma de Computação em Nuvem para o gerenciamento de infraestruturas de ambientes computacionais distribuídos heterogêneos (http://opennebula.org).
[6] Eucalyptus é um software livre de código aberto para a construção de uma Nuvem Computacional em IaaS (http://www8.hp.com/us/en/cloud/helion-eucalyptus-overview.html).

OpenStack[7] e Apache CloudStack[8]. Em todas estas soluções, a maneira como as máquinas virtuais (VM) são instanciadas tem efeitos importantes sobre as requisições de Entrada e Saída (E/S) dos discos que compõe o armazenamento da VM.

1.2 Motivação e Justificativa

O crescente aumento na utilização de soluções em Nuvem por médias e grandes organizações já seria uma motivação para se aprofundar no estudo sobre esta tecnologia. Pesquisa realizada pela Cisco[9] e a IDC[10] em 2015, aponta para a chegada de uma "segunda onda" de adoção de Computação em Nuvem pelas companhias, onde a visão tradicional de nuvem como forma de aumentar a eficiência e reduzir custos deverá ser substituída por um foco mais amplo no negócio, voltado à inovação e ao crescimento da companhia. Outra pesquisa, realizada pela Capgemini em 2014, indicou que até o ano de 2019, a nuvem privada seja a opção de 76% dos CIOs (*Chief Information Officer*) entrevistados, deixando o modelo público "puro" com apenas 17% da preferência.

Porém, a maior motivação para este trabalho está associada ao número reduzido de pesquisas relacionadas ao desempenho em ambientes de Infraestrutura como Serviço em Computação em Nuvem, principalmente utilizando o orquestrador Apache CloudStack. Além do orquestrador não possuir uma ferramenta nativa que realize estas funções, como ocorre com o OpenStack, até o desenvolvimento do trabalho existiam apenas pesquisas limitadas relacionadas ao desempenho de nuvens com Apache CloudStack. Não há, até o presente momento, nenhuma pesquisa que desenvolvesse um método capaz de avaliar diversos aspectos relacionados a consumo de recursos em um ambiente com as 3 camadas definidas no escopo deste trabalho (orquestrador, *hypervisor* e armazenamento).

---

[7] OpenStack é um software de código aberto, capaz de gerenciar os componentes de múltiplas infraestruturas virtualizadas (http://www.openstack.org).

[8] CloudStack é um software de computação em nuvem de código aberto para a criação, gerenciamento e implementação de serviços de nuvem em IaaS (http://cloudstack.apache.org/).

[9] Cisco Systems é uma companhia multinacional de soluções para redes e comunicações (http://www.cisco.com/c/pt_br).

[10] IDC é a empresa líder em inteligência de mercado e consultoria nas indústrias de tecnologia da informação, telecomunicações e mercados de consumo em massa de tecnologia (http://br.idclatin.com).

Outra motivação foi o crescimento no interesse pelo Apache CloudStack principalmente após ter se tornado um *Top-Level Project* (TLP) da Fundação Apache, em 2013.

1.3 Objetivos

Este trabalho tem como objetivos:

- Apresentar um estudo ao conceito de Computação em Nuvem com ênfase no modelo IaaS, a fim de analisar esse modelo computacional. Diante desta análise foram extraídos os principais pontos relevantes, bem como os problemas apresentados;
- Apresentar o StackAct, um mecanismo capaz de extrair e armazenar informações de consumo de recursos computacionais durante a utilização de uma nuvem IaaS criada com Apache CloudStack na camada de orquestração, XenServer na camada de virtualização e Openfiler na camada de armazenamento de dados;
- Validar o mecanismo desenvolvido em um ambiente IaaS criado com propósito de pesquisa, possibilitando a análise dos dados obtidos.

1.4 Organização do Texto

Este estudo está organizado em 3 capítulos, descritos a seguir:

- O Capítulo 2 apresenta uma revisão da literatura sobre o conceito de Computação em Nuvem, a importância da Virtualização para esses ambientes, selecionando alguns dos principais *hypervisors* utilizados em solução IaaS. São apresentados também alguns dos principais orquestradores de solução para Computação em Nuvem IaaS;

- O Capítulo 3 apresenta o StackAct, mecanismo desenvolvido para mensurar o consumo de recursos computacionais de uma solução em três camadas utilizando orquestrador IaaS Apache CloudStack com hypervisor XenServer e o

armazenamento dos dados no sistema Openfiler. São apresentados também fatores importantes de desempenho e métricas em Computação em Nuvem.

- Baseado no mecanismo apresentado, no capítulo 4 são apresentados os resultados dos testes de desempenho utilizando três diferentes tipos de perfil de instâncias em uma nuvem computacional privada, onde foi possível mensurar os consumos de CPU, E/S e Memória nas três camadas envolvidas, com diferentes tipos de cargas de instâncias;
- O Capítulo 5 apresenta as conclusões da pesquisa, contribuições realizadas e apresenta propostas para trabalhos futuros.

CAPÍTULO 2 – Revisão Bibliográfica

2.1 Considerações Iniciais

A Computação em Nuvem é um modo de computação distribuída criada após a Computação em Grade *(grid computing)* e computação ubíqua *(pervasive computing)*. A Computação em Nuvem é definida principalmente como um novo padrão e paradigma promissor da computação, que abriga enormes quantidades de dados e os serviços em *datacenters* escaláveis que podem ser acessados a partir de qualquer dispositivo conectado na Internet (GOYAL, 2014).

No mundo todo, as empresas têm se esforçado para reduzir os custos relacionados a tecnologia, e por essa razão a maioria destas empresas começam consolidando suas operações de T.I., normalmente tendo como ação inicial a virtualização de seus servidores. A Computação em Nuvem permite a estas empresas a busca a um novo nível que lhes permite reduzir ainda mais os custos através da otimização na utilização de recursos, administração reduzida e custos de infraestrutura, além de ciclos mais rápidos de implantação (BOSS et al., 2013).

A Computação em Nuvem é uma tecnologia que é suscetível a um desenvolvimento contínuo, devido a suas fortes vantagens de acesso a dados a partir de qualquer lugar do mundo através da Internet, sem se preocupar com a infraestrutura utilizada e os problemas envolvidos pelos processos de instalação e manutenção de um ambiente tradicional. (LONEA et al., 2012)

A Computação em Nuvem é definida pelo Instituto Nacional de Padrões e Tecnologia (NIST, 2011) como um modelo para permitir, de forma conveniente, o acesso a rede sob demanda a um pool compartilhado de recursos configuráveis de computação como redes, armazenamento, servidores e serviços, que podem ser rapidamente provisionados e liberados com um esforço mínimo de gerenciamento ou interação com o provedor de serviços. Uma das principais vantagens é que ela oferece a possibilidade de pagar somente pelos serviços que são utilizados, uma ideia que foi prevista por John McCarthy em 1961: "a Computação pode um dia ser organizado como uma utilidade pública".

PRASAD (2014) define que:

- Um usuário em nuvem (*cloud user*) é uma pessoa ou uma organização (como uma PME - Pequenas e Médias Empresas), que utilizam os serviços em nuvem;
- Um fornecedor de nuvem (*cloud vendor*) é uma organização que vende serviços em nuvem para usos diversos;
- Um *software* de nuvem (*cloud broker*) é um *middleware* que interage com os prestadores de serviços em nuvem, tornando-se responsável por realizar as configurações do usuário adequadamente, e para a aquisição de novos recursos.

Computação em Nuvem também pode significar simplesmente Computação na Internet, pois geralmente a Internet é vista como coleção de nuvens. Desta forma, o termo Computação em Nuvem pode ser definida como a utilização da Internet para prover tecnologia, habilitando serviços para as pessoas e organizações, permitindo aos consumidores acessar recursos *on-line*, através da Internet, de qualquer lugar, a qualquer momento, sem se preocupar com questões técnicas e físicas de gestão e manutenção dos recursos computacionais. Além disso, os recursos de Computação em Nuvem são dinâmicos e escaláveis. (SHAIKH, 2014)

A Computação em Nuvem é totalmente diferente da computação em grade *(grid computing)* e computação utilitária *(utility computing)*, mas pode ser entendida como uma evolução destes conceitos. O Google Apps[11] é um excelente exemplo de Computação em Nuvem, que permite acessar serviços através do browser e o implanta em milhões de máquinas através da Internet. Os recursos são acessíveis a partir de qualquer momento e qualquer lugar em todo o mundo usando a Internet. A Computação em Nuvem é também conhecida como T.I. sob demanda e esta escalabilidade, que é o ponto chave da Computação em Nuvem, é conseguido através da virtualização de servidores. (SHAIKH, 2014)

2.2 História da Computação em Nuvem

A cada dia a Computação em Nuvem possui mais usuários, mesmo estes não compreendendo exatamente que estão acessando um serviço computacional na Nuvem. Segundo XIAODONG (2014), a princípio, o termo "nuvem" foi usado para descrever grandes redes de caixas eletrônicos na década de 1990. A partir da década de 90, uma grande mudança foi observada devido ao surgimento da Internet e ao aumento da velocidade para conexões de Internet.

[11] Google Apps é um serviço do Google que oferece versões de vários produtos Google que podem ser personalizados de forma independente com o nome de domínio do cliente (http:// apps.google.com/intx/pt-BR).

Porém a origem do termo Computação em Nuvem não é totalmente clara. A expressão nuvem é comumente utilizada em ciência para descrever uma grande aglomeração de objetos que aparecem visualmente a partir de uma distância como uma nuvem, e descreve qualquer conjunto de coisas cujos detalhes não são inspecionados em um determinado contexto (XIAODONG, 2014). Uma outra explicação é que alguns *softwares* para desenhar ambientes de rede de computadores utilizavam ícones para servidores com um círculo, e um cluster de servidores - em um diagrama de rede - tinha vários círculos que se sobrepunham, que pareciam uma nuvem. (SCHIMIDT; JONATHAN, 2014). A partir de 1994 o símbolo de uma nuvem passou a ser utilizado para representar a Internet.

O final dos anos 90 e início dos anos 2000 foi um período para investir em empresas baseadas na Internet. Arquiteturas *multi-tenant*, a grande largura de banda e padrões universais de interoperabilidade de software foram fatores essenciais para o surgimento da Computação em Nuvem (SALESFORCE, 2014). Um dos primeiros marcos na história da Computação em Nuvem foi a chegada da Salesforce.com em 1999, que foi pioneira no conceito de fornecimento de aplicativos empresariais através de um *website*. Esta acabou abrindo caminhos tanto para especialistas de T.I., quanto para empresas de *software* que queriam oferecer seus *softwares* via Internet.

Em 2002 a Amazon Web Services passou a oferecer um conjunto de serviços baseados em nuvem, incluindo armazenamento, computação e plataforma, através do Amazon Mechanical Turk. A Amazon até hoje é referência em Computação em Nuvem em todo o mundo, principalmente após 2006 por lançar sua Elastic Compute Cloud (EC2) como um serviço web comercial que permite que usuários de pequenas empresas aluguem computadores nos quais são capazes de executar suas próprias aplicações.

Até 2009, a maioria dos formadores de opinião da indústria de Tecnologia da Informação já investiam na Computação em Nuvem, com empresas como Microsoft e Google, oferecendo aplicativos para o consumidor, bem como empresas sob a forma de serviços simples e acessíveis. (SALESFORCE, 2014)

2.3 Principais Características da Computação em Nuvem

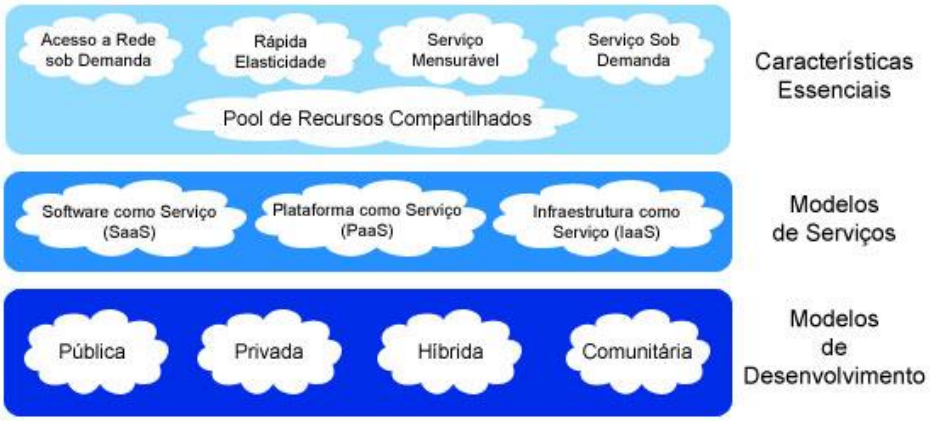
Segundo NIST (2011), há vários tipos de serviços oferecidos na Computação em Nuvem, que podem ser agrupados em três tópicos principais: *Software* como Serviço (SaaS), Plataforma como Serviço (PaaS) e Infraestrutura como Serviço (IaaS):

- ***Software* como Serviço (*SaaS*):** As aplicações são executadas nos provedores de serviços, provisionando para o consumidor em uma infraestrutura de nuvem. Os clientes podem acessar essas aplicações através de vários dispositivos e interfaces, como um navegador da *web*, dispositivos móveis e *desktops* (MELL; GRANCE, 2011). O consumidor não possui o controle, não dá manutenção ou administra qualquer item da infraestrutura de nuvem, incluindo sistema operacional, rede e armazenamento. No entanto, existe a possibilidade de customizar as definições específicas do usuário para o aplicativo. Os principais benefícios da adoção deste modelo são: redução de custos, agilidade, acessibilidade, flexibilidade e continuidade. (GUN, 2012). Segundo NIST (2011), o consumidor não administra ou controla a infraestrutura básica, incluindo nuvens de rede, servidores, sistemas operacionais, armazenamento, ou mesmo capacidades de aplicação individual, com a possível exceção de limitada aplicação específica e definições de configuração de utilizadores;
- **Plataforma como Serviço (*PaaS*)**: esta classe de serviço oferece para o consumidor a possibilidade de implantar seus próprios aplicativos criados ou adquiridos para uma infraestrutura de nuvem. Os consumidores não possuem o controle ou gerenciamento da infraestrutura em nuvem, exceto das definições de configuração para um ambiente de hospedagem de aplicativos implantados em uma nuvem (ALI, 2014). O modelo *PaaS* possibilita o desenvolvimento e a implantação de aplicativos entregues como um serviço para desenvolvedores através da *Web*. Ele facilita o desenvolvimento e a implantação de aplicativos sem os custos e a complexidade de uma infraestrutura tradicional, fornecendo todos os equipamentos necessários para suportar o ciclo de vida completo de construção e entrega de aplicações web e serviços totalmente disponíveis na Internet. Esta plataforma consiste em *software* de infraestrutura, e normalmente inclui um banco de dados, um *middleware* e ferramentas de desenvolvimento (BHARDWAJ et al., 2013);
- **Infraestrutura como Serviço (*IaaS*)**: de forma simplificada, é a entrega de *hardware* (processador, memória, armazenamento e rede) e *software* (sistemas operacionais de tecnologia de virtualização, sistema de arquivos) como um serviço. É uma evolução da hospedagem tradicional, que normalmente não requer nenhum

compromisso contratual em longo prazo e permite aos usuários recursos de provisão sob demanda, ou seja, elástico. (BHARDWAJ et al., 2013). O utilizador não tem o controle da infraestrutura física, mas através de ferramentas de virtualização, tem controle sobre os sistemas operacionais, armazenamento, aplicações instaladas e, possivelmente, um controle limitado dos recursos de rede. (NIST, 2011).

A Figura 2 ilustra as principais utilizações da Computação em Nuvem, características e aplicações nos modelos descritos anteriormente. A modalidade de Computação em Nuvem que é foco deste estudo é a Infraestrutura como Serviço, que será abordado mais detalhadamente no próximo tópico.

Figura 2 – Modelos e Características da Computação em Nuvem.



FONTE: LUO ET. AL., 2012 (adaptado pelo autor)

NIST (2011) também determina que a Computação em Nuvem é um modelo composto por cinco características essenciais, descritas nos seguintes tópicos:

- **Autoatendimento sob demanda:** O usuário pode adquirir unilateralmente recurso computacional, como tempo de processamento no servidor ou armazenamento na rede na medida em que necessite e sem precisar de interação humana com os provedores de cada serviço. Dentro de uma nuvem, o *hardware* e o *software* podem ser automaticamente reconfigurados. Estas modificações são apresentadas de forma transparente para os usuários, que possuem perfis diferentes e assim podem personalizar os seus ambientes computacionais, por exemplo, a configuração de rede para a definição de determinados privilégios, instalação de algum *software*;

- **Amplo acesso à rede:** Os recursos computacionais encontram-se disponíveis através da internet e podem ser acedidos através de mecanismos padronizados, que possibilitem o uso por plataformas heterogêneas, tais como, por exemplo, telefones móveis, *tablets*, portáteis, computadores pessoais ou outras tecnologias. A interface de acesso a nuvem não obriga os usuários a mudarem suas condições e ambientes de trabalho, como por exemplo, linguagens de programação e sistema operacional;
- ***Pooling* de recursos:** Os recursos computacionais do provedor são organizados em um *pool* para servir múltiplos usuários usando um modelo *multi-tenant* ou multiinquilino, com diferentes recursos físicos e virtuais, dinamicamente atribuídos e ajustados de acordo com a demanda dos usuários. Existe assim um senso de independência local em que o cliente geralmente não tem nenhum controle ou conhecimento sobre a localização exata dos recursos disponibilizados, mas pode ser capaz de especificar o local em um nível maior de abstração (por exemplo, país, estado ou do *data center*). Exemplos de recursos incluem o armazenamento, processamento, memória, largura de banda de rede e máquinas virtuais;
- **Elasticidade rápida:** Recursos podem ser adquiridos de forma rápida e elástica, em alguns casos automaticamente, caso haja a necessidade de escalar com o aumento da demanda, e liberados, na retração dessa demanda. Para o utilizador tudo deve ser transparente, de modo a criar a sensação que os seus recursos são ilimitados e que podem ser adquiridos em qualquer quantidade e a qualquer momento. Deste modo, o utilizador só paga aquilo que consume em cada momento, evitando assim custos desnecessários;
- **Serviços mensuráveis:** Os sistemas de gestão utilizados para a nuvem, controlam, monitorizam e otimizam automaticamente o uso de recursos, em cada tipo de serviço (processamento, armazenamento, largura de bando e ativos). O uso de recursos pode ser monitorado e controlado, possibilitando transparência para o provedor e o usuário do serviço utilizado.

Para tornar este modelo possível, é necessário reunir todas as aplicações e dados dos usuários em grandes centros de armazenamento, conhecidos como *data centers*. Uma vez reunidos, a infraestrutura e as aplicações dos usuários são distribuídas na forma de serviços disponibilizados por meio da Internet. (SILVA, 2010).

2.4 A Computação em Nuvem e a Virtualização de Servidores

Em um modelo computacional tradicional, cada *host* possui apenas um sistema operacional, podendo oferecer um ou mais serviços de rede. Segundo a VMWARE (2012), a média de consumo de recursos computacionais (processamento, memória, rede e armazenamento) em um ambiente não consolidado é de 10% a 15%. Além disso, o *deploy* de um novo ambiente utilizando este modelo tende a ser mais complexo, já que tudo deve ser realizado manualmente. Alta disponibilidade e balanceamento de carga também tendem a ser mais complexos, dependendo de componentes extras ao ambiente.

Já em um modelo computacional virtualizado, apenas um *host* pode hospedar diversas máquinas virtuais com sistemas operacionais diferentes. Além da utilização de recursos consolidados, este modelo tende a ter menor complexidade para realizar um *deploy*, possibilitando a criação de novos ambientes a partir de modelos (*templates*) já instalados previamente.

Segundo SALAPURA (2012), o primeiro sistema de computador com a virtualização de sistema foi lançado em 1972, quando a IBM lançou o Servidor Virtual Facility/370 (VM/370). Pela primeira vez foi possível particionar um único *mainframe* em várias máquinas virtuais isoladas, utilizando os recursos da máquina física em diversas máquinas virtuais. Mas foi no final da década de 90 que as técnicas de virtualização se tornaram amplamente disponíveis para sistemas x86, com base em soluções de software para superar as limitações intrínsecas na arquitetura Intel x86 que anteriormente impediam a virtualização nesta arquitetura.

A Virtualização de servidores é uma maneira prática e rápida utilizada na criação dos recursos computacionais utilizados na Computação em Nuvem. Sua principal característica é utilizar, gerenciar e prover de maneira otimizada a capacidade de dispositivos físicos evitando assim a sua subutilização. Apesar de muitos acreditarem que ela é aplicada apenas a servidores, este conceito pode ser estendido para *desktops*, aplicações e perfis de usuário. (VERAS, 2012).

A virtualização é considerada a noção primária da Computação em Nuvem, além de ser um meio de fornecer um ambiente de computação que fisicamente não existe, mas que na verdade é gerado em um ambiente de hospedagem. A Nuvem utiliza-se de conceitos existentes, como virtualização e emulação, e a partir disto, monta-se uma nova arquitetura reunindo-os para atingir seu objetivo, disponibilizando recursos conforme demanda. Segundo (BUYYA et al., 2011) a virtualização de *hardware* permite criar múltiplas máquinas virtuais sobre uma máquina física.

Em um ambiente de Computação em Nuvem, a camada de infraestrutura virtualizada envolve máquinas virtuais, armazenamento de dados e elementos de rede virtualizada, usados para desenvolver a estrutura sobre a qual a plataforma de computação pode ser estabelecida.

A virtualização também ajuda na característica relacionada com a rápida elasticidade na Computação em Nuvem, criando várias instâncias de recursos requisitados utilizando um único recurso real (ABOULNAGA ET AL., 2009). Além disso, a virtualização é uma maneira de abstrair características físicas de uma plataforma computacional dos usuários, exibindo outro *hardware* virtual e emulando um ou mais ambientes que podem ser independentes.

Um dos principais benefícios da utilização de virtualização é a capacidade de hospedar vários ambientes de sistema operacional que estão completamente isoladas umas das outras na mesma máquina física. Outro benefício é a capacidade de configurar a virtualização para utilizar diferentes divisões de recursos na mesma máquina física. Por exemplo, em uma máquina física, uma máquina virtual pode ser atribuída 10% do poder de processamento, enquanto a outra máquina virtual pode ser atribuída 20% do poder de processamento. (BUYYA et al., 2008)

Os termos virtualização e Computação em Nuvem são comumente confundidos e utilizados para a mesma definição, o que é um equívoco. Essencialmente a virtualização é diferente de Computação em Nuvem porque a primeira é obtida através de um *software* que manipula *hardware*, enquanto a segunda refere-se a um serviço que é realizado através desta manipulação (ANGELES, 2014). A maior parte desta confusão ocorre porque a virtualização e a Computação em Nuvem trabalham em conjunto para proporcionar diferentes tipos de serviços, como é o caso das nuvens privadas.

Existem no mercado diversas soluções para máquinas virtuais, sendo a maior parte soluções *opensource* e suas diferenças se destacam na gama de serviços oferecidos. Os principais *hypervisors*, que são os sistemas capazes de criar e manipular máquinas virtuais, são XEN[12], VMware[13], KVM[14] e QEMU[15]. Este último, diferentemente de um virtualizador, é um emulador, em que o *hardware* é totalmente simulado através de *software* (BUYYA et al, 2011).

A Virtualização é usada extensivamente em Nuvens IaaS, e através de seus recursos de crescimento e diminuição de recursos de forma dinâmica para atender a demanda de recursos

[12] Xen é um software livre de virtualização para as arquiteturas x86, x86-64, IA-32, IA-64 e PowerPC (http://www.xenproject.org).

[13] VMware é um software/máquina virtual que permite a instalação e utilização de um sistema operacional dentro de outro dando suporte real a softwares de outros sistemas operativos (http://www.vmware.com).

[14] KVM (Kernel Based Virtual Machine) é uma infraestrutura de virtualização, integrada ao Linux que suporta virtualização nativa usando Intel VT ou AMD-V (http://www.linux-kvm.org).

[15] QEMU é um software livre que implementa um emulador de processador, permitindo uma virtualização completa de um sistema PC dentro de outro (http://www.qemu.org).

por parte dos consumidores em nuvem, a estratégia básica da virtualização é a criação de máquinas virtuais independentes (VM) que são isoladas (DILLON; CHANG, 2010). Note-se que esta estratégia é diferente do modelo *multi-tenancy*, que visa transformar a arquitetura de *software* aplicativo de modo que várias instâncias (de vários clientes de nuvem) possam ser executadas em uma única aplicação, ou seja, a mesma máquina lógica.

De uma perspectiva técnica, a Computação em Nuvem traz os benefícios da virtualização e *multi-tenancy* para sistemas computacionais tradicionais. Técnicas de virtualização permitem que várias imagens do sistema compartilhem os mesmos recursos de hardware. O *Multi-tenancy* permite que vários usuários diferentes compartilham o mesmo *hardware*, proporcionando forte isolamento entre as cargas de trabalho para garantir a segurança e privacidade dos dados (SALAPURA, 2012).

Ainda segundo SALAPURA, a tecnologia de virtualização é a chave que possibilita a Computação em Nuvem, permitindo ultrapassar a visão estática de um sistema como uma peça de *hardware*, mas os sistemas do ponto de vista como entidades dinâmicas.

As principais plataformas de Infraestrutura como Serviço utilizam diversos *hypervisors* para virtualização de sua infraestrutura, que são citados a seguir.

2.4.1 KVM

O KVM foi desenvolvido e patrocinado por Qumranet em Israel e teve seu código submetido para a comunidade do kernel do Linux em dezembro de 2006. Em janeiro de 2007, quando foi lançada a versão 2.6.20 do kernel, o KVM foi integrado ao kernel (KVM, 2012). Isso se deve ao fato do KVM ser uma ferramenta extremamente otimizada, com aproximadamente 10000 linhas de código, e ainda capaz de tornar o próprio kernel do Linux em um *hypervisor* (REGOLA; DUCOM, 2010).

A inclusão do KVM no kernel do Linux é uma das vantagens apresentadas por essa tecnologia, devido ao fato do KVM passar a ter seu desenvolvimento apoiado pela comunidade Linux e também por alguns dos principais fornecedores de software da indústria, como Red Hat, AMD, HP, IBM, Intel, Novell, Siemens, SGI, entre outros (REDHAT, 2009). Como técnica de virtualização, o KVM utiliza a virtualização total para suportar os sistemas convidados. Ele suporta como sistema convidado a maioria dos SOs, entre eles Fedora/Red Hat e derivados, Ubuntu/Debian e derivados, outras distribuições Linux (Slackware, Suse, Android, Arch Linux, Gentoo, etc), BSD, Solaris/OpenSolaris, Windows (KVM, 2012).

2.4.2 Vmware ESX

Segundo a VMWARE (2010), criado no ano de 1999, o VMware ESX é um dos aplicativos de virtualização para plataforma x86 mais populares atualmente. Ele foi a primeira solução de virtualização para a arquitetura x86 e fornece uma implementação completa para o sistema convidado. A VMware Inc., empresa que desenvolve o *software* de mesmo nome, possui uma vasta linha de produtos voltados para a virtualização.

A VMware também possui algumas versões de seus produtos sem custo de licenciamento. Estas versões, normalmente, possuem algum tipo de limitação (VMWARE, 2010). No VMware ESX, o monitor emula algumas instruções, denominadas instruções sensíveis, objetivando representar corretamente o processador virtual para cada máquina hospedada. Para executar essas instruções sensíveis, a máquina virtual utiliza o mecanismo de interrupção de *software* (*trap*) do processador (VMWARE, 2010), com intuito de executar as instruções em multitarefa.

No VMware ESX, o próprio sistema hóspede gerencia a memória e, para garantir que não ocorrerá a tentativa de utilização do mesmo endereço de memória pelo sistema hóspede e pelo sistema hospedeiro, o VMware reserva uma parte da memória para seu uso exclusivo. É essa memória reservada previamente que é disponibilizada ao sistema hóspede (POLLON, 2008). Dentre os produtos desenvolvidos pela VMware Inc., destacam-se: o VMware ESX Server que é voltado para servidores de grande porte; o VMware Server, disponibilizado gratuitamente a partir de 2006, que é indicado para aplicação em ambientes de pequeno porte (VMWARE, 2010).

2.4.3 XEN

Segundo Ideler (2012), o Xen foi lançado em 2003 por um grupo de pesquisa em Cambridge, que possui nativamente hypervisor paravirtualizado e amplamente utilizado em implantações de nuvem pública. Na paravirtualização, segundo Galdino (2012), o sistema operacional da máquina virtual sabe que está rodando num ambiente virtualizado. Há um elemento, chamado *Hypervisor*, que podemos dizer tratar-se de uma API para que o sistema operacional *guest* tenha acesso ao hardware. Este *hypervisor* foi o primeiro a introduzir a técnica de paravirtualização e é até hoje a mais popular implementação nessa técnica

(CHAUDHARY et al., 2008). Ele foi criado por KeirFraser e Ian Pratt como parte do projeto de pesquisa de Xen Server (HAND et al., 2003) na Universidade de Cambridge na década de 1990 (XEN, 2012).

Ainda segundo Ideler, o *hypervisor* Xen foi desenvolvido com a filosofia que o *hypervisor* deve ser uma camada fina e mínima, devendo ter impacto mínimo para a solução como um todo. Devido a esta decisão de design, certos trabalhos de virtualização são delegados a um nível acima do *hypervisor*, ou seja, o S.O. Desta forma, o *hypervisor* Xen é gerido por um cliente em execução com privilégios diferenciados, conhecido como Domain-0 ou Dom0. O Dom0 pode ser definido como um kernel Linux especialmente modificado que é iniciado pelo *hypervisor* Xen durante inicialização do sistema. É responsável pela gestão de todos os aspectos da outra máquina virtual sem privilégios, conhecidos como Domain-Us (DomU), que também estão em execução no *hypervisor*, conforme pode ser visualizado na imagem 3. No entanto, este tipo de virtualização necessita de processadores que apoiam especificamente as extensões de virtualização de *hardware* (Intel VT ou AMD-V).

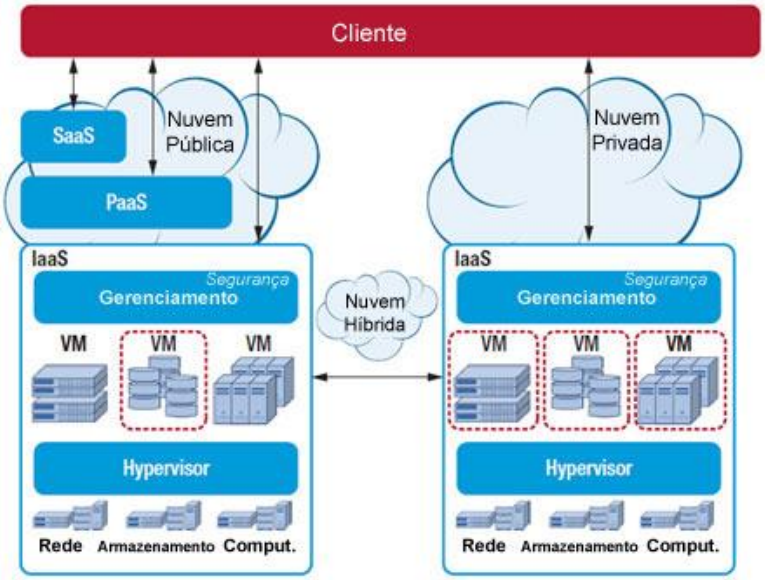
Em 2002 o XEN tornou-se *opensource*, licenciado sob a GPL2 (GNU - *General Public License*), para que pudesse haver uma grande contribuição da comunidade para seu desenvolvimento. Em 2004 foi lançada oficialmente a versão do XEN 1 .0 que logo foi substituída pela versão 2.0, quando Ian Pratt junto com outros líderes fundaram XenSource com o intuito de tornar o XEN *hypervisor* uma ferramenta de computação competitiva no mercado empresarial (XEN, 2012).

2.5 IaaS e os Gestores de Computação em Nuvem

Segundo (NIST, 2011) o modelo de serviços em Nuvem IaaS tem como principal objetivo tornar mais fácil e acessível o fornecimento de recursos computacionais, como processamento, rede, armazenamento e outros recursos de computação fundamentais em que o usuário pode instalar e executar *softwares* arbitrários, que podem incluir sistemas operacionais e *softwares*. Em geral, o usuário não administra ou controla a infraestrutura da nuvem, mas tem controle sobre os sistemas operacionais, armazenamento e aplicativos implantados, e, eventualmente, seleciona componentes de rede, tais como *firewalls*.

A Figura 3 apresenta a utilização dos três modelos de Computação em Nuvem apresentados acima (IaaS, SaaS e PaaS) e a infraestrutura de cada camada baseada nos modelos propostos.

Figura 3 – Os 3 modelos de Nuvem.



FONTE: SERRANO ET. AL., 2015

Segundo (SOUSA ET. AL., 2009), o termo IaaS refere-se a uma infraestrutura computacional baseada em técnicas de virtualização de recursos de computação. Esta infraestrutura pode escalar dinamicamente, aumentando ou diminuindo os recursos de acordo com as necessidades das aplicações. Do ponto de vista de economia e aproveitamento do legado, em vez de comprar novos servidores e equipamentos de rede para a ampliação de serviços, podem-se aproveitar os recursos disponíveis e adicionar novos servidores virtuais à infraestrutura existente de forma dinâmica.

Este tipo de serviço oferecido em Nuvem normalmente oferece aos consumidores um maior grau de controle comparado aos outros dois modelos de serviços, SaaS e PaaS, já que o consumidor fica responsável por saber quais as necessidades de recursos para as aplicações.

Segundo (BHARDWAJ et al., 2013), algumas características e componentes de IaaS incluem:

- Serviço de *Utility Computing*, cobrando por uso e demanda;
- Automatização das tarefas administrativas;
- Dimensionamento dinâmico;

- Virtualização de *desktops*;
- Serviços baseados em políticas;
- Conectividade com a Internet.

Ainda segundo (BHARDWAJ et al., 2013), o IaaS fornece um ambiente para o usuário executar sistemas virtualizados construídos na Nuvem, ou seja, o ambiente é construído através de um conjunto de máquinas virtuais que são orquestradas por um sistema que possibilita a administração centralizada. Usando esta técnica, este conjunto de *hosts* que hospedam máquinas virtuais são criados, carregados e gerenciados por este orquestrador. Tem também a função de auxiliar os fornecedores IaaS de realizar o seu gerenciamento, dimensionamento e comercialização de suas soluções, baseadas em demanda de recursos computacionais, tais como processador, memória, rede e armazenamento. O Quadro 1 apresenta as características básicas do modelo de Infraestrutura como Serviço, baseado em oferta, unidade de desenvolvimento e cobrança.

Quadro 1 - Características básicas do IaaS.

Oferta	Poder Computacional, Armazenamento, Infraestrutura de Rede
Unidade de Desenvolvimento	Imagem de Máquina Virtual
Cobrança	Cálculo por hora, transferência de dados (E/S) por GB, Armazenamento por GB, Requisições de Armazenamento.
Cliente	Proprietário de *software* que deseja ter sua aplicação hospedada na Internet para usuários finais.
Exemplos	Amazon, GoGrid e RackSpace

FONTE: MONTERO, 2012

Segundo BHARDWAJ et al. (2013), uma vez configurada, a máquina virtual pode ser criada e iniciada através de alguma forma que automaticamente encontra recurso disponível entre os hosts pertencentes ao ambiente para executar a máquina virtual. Após a máquina virtual ser iniciada, o fornecedor IaaS pode garantir a máquina virtual em execução ininterrupta e monitorar o consumo de recursos do sistema. Em um ambiente IaaS comercial, os computadores necessários para executar a aplicação e o armazenamento são de propriedade e suporte do fornecedor IaaS, assim como é da responsabilidade do cliente o monitoramento do *software* hospedado sobre a solução. Esta modalidade é uma opção muito flexível e normalmente é uma boa escolha para migrar aplicações de um ambiente tradicional para a nuvem.

Esta modalidade também é chamada de Sistema Operacional de Nuvem. Este S.O. de Nuvem administra as infraestruturas físicas e virtuais e controla o provisionamento de serviços, através da orquestração da implantação de recursos virtuais (MONTERO, 2012). Atualmente, algumas soluções *middleware* de código específico e aberto estão dominando o mercado de Computação em nuvem na modalidade IaaS. Soluções comerciais são oferecidos por algumas grandes empresas, como a VMware, Cisco, HP, IBM, Amazon, e uma grande quantidade de empresas de menor dimensão. Os principais concorrentes *opensource* são o Apache CloudStack, Eucalyptus, Open Nebula e Open Stack.

Existem dois papéis básicos em um ambiente de Computação em Nuvem IaaS: o cliente do serviço e o fornecedor do serviço. Um cliente de serviço para nuvem necessita de acesso seguro de qualquer local, e serviço de baixo custo sendo realizado por um sistema em que irá pagar pelo que consumir, além de ser de fácil utilização. Normalmente o maior obstáculo para a adoção de um serviço em Nuvem tem a ver com o desconforto dos clientes quanto a segurança do serviço e os dados subjacentes, disponibilidade e confiabilidade do serviço, gestão de serviços para garantir acordo de nível de serviço e a administração adequada para facilitar a estruturas de preços flexíveis (BHARDWAJ et al., 2013).

Segundo (BHARDWAJ et al., 2013), a visão dos clientes em IaaS:

- Permite que os usuários acessem aplicativos de qualquer local;
- Sistema modular, que é flexível, escalável, virtualizado e automatizado;
- Resiliente e sempre disponível;
- Permite colocar aplicativos e dados em plataforma provisionada e mantida pelo provedor.

Já a visão dos fornecedores em IaaS é descrita como:

- Oferece uma infraestrutura virtualizada (servidor, armazenamento e rede);
- É responsável pelo provisionamento de espaço, energia elétrica e climatização do ambiente;
- Implanta aplicações web facilitando o provisionamento de infraestrutura para o cliente sob demanda;
- Responsável por prover serviços de balanceamento de carga;
- Facilita o processo de clonagem de aplicativos em casos de infraestruturas adicionais;
- Mantém um acordo de níveis de serviços com o cliente;
- Garante a segurança dos dados e rede em um ambiente denso, compartilhado e em *pool*;

- Gerencia os contratos dos usuários e garante o pagamento através da demanda de consumo do cliente.

Segundo (LEI; YANG, 2014), a decisão de escolher uma plataforma de Nuvem mais apropriada entre muitas plataformas disponíveis deve vir a partir dos seguintes pontos: a criticidade do serviço a ser executado sobre a plataforma, características da plataforma, suporte a um sistema de arquivos, suporte a Sistemas Operacionais e *Hypervisors*. Tanto SaaS quanto PaaS podem ser construídos sobre uma plataforma IaaS, tanto que algumas empresas e instituições acadêmicas muitas vezes optam por construir plataformas em nuvem IaaS primeiro para que eles possam facilmente implementar seus aplicativos, personalizar as plataformas de aplicação ou melhorar as tecnologias de nuvem para melhor atender às suas necessidades de pesquisa e de negócios.

As funcionalidades de Infraestrutura como Serviço são projetadas principalmente seguindo recursos de Computação em Nuvem da Amazon EC2 - que é um padrão de Computação em Nuvem atualmente - para construir ambientes de Nuvem privadas e híbridas. Plataformas de Computação em Nuvem devem fornecer aos usuários a possibilidade de executar e controlar instâncias de máquinas virtuais (LEI; YANG, 2014).

Entre as principais plataformas de construção de Infraestrutura como Serviço na Nuvem, é possível citar o Eucalyptus, o OpenNubula, o OpenStack e o Apache CloudStack, que são apresentados a seguir.

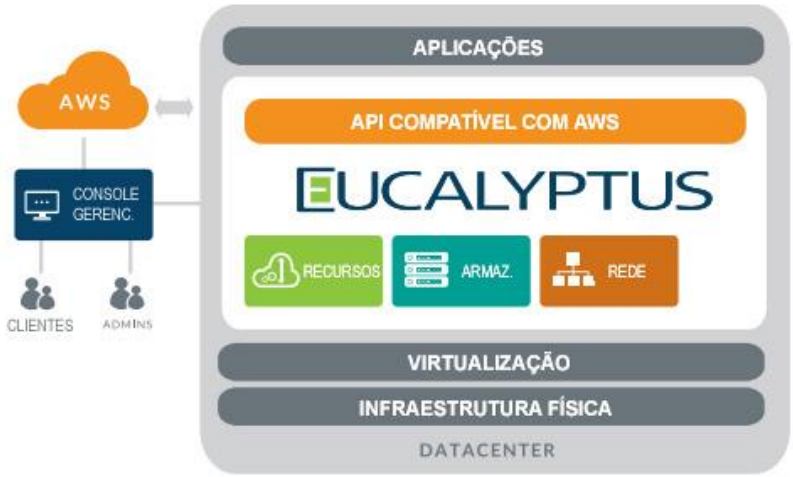
2.5.1 Eucalyptus

O Eucalyptus (*Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems*) é um sistema de infraestrutura para implementação de uma nuvem híbrida do tipo IaaS. Segundo a Eucalyptus (2014) o seu gestor de nuvem é um *software* de código aberto para a construção de nuvens compatíveis com a AWS (Amazon Web Service ™) através de nuvens privadas e híbridas. Ele aproveita a infraestrutura de TI existente para criar uma nuvem privada *self-service*, ou seja, possibilita ao cliente criar sua própria estrutura de nuvem. Uma Infraestrutura como Serviço (IaaS) pode ser ativada, abstraindo os recursos de computação, rede e armazenamento disponíveis. O Eucalyptus cria um *pool* de recursos escaláveis que pode dimensionar dinamicamente, dependendo da demanda da aplicação. Através de uma parceria com a Amazon Web Services ™ (AWS), o Eucalyptus mantém a compatibilidade da API, possibilitando aos usuários deslocar cargas de trabalho entre ambientes AWS e Eucalyptus.

Ainda segundo a Eucalyptus (2014), o desenvolvimento do seu produto tem suas raízes no Projeto de Rede Virtual de Desenvolvimento de *Software*, da Universidade Rice e outras instituições entre 2003 a 2008. Rich Wolski liderou um grupo da Universidade da Califórnia, Santa Barbara, e se tornou o diretor técnico da a empresa sediada em Goleta, na Califórnia, antes de voltar a ensinar na UCSB.

O Eucalyptus é um *software* baseado na arquitetura de *software* Linux que implementa nuvens escaláveis e eficientes, tanto privadas quando híbridas, dentro das organizações de TI. (EUCALYPTUS, 2010). A Figura 4 apresenta de forma simplificada a arquitetura do Eucalyptus.

Figura 4 - Arquitetura do gestor Eucalyptus.



Fonte: Eucalyptus, traduzido pelo autor (2014)

Segundo NIST (2013) os comandos do Eucalyptus podem gerenciar tanto instâncias Amazon quanto Eucalyptus. Os usuários também podem mover instancias entre uma nuvem privada Eucalyptus e o Amazon Elastic Compute Cloud para criar uma nuvem híbrida, já que a virtualização de *hardware* isola as aplicações de detalhes físicos do hardware utilizado.

Segundo Nurmi et al (2012), há quatro componentes de alto nível, cada um com sua própria interface de serviço Web, que compõem uma instalação Eucalyptus, que são descritas a seguir:

- **Nó Controlador:** controla a execução, inspeção e terminação de instâncias de máquinas virtuais no host em que ele é executado;

- **Controlador do cluster:** reúne informações sobre horários e execução de máquinas virtuais em controladores de nó específico, bem como administra a rede virtualizada;
- **Controlador de armazenamento:** é administrador de armazenamento que implementa a interface S3 da Amazon, fornecendo um mecanismo para armazenar e acessar imagens de máquinas e dados do usuário;
- **Controlador da Nuvem:** é o ponto de entrada para a nuvem para usuários e administradores. Ele consulta gerenciador do nó para obter informações sobre recursos, toma decisões de programação de alto nível, e os implementam, fazendo solicitações para controladores de cluster.

Ainda segundo Nurmi et al (2012), o Eucalyptus pode executar várias versões de imagens de máquinas virtuais Windows e Linux. Os usuários podem construir uma biblioteca de Eucalyptus Machine Images (EMIs) com metadados do aplicativo que são dissociados dos detalhes de infraestrutura, que possibilita executar em outras nuvens com Eucalyptus. Amazon Machine Images (AMI) também são compatíveis com nuvens Eucalyptus, assim como VMware, que pode ser convertido para funcionar com Eucalyptus e nuvens públicas AWS.

Segundo Buyya et al. (2011), o Eucalyptus se distingue de outras plataformas IaaS por fornecer uma nuvem de armazenamento emulando a API do Amazon para armazenar dados do usuário e imagens gerais da VM, oferecendo os seguintes recursos: administração Web; EC2 (SOAP, Query) e S3 (SOAP9, REST) CLI10 e interfaces Web, Xen, KVM e VMWare backends; Amazon EBS11 compatíveis com dispositivos de armazenamento virtuais, interface com o EC2 da Amazon, redes virtuais.

2.5.2 OpenNebula

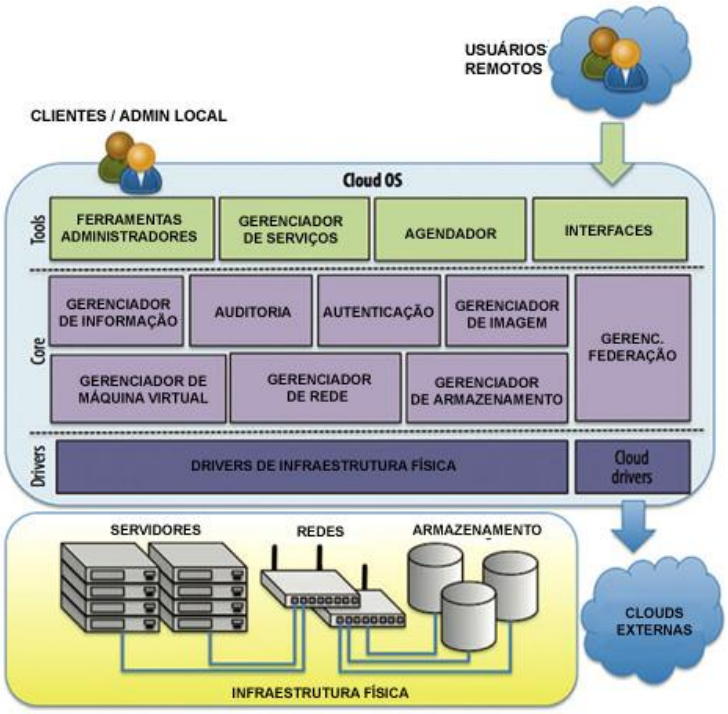
Segundo a OpenNubula (2014) sua solução de nuvem oferece uma solução simples, mas com recursos e flexibilidade para construir e gerenciar nuvens empresariais e *data centers* virtualizados. O OpenNebula é projetado para ser simples de instalar, atualizar e operar pelos administradores, e simples de usar pelos usuários finais. O OpenNebula possibilita a orquestração de sistema de armazenamento, rede, virtualização, monitoramento e tecnologia de segurança para implantar serviços multicamada, como por exemplo, *clusters* computacionais,

máquinas virtuais em infraestruturas distribuídas, combinando recursos de *datacenter* e Nuvem remotas, de acordo com as políticas de alocação.

O OpenNebula é um *kit* de ferramentas de computação em nuvem para gestão de infraestruturas de data centers distribuídos. O *toolkit* OpenNebula gerencia uma infraestrutura virtual de um data center para construir implementações privadas, públicas e híbridas de infraestrutura como serviço (IaaS). Este *toolkit* inclui recursos para integração, gerenciamento, escalabilidade, segurança e contabilidade. Ele também possui padronização, interoperabilidade e portabilidade, oferecendo aos usuários de nuvem e administradores com uma escolha de várias interfaces em nuvem (Amazon EC2, OGF Open Cloud Computing interface e vCloud) e *hypervisors* (Xen, KVM e VMware). Ainda segundo a OpenNubula, seu gestor de nuvem é um *software* livre e de código aberto, sob Licença Apache versão 2.

A arquitetura interna do OpenNebula engloba vários componentes especializados em diferentes aspectos para o gerenciamento da infraestrutura virtual, conforme pode ser visualizado na Figura 5.

Figura 5 - Arquitetura do gestor OpenNebula.



Fonte: OpenNubula, traduzido pelo autor (2014)

Segundo OpenNebula (2013), essa plataforma fornece 6 tipos de armazenamento de dados, conforme descritas a seguir:

- ***System*:** neste tipo de armazenamento de imagem, dependendo da tecnologia de armazenamento, podem ser utilizadas cópias completas da imagem original, qcow (formato de imagem utilizado pelo qemu) de disco deltas ou links para o arquivo original;
- ***File-System*:** a imagem é armazenada em formato de arquivo. São armazenadas em um diretório montado com acesso a um servidor SAN/NAS;
- **iSCSI/LVM:** a imagem é armazenada em forma de dispositivo de bloco;
- **Vmfs:** um *datastore* especializado em formato Vmfs para ser utilizado com o *Hypervisor* Vmware;
- **Ceph:** a imagem de disco é armazenada utilizando dispositivo em bloco Ceph;
- ***Files*:** este tipo de armazenamento é para arquivos como Kernel e Ramdisk.

Ainda segundo a OpenNubula (2016), existem 3 componentes principais:

- ***Host Management*:** o gerenciamento de host é obtido através do comando onehost CLI ou através do Sunstone GUI;
- ***Host Monitorization*:** A fim de manter o controle dos recursos disponíveis no conjunto de nós computacionais, o OpenNebula possui um host de monitoramento, chamado IM (Information Driver), que reúne todas as informações necessárias e envia para o Core;
- ***Cluster Management*:** Os hosts podem ser agrupados em *clusters*. Estes aglomerados são gerenciados com o comando CLI onecluster ou através do Sunstone GUI.

Segundo OpenNubula (2016) o seu orquestrador possui as seguintes características e recursos: baseado em Linux; CLI, XML-RPC, EC2 e OpenNebula Cloud API; Xen, KVM e VMware backend; interface para as nuvens públicas (Amazon EC2, ElasticHosts); redes virtuais; alocação dinâmica de recursos; reserva antecipada de capacidade.

2.5.3 OpenStack

Segundo OpenStack (2014), o seu gestor de nuvem é um *software* de código aberto, capaz de gerenciar os componentes de múltiplas infraestruturas virtualizadas, assim como o sistema operacional gerencia os componentes de nossos computadores. O OpenStack é chamado de Sistema Operacional da Nuvem, por cumprir o mesmo papel em maior escala. É considerado uma plataforma de *software*, por fornecer APIs que em conjunto são capazes de controlar todos os recursos disponíveis na oferta dessa infraestrutura: máquinas virtuais, rede, armazenadores, balanceadores de carga, até mesmo um painel de controle *web* está presente entre os *software*s do OpenStack, a maior parte escrita em linguagem Python.

Segundo (WEN ET. AL., 2012), o OpenStack é uma coleção de projetos de *software* de código aberto que as empresas ou provedores de nuvem podem utilizar para configurar e executar a sua própria computação em nuvem e infraestrutura de armazenamento. O projeto visa construir uma comunidade *opensource* com pesquisadores, desenvolvedores e empresas, eles compartilham um objetivo comum para criar uma nuvem que é simples de implementar, altamente escalável e cheio de recursos avançados.

Para (KHAN ET. AL., 2011), o OpenStack é uma das maiores comunidades de desenvolvimento de computação em nuvem *middleware* de código aberto do mundo, e ele é capaz de gerenciar os seguintes recursos de computação: *CPU*, memória, espaço em disco e banda de rede. A aplicação *middleware* utiliza um *hypervisor* rodando no *back-end* para permitir a criação de máquinas virtuais (*VMs*). Essas *VMs* emulam computadores físicos, e cada um tem um recurso de CPU, memória, disco e rede. Os recursos físicos reais para a criação das VMs são fornecidos por hosts virtuais. O OpenStack suporta virtualização com KVM, UML, XEN, e HyperV, usando o emulador QEMU. Na implementação, o libvirt, biblioteca C / C++ é usado para se comunicar com o *hypervisor* da camada de *middleware*.

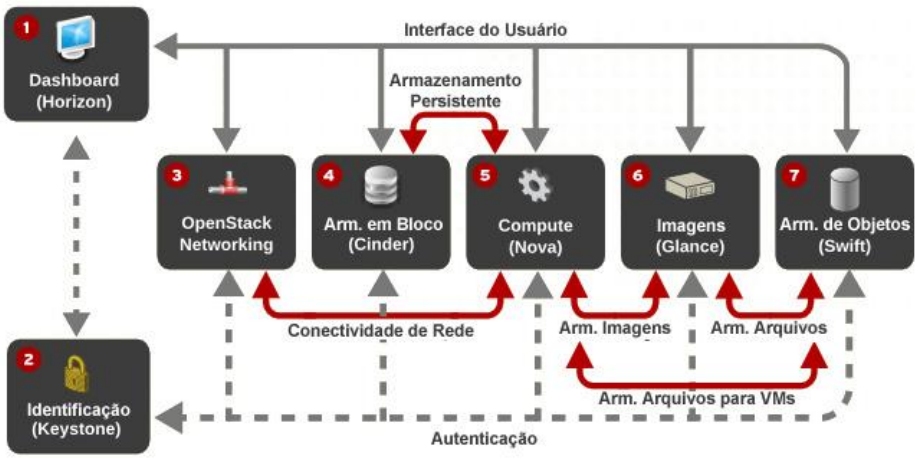
Ainda segundo (WEN ET. AL., 2012), o Rackspace - um dos maiores provedores de hospedagem dos EUA - e NASA são os dois maiores colaboradores do projeto. Além disso, é possível incluir três partes principais: a parte da computação que é chamado de Nova é previsto a partir de arquivos de computação da NASA, o armazenamento de objetos que é chamado de Swift que é do projeto Rackspace's "Cloud Files" e do serviço de imagem que é conhecido como Glance. O projeto tornou-se muito popular e amplamente aceito por adeptos da computação em nuvem e empresas como Citrix, Dell, SUSE, quando ele surgiu em julho de 2010. Nos dias atuais a comunidade OpenStack conta com mais de 150 empresas e 2.600 pessoas e esses números estão aumentando continuamente.

Sobre a arquitetura do OpenStack, (Hu e YU, 2014) o *software* é composto de vários subcomponentes desenvolvidos de forma independente, organizados em um subprojeto separado para cada serviço, oferecendo uma ampla gama de funcionalidades necessárias para construir uma nuvem IaaS.

Segundo Openstack (2014), sua versão Havana possui sete componentes, que podem ser visualizados na Figura 6. A seguir a descrição da funcionalidade de cada componente:

- **Horizon - Dashboard:** responsável pela interface *web* modular para os serviços OpenStack, onde é possível gerenciar seus recursos;
- **Keystone - Identificação:** provê autenticação para os serviços oferecidos pelo OpenStack;
- **Neutron - Networking:** módulo responsável pelos serviços de rede para o OpenStack, permite que redes virtuais sejam criadas e configuradas entre os recursos;
- **Cinder - Armazenamento de Bloco:** Fornece armazenamento em nível de bloco que pode ser montado como um volume pelas instâncias;
- **Nova - Compute:** módulo que disponibiliza máquinas virtuais conforme demanda dos clientes;
- **Glance - Imagens:** tem a função de gerenciar as imagens de VM usadas pelo Nova;
- **Swift - Armazenamento de Objetos:** oferece um serviço de armazenamento de objetos onde os clientes e administradores podem armazenar ou buscar seus arquivos.

Figura 6 - Arquitetura do OpenStack.



Fonte: RedHat, traduzido pelo autor (2014)

Ainda segundo (Hu e Yu, 2014), os administradores geralmente implantam o OpenStack usando um dos vários *hypervisors* suportados em um ambiente virtualizado. O KVM e o XenServer são as escolhas mais populares para tecnologia de *hypervisor*, além de ser recomendados para a maioria dos casos de uso.

2.6 Apache CloudStack

Segundo a Apache Foundation (2014), o Apache CloudStack foi originalmente desenvolvido pela empresa Cloud.com, anteriormente conhecido como VMOps. Em maio de 2010, a Cloud.com converteu a maior parte do Apache CloudStack como software livre (*OpenSource*) sob a Licença GNU - *General Public License*, versão 3 (GPLv3) e mantiveram-se cerca de 5% de propriedade. A Citrix, empresa de Tecnologia da Informação, comprou em 2011 a Cloud.com e em abril de 2012 a Citrix ofereceu o projeto CloudStack à Apache Software Foundation (ASF), onde foi aceito na incubadora Apache, que modificou a licença para a Licença Apache versão 2. Como parte dessa mudança a Citrix também deixou o seu envolvimento no OpenStack. Para a Apache Foundation (2014), o CloudStack é uma plataforma para criar e gerenciar infraestruturas escalonáveis e é escrito inteiramente em Java.

Muitas empresas utilizam o Cloudstack para a criação e administração de suas infraestruturas de nuvem, entre elas: Nokia, Orange, Apple, Disney, entre outras. (BARKAT ET. AL., 2014)

Segundo a Apache Foundation (2014), o Apache CloudStack é um *software* de código aberto projetado para implantar e gerenciar grandes redes de máquinas virtuais, como alta disponibilidade, Infraestrutura altamente escalável (IaaS), e uma plataforma de Computação em Nuvem. O Apache CloudStack é usado para oferecer serviços de nuvem pública, e por muitas empresas para fornecer uma infraestrutura local (privado) em nuvem, ou como parte de uma solução de nuvem híbrida. Ainda segundo a Apache Foundation o Apache CloudStack inclui toda a lista de recursos que a maioria das organizações precisam com uma nuvem IaaS: gerenciamento de máquinas virtuais, rede como serviço, gerenciamento de usuários e conta e uma API nativa e aberta.

Segundo (CHEN e XU, 2013), o Apache CloudStack usa um critério de design portátil e modular e é compatível com os recursos de infraestrutura existentes, com boa usabilidade. O design modular permite que o usuário personalize o funcionamento do sistema conforme

necessário. Além disso, o Apache CloudStack fornece uma API que é compatível com a AWS EC2 e S3 para as organizações que desejam implantar nuvens híbridas.

Ainda segundo (CHEN e XU, 2013), os recursos de *hardware* são abstraídos e separados em recursos de computação, de armazenamento e *cybersources* pelo *hypervisor*, que são organizadas como recursos virtuais. O Apache CloudStack integra e gerencia esses recursos virtuais para construir plataforma de nuvem, e fornece serviços de nuvem transparente para os usuários. Ele é responsável pelo mapeamento de recursos virtuais para recursos de *hardware* e possibilita o isolamento de segurança entre os usuários. Logicamente a implantação do Apache CloudStack inclui cinco partes, que podem ser definidas como: *host*, *cluster*, *pod* e zona de disponibilidade do servidor de gerenciamento e região.

O Apache CloudStack trabalha com uma variedade de *hypervisors*, inclusive sendo que uma única nuvem pode conter múltiplas implementações de *hypervisors* diferentes. A versão atual do Apache CloudStack suporta: BareMetal (via IPMI), Hyper-V, KVM, LXC, vSphere, XenServer e Xen Project.

Segundo a Apache Foundation (2015), o CloudStack possui muitas propriedades que motivam as empresas a usá-lo para gerenciar sua infraestrutura. As principais propriedades do Apache CloudStack são:

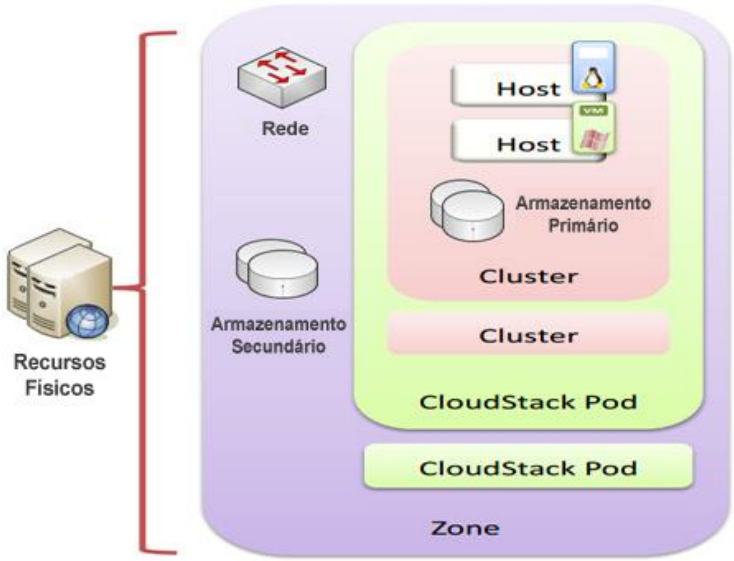
- ***Live Migration*:** A migração de máquinas virtuais em tempo de execução entre os hosts é permitida no Apache CloudStack através do seu painel de administração. Dependendo do *hypervisor*, as condições de migração de VMs entre diferentes hosts podem ter características diferentes. Por exemplo, Live Migration usando *hypervisor* KVM não suporta o uso de armazenamento em disco local, além dos hosts de origem e de destino terem que estar no mesmo Cluster. Já os *hypervisors* Xen e VMWare suportam armazenamento em disco local e permitem a migração entre diferentes grupos;
- **Balanceamento de carga:** o balanceador de carga é um componente do Apache CloudStack que permite distribuir o tráfego entre diferentes servidores de gerenciamento. Além de criar regras, usando algoritmos de balanceamento de carga, o CloudStack oferece a possibilidade de integração com balanceadores de carga externos, tais como Citrix NetScaler;
- **Tolerância a falhas:** no Apache CloudStack, a tolerância a falhas é conseguida em diferentes escalas. A fim de evitar falhas de servidor de gerenciamento, o servidor pode ser implementado numa configuração multi-nó. No caso de um nó de gerenciamento

falhar, os outros nós podem ser usados sem afetar o funcionamento da Nuvem. As falhas ao nível de base de dados são tratadas através de uma ou mais réplicas do banco de dados conectados ao servidor de gerenciamento. Para o *failover* dos *hosts*, o Apache CloudStack recupera as instâncias das máquinas virtuais utilizando as imagens do armazenamento secundário, e usando dados de aplicativo no armazenamento primário;

- **Disponibilidade:** o Apache CloudStack garante alta disponibilidade do sistema usando múltiplos nós do servidor de gerenciamento que podem ser implantados com balanceadores de carga;
- **Segurança:** além do isolamento utilizando diferentes contas, *VPNs* e *firewalls*, o CloudStack oferece o isolamento de tráfego usando a estratégia de grupos de segurança que são conjuntos de VMs que filtram o tráfego com base em regras de configuração pré-estabelecidas. O Apache CloudStack também fornece um grupo de segurança padrão com regras pré-definidas, porém eles podem ser modificados se necessário;
- **Compatibilidade:** o Apache CloudStack é construído com base em uma arquitetura plugável, onde uma nuvem pode suportar diferentes implementações de *hypervisors*, incluindo: Hyper-V, KVM, LXC, Vmware vSphere, XenServer, Xen Project e também o provisionamento *bare metal*. Além disso, Apache CloudStack é compatível com a API do Amazon e permite a integração de suas plataformas;
- **Escalabilidade:** o Apache CloudStack tem a capacidade de gerenciar milhares de servidores de diferentes *hypervisors* distribuídos em *datacenters* geograficamente distantes, graças a capacidade de gerenciamento do servidor de gerenciamento.

Para a Apache Foundation (2015), a arquitetura do Apache CloudStack inclui cinco módulos principais como mostrado na Figura 7, e descritos em seguida.

Figura 7 - Arquitetura do Apache CloudStack.



Fonte: Citrix, traduzido pelo autor (2014)

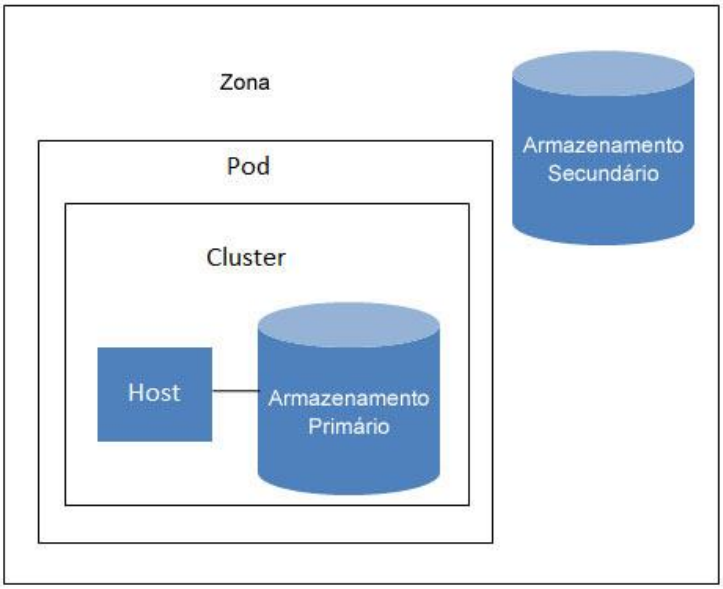
- **Região:** A região é a maior unidade organizacional disponível em uma implantação Apache CloudStack. A região é composta de várias zonas de disponibilidade, em que cada zona é aproximadamente equivalente a um *datacenter*. Cada região é controlada pelo seu próprio conjunto de servidores de gerenciamento, que devem executar em uma das zonas. As zonas em uma região geralmente estão localizadas geograficamente próximas. Regiões são uma técnica útil para fornecer tolerância a falhas e recuperação de desastres;
- **Zonas:** Uma zona é a segunda maior unidade organizacional dentro de uma implantação Apache CloudStack. Uma zona geralmente corresponde a um único *datacenter*, embora seja permitido ter várias zonas em um *datacenter*. O benefício de organizar infraestruturas em zonas é fornecer isolamento físico e redundância. Por exemplo, cada zona pode ter sua própria fonte de alimentação e rede, e as zonas podem ser amplamente separadas geograficamente. Zonas são visíveis para o usuário final e podem ser privadas ou públicas. Zonas públicas são visíveis para todos os usuários na nuvem, enquanto zonas privadas são visíveis apenas para os usuários de um determinado domínio;
- **Pods:** Um Pod geralmente representa um único *rack* que contém um ou mais *hosts* e um ou mais servidores de armazenamento primários, e é a terceira maior unidade organizacional dentro de uma implantação CloudStack. Os Pods estão contidos dentro de uma Zona e cada Zona pode conter uma ou mais Pods;

- ***Cluster*:** Proporciona uma forma de organizar os *hosts*, dentro de um mesmo *pool* computacional. Se muitos nós computacionais estão no mesmo *cluster* e eles têm o mesmo *hardware* e *hypervisor*, estando na mesma sub-rede, podem acessar a armazenamento principal compartilhada;
- ***Host*:** Um *host* representa uma máquina computacional física que contém um armazenamento local. Os hosts físicos são virtualizados por *hypervisors*. O CloudStack suporta diversos *hypervisors* para gerenciamento de máquinas virtuais, como Xen, KVM, vSphere, o Hyper-V, VMWare, entre outros. No CloudStack, *hosts* que pertencem ao mesmo *cluster* devem ter o mesmo *hypervisor*, com a possibilidade de ter *hypervisors* diferentes em diferentes *clusters*;
- **Servidor de Gerenciamento:** É usado para gerenciar todos os recursos em infraestrutura em nuvem através de APIs ou UI. Um servidor de gerenciamento pode suportar em torno de 10 mil *hosts* e pode ser implantado em um servidor físico ou em uma máquina virtual. Um banco de dados MySQL é necessário para servidores de gerenciamento, que podem estar instalados na mesma máquina do servidor de gerenciamento ou em máquina separada.

Para (KAI ET. AL., 2014), o Apache CloudStack oferece serviços altamente disponível e altamente escalável para uma plataforma *IaaS*, projetado para implantar e gerenciar grandes redes de máquinas virtuais. Além disso, ele pode oferecer serviços de nuvem pública e serviços de nuvem privada ou nuvens híbridas, além de oferecer uma interface de usuário amigável.

Segundo APACHE FOUNDATION (2015), o Apache CloudStack define dois tipos de armazenamento: primário e secundário. No armazenamento primário são utilizados dispositivos conectados via iSCSI ou NFS, além do armazenamento anexado diretamente que também pode ser utilizado neste tipo de armazenamento. Já o armazenamento secundário é sempre acessado utilizando NFS. Não há armazenamento efêmero no Apache CloudStack e todos os volumes em todos os nós são persistentes. A Figura 8 apresenta a camada em que os armazenamentos primários e secundários são localizados dentro da arquitetura Apache CloudStack.

Figura 8 - Armazenamento no Apache CloudStack.



Fonte: Apache Foundation, traduzido pelo autor (2015)

Para (BARKAT ET. AL., 2014), além do armazenamento local do próprio *host*, o Apache CloudStack gerencia dois tipos principais de armazenamento: armazenamento primário e de armazenamento secundário:

- **Armazenamento primário**: é um armazenamento associado a um *Cluster* ou uma Zona. No mesmo *cluster*, é possível implantar vários *storages* primários. Este tipo de armazenamento é basicamente usado para executar as máquinas virtuais e os dados do aplicativo. Uma vez que este tipo de armazenamento interage diretamente com os aplicativos de virtualização, ele pode ser exigido em termos de operações de entrada/saída e esta é a razão pela qual ele é colocado normalmente fisicamente próximo dos *hosts* - na mesma rede;
- **Armazenamento Secundário:** é usado para armazenar imagens ISO (é usado quando o usuário quer criar uma *VM*), *templates* (é uma imagem pronta de um sistema operacional que o usuário pode escolher quando desejar criar nova instância), *snapshots* (usado como um ponto de restauração para o serviço de recuperação de dados) entre outros. Ele suporta tanto armazenamento NFS (*Network File System*) quanto *Object Storage.*

O armazenamento primário está associado a um *cluster*, e armazena discos virtuais para todas as *VMs* em execução nos *hosts* de um *cluster*. Para os *hypervisors* KVM e VMware, é

possível provisionar armazenamento primário baseado em uma Zona. É possível também adicionar vários servidores de armazenamento primário para um *Cluster* ou Zona, mas ao menos uma é essencial para criação do ambiente. Estes são normalmente localizados próximos aos *hosts* para evitar degradar o desempenho.

Já os dados do armazenamento secundário estão disponíveis para todos os *hosts* orquestrados por um gerenciador Apache CloudStack, que podem ser definidos por Zona ou por Região. Além disso, o Apache CloudStack fornece *plugins* que permitem tanto OpenStack Object Storage (Swift) quanto Amazon Simple Storage Service (S3) para o armazenamento de objetos.

Segundo a Apache Foundation (2015), existem algumas práticas que são consideradas essenciais para o bom funcionamento de um ambiente Apache CloudStack relacionadas ao armazenamento. A velocidade do Armazenamento Primário irá afetar o desempenho da Máquina Virtual hóspede. É essencial a utilização de discos com tecnologia mais rápidas, como SSD e redes de armazenamento que suportem altas velocidades, como *fibre channel*. Além disso, existem 2 modelos de utilização que podem aumentar o desempenho associado ao Armazenamento Primário:

- **Estático:** Este é o modo tradicional de utilização do Armazenamento Primário no Apache CloudStack. Neste modelo, uma quantidade pré-distribuída de armazenamento, como por exemplo, um volume de uma SAN, é apresentada ao Apache CloudStack, que, em seguida, permite que muitos dos seus volumes possam ser criados nesta área de armazenamento;
- **Dinâmico:** Neste modelo, um sistema de armazenamento, ao invés de uma quantidade pré-distribuído de armazenagem, é apresentado ao Apache CloudStack, que trabalhando em conjunto com um plug-in de armazenamento, cria dinamicamente volumes no sistema de armazenamento. Cada volume no sistema de armazenamento mapeia para um único volume Apache CloudStack. Isto pode ser utilizado para recursos como armazenamento que necessitam realizar controle de Qualidade de Serviço.

O quadro 2 mostra as opções de armazenamento e parâmetros para diferentes *hypervisors*, utilizando o Apache CloudStack. Com suporte a diversos formatos e protocolos de armazenamento, Apache CloudStack e *hypervisor* Xen oferecem possibilidades para armazenamentos de imagens, *snapshopts*, *templates* entre outros.

Quadro 2 - Armazenamento no Apache CloudStack com diferentes *Hypervisors*.

Armazenamento\ Hypervisor	VMware Sphere	XEN	KVM	Hyper-V
Formato para discos, Templates e Snapshots	VMDK	VHD	QCOW2	Snapshots VHD não são suportados.
Suporte iSCSI	VMFS	Clustered LVM	Sim, via ponto de montagem compartilhado	Não
Suporte Fiber Channel	VMFS	Sim	Sim, via ponto de montagem compartilhado	Não
Suporte NFS	Sim	Sim	Sim	Não
Suporte a Armazenamento Local	Sim	Sim	Sim	Sim
SMB/CIFS	Não	Não	Não	Sim

Fonte: Apache Foundation, 2015

Entre os formatos possíveis, o iSCSI (*Internet Small Computer System Interface*), é um protocolo de transporte que possibilita comandos SCSI entre um computador anfitrião (*Initiator*) e um dispositivo de destino (*Target*). É um protocolo tipicamente usado no contexto de uma SAN (*Storage Area Network*) mas que, ao contrário do *Fibre channel*, não necessita de uma infraestrutura especializada e dedicada, podendo funcionar sobre uma rede IP convencional (HUFFERD, 2002). Já o NFS (Network File System) é um sistema de arquivos distribuídos, desenvolvido inicialmente pela Sun Microsystems, Inc., para compartilhar arquivos e diretórios entre computadores conectados em rede, formando assim um diretório virtual. (STOKELY, 2004)

2.7 Trabalhos Relacionados

Neste tópico são apresentados alguns aspectos referentes a trabalhos encontrados na literatura atual que apresentam abordagens para avaliação de desempenho em Computação em Nuvem, modalidade *IaaS*, com ênfase no Apache CloudStack. No capítulo 4 é realizado um comparativo dos principais tópicos levantados aqui com o trabalho desenvolvido.

Reddy e Rajamani (2014) utilizaram o framework SIGAR (*System Information Gatherer and Reporter)*, uma ferramenta que acessa informações do Sistema Operacional, para realização dos testes de que permitiram mensurar o consumo de CPU, memória, E/S e redes em um ambiente Apache CloudStack, com 4 diferentes *hypervisors*: XenServer, Vmware ESXi, KVM e Hyper-V. Os autores concluem a partir dos resultados dos testes realizados que os *hypervisors* VMware ESXi Server, XenServer e Hyper-V são melhores preparados para atender as demandas de um *datacenter* corporativo do que o *hypervisor* KVM. Enfatizam que o KVM precisa de melhorias significativas para se tornar um *hypervisor* aceito corporativamente, em especial devido ao desempenho apresentado.

Kai, Weiqin, Liping e Chao (2013) apresentaram uma análise de desempenho baseado na coleta de métricas via SNMP (*Simple Network Managment Protocol*) dos recursos físicos e virtuais, incluindo os principais componentes do CloudStack (máquinas virtuais, armazenamento secundário, armazenamento primário e servidores de gerenciamento), e tornam esses dados acessíveis e legíveis, o que é bastante amigável para os utilizadores Apache CloudStack. Os autores concluem que o sistema de monitoramento SCM realiza a coleta de diversos valores e métricas, tanto de recursos físicos quanto virtuais e armazena esses dados para posterior análise, além de desenvolverem um protótipo de sistema para validar o sistema proposto.

Paradowski, Liu e Yua (2014) demonstraram a importância de compreender a necessidade de benchmark em plataformas em nuvem, a fim de obter uma visão geral sobre o desempenho das plataformas, com foco nas duas mais utilizadas de código aberto, OpenStack e CloudStack. O autor enfatizou que os testes de desempenho foram focados estritamente nas plataformas, testando em *hypervisor* e recursos disponíveis iguais. Os autores identificaram nos testes realizados que o desempenho do OpenStack é superior ao do Apache CloudStack, evidenciado ao longo dos testes de desempenho, onde o OpenStack executou tarefas em menor tempo do que o Apache CloudStack.

2.8 Considerações Finais

Este capítulo apresentou conceitos gerais sobre a Computação em Nuvem, com ênfase na modalidade *IaaS*, que é o foco deste estudo. Foram apresentados alguns dos principais orquestradores *opensource* IaaS atualmente: Eucalyptus, OpenNebula e Openstack, além do

Apache CloudStack, além de abordar as principais características, conforme pode ser visualizado no Quadro 3. Todos são equivalentes nos recursos oferecidos, como balanceamento de carga e armazenamento permitido, além de serem compatíveis com a maioria dos *hypervisors* disponíveis no mercado atualmente.

Quadro 3 – Características dos orquestradores.

Ferramenta	Interface	Balanceamento de Carga	Virtualização	Segurança	Rede	Armazenamento
Eucalyptus	SSH e WEB	Elastic Load Balancer	Xen, KVM e Vmware ESXi	Autenticação, Cug e Active Directory	Bridge e VLAN	AoE, iSCSI e NFS
OpenNebula	SSH e WEB	Sim	Xen, KVM e Vmware ESXi	Autenticação e Cug	Bridge , VLAN e Open Vswitch	iSCSI. NFS e LVM
OpenStack	SSH e WEB	Quantum Network Load Balancing	Xen, KVM e Hyper-V	Keystone, LDAP e métodos externos	VLAN e OpenVSwitch	AoE, iSCSI e NFS
CloudStack	SSH e WEB	NetScaler	Xen, KVM e Vmware ESXi	Autenticação e Cug	VLAN	iSCSI e NFS

Fonte: Hu e Yu, 2014

Posteriormente, foi abordado o orquestrador de Nuvem IaaS Apache CloudStack, apresentando suas principais características, como suporte a múltiplos *hypervisors* e escalabilidade. Foi apresentada também sua arquitetura, composta basicamente por Região, Zone, Pod, Cluster e Host, além do servidor de gerenciamento. Por fim o modo que o Apache CloudStack realiza o armazenamento de suas imagens, VMs, *snapshots* entre outros.

O orquestrador de Nuvem Apache CloudStack foi escolhido para realização deste trabalho, além de possui um alto nível de relevância de projetos para Apache Software Foundation, por apresentar características relacionadas ao armazenamento de dados, como tecnologias de disco e protocolos de armazenamento, que possibilitam realizar diversos experimentos, além de contar com características como o suporte a múltiplos *hypervisors*, que tornam atrativos para pesquisas de desempenho. Já o *hypervisor* XenServer foi escolhido por ter total compatibilidade ao orquestrador, já que ambos foram desenvolvidos inicialmente pela Citrix Systems, Inc.

CAPÍTULO 3 – StackAct

3.1 Considerações Iniciais

O termo desempenho em Computação em Nuvem pode ser compreendido de diversas maneiras, como por exemplo, o desempenho no provisionamento de novas instâncias, passando pelo remanejamento de recursos computacionais, escalabilidade, tanto no nível de recursos físicos quanto virtuais. Além disso, diversos fatores também podem impactar no desempenho da solução adotada – desde a configuração da própria solução, equipamentos utilizados, recursos disponíveis, modelo de solução adotada, entre outros. Segundo Joyent (2013), os aplicativos baseados em Nuvem também devem manter um bom desempenho, já que a produtividade dos funcionários se baseia no desempenho e confiabilidade do aplicativo para completar o trabalho com precisão e rapidez.

Falhas de aplicação devido à falta de desempenho podem trazer prejuízo para as empresas e dificultar a expansão dos negócios. Se as aplicações não podem ter um bom desempenho durante um aumento do tráfego, por exemplo, há perdas de clientes e receitas. Por exemplo, uma aplicação que serve adequadamente 100 usuários por hora pode ter uma demanda temporária repentina por 500 usuários, com o mesmo comportamento. O planejamento de capacidade da Nuvem com base em testes de estresse pode ajudar as empresas a tomar decisões mais precisas, com melhor provisionamento de suas plataformas de Nuvem.

O OpenStack possui uma ferramenta de *benchmark*, chamada de Rally[16], que permite avaliar como um ambiente de Nuvem orquestrado pelo OpenStack responde ao rápido provisionamento de recursos. Para tornar isto possível, o Rally automatiza e unifica a implantação multi-nó do OpenStack, através de um tipo de *benchmark*. O Rally faz isso de uma maneira modularizada, possibilitando ao administrador da Nuvem verificar se OpenStack vai funcionar adequadamente em uma instalação de um ambiente sob carga alta. Ele pode ser usado

[16] https://wiki.openstack.org/wiki/Rally

como uma ferramenta de base para um sistema de OpenStack que permite melhorar continuamente a sua SLA, desempenho e estabilidade.

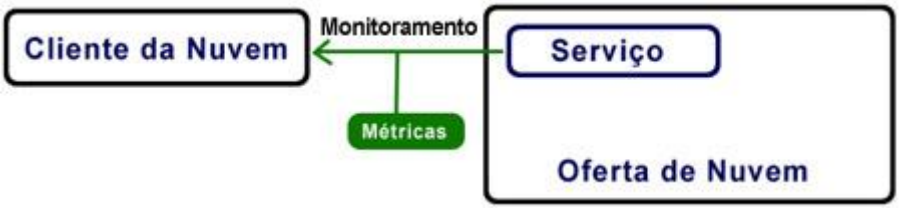
Já o Apache CloudStack, até o momento do desenvolvimento deste estudo, não possui nenhuma ferramenta oficial para avaliar a escalabilidade, desempenho e comportamento de um ambiente de Nuvem diante de uma necessidade de rápido provisionamento

Segundo Kai et. al. (2013), em ambientes computacionais em Nuvem, as métricas de monitoramento são importantes para a sistemas de tarifação, programação de novas tarefas e de outros componentes da nuvem. Devido às características do ambiente de nuvem, as métricas de monitoramento são alteradas de forma dinâmica, e o volume de dados pode se tornar muito grande, sendo necessário um sistema de armazenamento escalável e de alto desempenho.

No ambiente Apache CloudStack, os *hosts* podem ter diferentes formas de significado, podendo ser físicos ou virtuais, como instâncias utilizadas pelos clientes da Nuvem ou máquinas virtuais do sistema, de modo que as métricas adotadas pela solução de monitoramento podem ser utilizadas em diferentes formas de visualização. Ainda segundo Kai. Et. Al. (2013), estes coletores devem ser facilmente configurados para coletar métricas diferentes, de acordo com a necessidade do cliente da Nuvem.

Segundo o documento de métricas para Computação em Nuvem do NIST (2015), o uso de métricas para sistemas de Computação em Nuvem na interface de serviço pode ser dividido em três áreas gerais: seleção de serviço, contrato de serviço, e verificação de serviço. Métricas são essenciais, não apenas para entender cada uma dessas áreas, mas para ligar estas três partes distintas do processo de aquisição nuvem. Ainda segundo o NIST, conforme apresentado na Figura 9, as métricas para medição de serviço tem início que uma vez que o cliente compra um serviço de nuvem, é necessário para garantir o os objetivos a nível do serviço contratado estão sendo atendidas, e caso isto não aconteça, uma solução deve ser iniciada.

Figura 9 – Objetivos de Monitoramento em Serviços de Nuvem.



Fonte: NIST, traduzido pelo autor (2015)

Este trabalho apresenta um mecanismo que realiza a coleta de dados referentes ao consumo de recursos computacionais e permite a avaliação do desempenho no provisionamento de instâncias em um ambiente *IaaS* com o Apache CloudStack, denominado StackAct. A estrutura adotada permite a coleta de informações de diversos recursos a nível de host, ou seja, sobre os recursos dos servidores físicos, não fazendo parte do escopo do trabalho a coleta de dados a nível de instância (*VM*).

3.2 Fatores importantes de desempenho em Computação em Nuvem

Com o crescente aumento na utilização de ambientes em Nuvem, a elasticidade tornou-se uma característica essencial para o sucesso da solução. Segundo (NIST, 2014), a elasticidade é a habilidade do rápido provisionamento e liberação de recursos, com capacidade de recursos virtuais praticamente infinita e quantidade adquirível sem restrição a qualquer momento. Herbst et. al. (2013) destaca que a elasticidade consiste em quanto um sistema e capaz de se adaptar a variações na carga de trabalho pelo provisionamento e liberação de recursos de maneira automática, de modo que em cada ponto no tempo os recursos disponíveis combinem com a demanda da carga de trabalho o mais próximo possível.

Um dos problemas presentes na elasticidade é que ela pode levar tempo para ocorrer em um ambiente em Nuvem. Uma máquina virtual pode ser provisionada a qualquer momento pelo usuário da Nuvem, porém, pode demorar até vários minutos para que a *VM* esteja pronta para ser utilizada. O tempo de inicialização *VM* é dependente de fatores, como tamanho da imagem, tipo de *VM*, a localização do *datacenter*, número de *VMs*, entre outros (MAO e HUMPHREY, 2012). Os provedores de Nuvem possuem desempenho diferente de inicialização de uma instância ou *VM* na Nuvem. Isto implica que qualquer mecanismo de controle concebido para aplicações elásticas deve considerar em seu processo de decisão o tempo necessário para as ações de elasticidade para entrar em vigor, tais como provisionamento de outra *VM* para um componente de aplicação específica.

No entanto, segundo MAO e HUMPHREY (2012), essa elasticidade é importante apenas para os usuários da Nuvem quando as *VMs* podem ser provisionados em tempo hábil e estares prontas para usar dentro da necessidade do usuário. O longo tempo de inicialização não esperada de uma *VM* pode resultar em recursos super provisionados, o que, inevitavelmente,

pode prejudicar o desempenho do uso do ambiente. Impõe-se uma melhor compreensão do tempo de inicialização *VM* para ajudar os usuários na nuvem para planejar com antecedência e tomar decisões de provisionamento de recursos em tempo. Um melhor entendimento sobre como a *VM* realiza suas atividades (criação, inicialização, entre outros) ajudam os usuários da Nuvem a planejar com antecedência e tomar melhores decisões de provisionamento de recursos.

Segundo XU e LIU (2013) os atuais sistemas *VMMs* ou hypervisors, como por exemplo, o Citrix XenServer, VMware vSphere e Microsoft Hyper-V, já fornecem bons mecanismos de isolamento de desempenho sobre o compartilhamento de recursos de núcleos de CPU, memória e capacidades de E/S de disco entre as *VMs* armazenadas. A fim de ter um bom entendimento sobre a sobrecarga das *VMs* no desempenho de um *datacenter* em Nuvem, a exigência principal é medir a sobrecarga da *VM* ou das *VMs* no desempenho da solução como um todo. Alguns estudos atuais têm dedicado grandes esforços para modelar a sobrecarga de *VMs* no desempenho em dois aspectos: a interferência de *VMs* armazenadas em um servidor físico e a sobrecarga de desempenho??? *VM* causada pela migração em tempo real de máquinas virtuais.

Já com relação ao sistema de armazenamento de instâncias em um ambiente em Nuvem, XU et. al. (2013) reforçam que um serviço de armazenamento compartilhado em Nuvem implanta um sistema de armazenamento inteiro para apoiar as solicitações de E/S de várias *VMs* hospedadas em vários servidores físicos. Para isolar o consumo de recurso de E/S entre várias *VMs*, pode-se gerenciar os dois tipos de fila de requisições de E/S em cada servidor físico e sistema de armazenamento compartilhado. Além disso, outros controles utilizam o sistema de armazenamento, como o controle de snapshots, logs, entre outros. O impacto das requisições para o armazenamento de instâncias e outras funções relacionadas às *VMs* no desempenho do ambiente de Nuvem deve ser um fator crítico na decisão para uma solução em Nuvem.

No caso do StackAct, vários fatores relacionados ao desempenho de cada camada podem ser coletados para análise. Para elaboração deste trabalho foram selecionados para análise os mesmos recursos em todas as camadas, descritas no tópico 4.2, possibilitando um comparativo entre elas. Porém é possível selecionar os itens mais relevantes, de acordo com as características de cada camada, conforme segue:

- **Camada do Orquestrador:** Esta é uma camada cuja principal característica é a solicitação de eventos para as demais camadas. Desta forma, é possível determinar que o consumo de *CPU* é determinante para um bom desempenho. O StackAct permite o monitoramento de diversas características relacionadas ao consumo de *CPU*, consumo

por processos dos usuários, porcentagem aguardando E/S em disco e *CPU* total disponível.

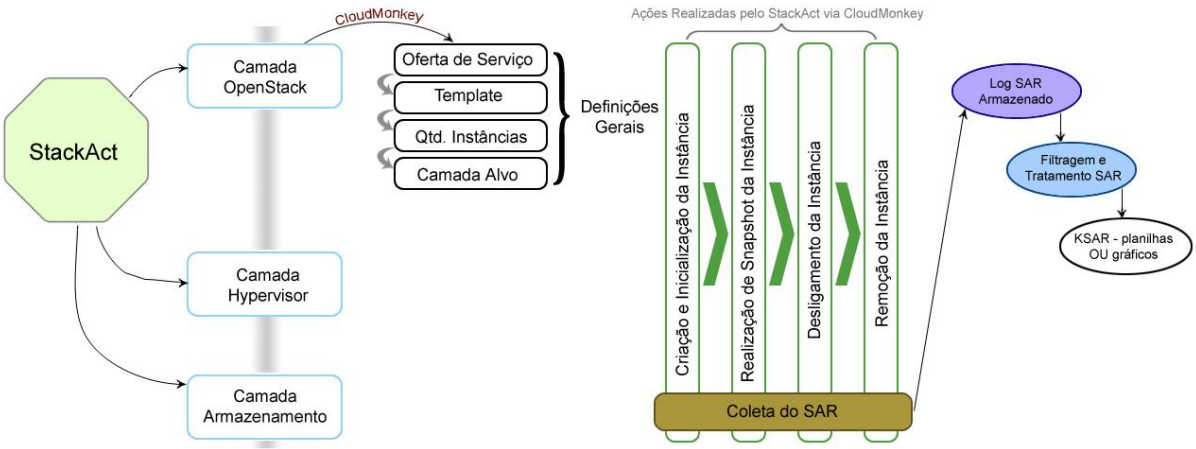
- **Camada do *hypervisor*:** Nesta camada, o monitoramento da memória é determinante para o desempenho da solução, já que cada máquina virtual utiliza a quantidade provisionada pelo *host* que ela é executada. É possível coletar estatísticas diversas relacionadas a memória, como memória livre, memória em uso apenas por processos de sistema e *swap*, que é a utilização da memória secundária.
- **Camada do armazenamento:** As informações relacionadas ao consumo de disco são as mais relevantes nesta camada. É possível obter diversas estatísticas de fluxo de E/S, como transações por segundo, total de transações de leitura ou gravação num período e total de *bytes* de leitura ou gravação, que apresentam relação direta com o desempenho da aplicação.

3.3 Estrutura do StackAct

O StackAct permite realizar o monitoramento e coleta de dados referentes ao consumo de diversos recursos computacionais, como *CPU*, Memória, Disco, Processos e rede. Foi desenvolvido para permitir o monitoramento nos componentes de uma Nuvem computacional em até 3 camadas, sugeridas neste trabalho por: camada do orquestrador, utilizando o Apache CloudStack, camada do *hypervisor*, utilizando o XenServer, e finalmente a camada de armazenamento de dados, utilizando o Openfiler.

A arquitetura do mecanismo de monitoramento do StackAct é apresentada na Figura 10, e compreende 3 etapas principais: definições gerais sobre a coleta, eventos realizados pelo StackAct sobre a(s) instância(s) e o armazenamento e tratamento dos *logs* gerados.

Figura 10 – Estrutura do StackAct.



Fonte: Elaborado pelo autor (2016)

Nas definições gerais sobre a coleta, são realizadas as configurações para execução do StackAct, compreendida em 4 etapas que devem ser determinadas pelo usuário do sistema, conforme segue:

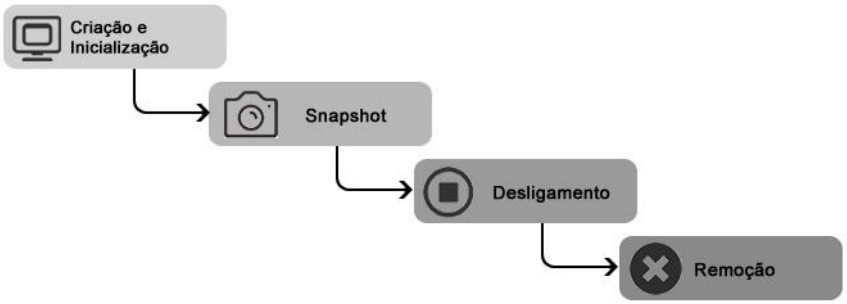
- **Definição da Oferta de Serviço:** As ofertas de serviço no Apache CloudStack são a definição da capacidade de processamento, a quantidade de memória, a rede (vLAN), a velocidade e o tamanho dos discos de uma instância. Para execução do StackAct, é necessário a existência prévia de ao menos uma oferta de serviço, já que nas ações com a(s) instância(s) serão utilizadas as características de *VM* baseadas na oferta selecionada. A configuração no StackAct é realizada através da identificação da oferta de serviço escolhida para a execução.
- **Definição do *Template*:** Um *template* no Apache CloudStack é uma configuração reutilizável para *VMs*, com um Sistema Operacional já instalado e configurado. Quando um usuário cria uma nova *instância*, pode escolher a partir de uma lista de *templates* disponíveis. Para a execução do StackAct, é necessário a existência prévia de ao menos um *template*, que será utilizado para as ações com a instância. A configuração no StackAct é realizada através da identificação do *template* escolhida para a execução.
- **Definição da Quantidade de Instâncias:** É necessário configurar no StackAct a quantidade de instâncias simultâneas que o mecanismo irá realizar a sequência de ações, possibilitando avaliar o comportamento do ambiente durante um provisionamento de múltiplas instâncias, como ocorre em um cenário real de Nuvem *IaaS*. O valor mínimo é 1, e o valor máximo vai depender da quantidade de recursos disponíveis na Nuvem.

- **Definição da Camada Alvo:** É possível realizar a coleta e monitoramento dos dados em uma, duas ou todas as camadas contempladas neste trabalho. Para isto é necessário indicar o endereço IP (*Internet Protocol*) correspondente ao dispositivo que está executando o serviço que se deseja monitorar, além de configuração para acesso remoto SSH.

Na segunda grande etapa presente na arquitetura do StackAct, é realizada a rotina definida na concepção do mecanismo, que é executada em 4 ações, conforme pode ser visualizado na Figura 11:

- **Criação e inicialização de uma nova instância:** este processo, como os outros processos presentes no StackAct, apesar de ser iniciado na camada do orquestrador, é realizado pela camada do *hypervisor* – com a criação e inicialização de uma nova máquina virtual. Ou seja, baseado na oferta de serviço e *template* escolhidos na camada do orquestrador, é enviado uma solicitação para o *hypervisor* criar e inicializar uma nova VM. Além disso, como os arquivos referentes as *VMs* ficam armazenados fisicamente na camada de armazenamento de dados, existe um impacto sobre esta camada;
- **Criação de um *snapshot*:** é criada uma cópia da instância, preservando o estado e os dados de uma máquina virtual em um determinado momento. Um *snapshot* é comumente utilizado para restauração de um estado anterior de uma máquina virtual, e está presente na maioria dos *hypervisors*. Para realização de um *snapshot*, a camada do orquestrador requisita a camada do *hypervisor* a geração desta cópia, que é fisicamente realizada na camada de armazenamento dos arquivos;
- **Desligamento a instância:** Ao final da geração do *snapshot*, ocorre uma solicitação de desligamento da instância, que é realizada através de um envio de sinal para o Sistema Operacional fechar os aplicativos e fazer um *shutdown*;
- **Remoção da instância do disco:** Com a máquina virtual já desligada, é iniciado um processo para remoção dos arquivos pertencentes do disco.

Figura 11 – Sequência de eventos realizados para definição da coleta de dados.



Fonte: Elaborado pelo autor (2016)

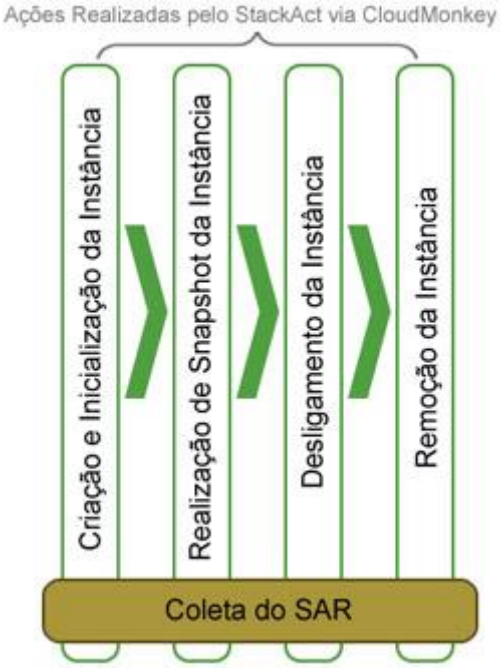
Para realização da sequência de eventos e execução da rotina, foi necessário a utilização do sistema CloudMonkey, que é uma ferramenta distribuída pela comunidade CloudStack, de interface de linha de comando (CLI), escrito em Python[17], e que pode ser utilizado tanto como um *shell* interativo como uma ferramenta de linha de comando que simplifica a configuração e gerenciamento do CloudStack. O CloudMonkey possibilita, por exemplo, a criação e remoção de novas instâncias na Nuvem, configuração de Zona, Pod e Cluster, gerenciamento de perfis do servidor, entre outros. Praticamente tudo o que é possível realizar por interface gráfica (GUI) do CloudStack, é possível realizar utilizando o CloudMonkey, por meio de linha de comando. O CloudMonkey foi utilizado no desenvolvimento do método proposto para realizar operações relativas a criação, snapshot e exclusão das máquinas virtuais, executando comandos que podem ser visualizados no apêndice B.

De forma paralela a realização da rotina programada, ou seja, com início simultâneo ao início da criação da *VM* e encerramento ao final da remoção de todos os arquivos que compõe a *VM*, realiza-se o monitoramento dos recursos e a coleta de dados, baseado nas definições iniciais, conforme pode ser visualizado na Figura 12. Este monitoramento é feito pela ferramenta SAR[18] (*System Activity Reporter*) para coleta do consumo de recursos dos hosts do ambiente proposto. O SAR é um conjunto de ferramentas, originalmente desenvolvido para ambiente SOLARIS e posteriormente portado para Linux, que realiza o monitoramento de diversos recursos do sistema, como CPU, Memória e *Swap*, E/S de leitura e gravação, *System Load Average*, Atividade de Rede, Atividade de Criação de Processo, entre outros. O SAR é uma das ferramentas de coleta de desempenho para ambientes UNIX/Linux que fazem parte do

[17] Python é uma linguagem de programação de alto nível, interpretada, de script, imperativa, orientada a objetos, funcional, de tipagem dinâmica e forte (http://www.python.org.br).
[18] https://docs.oracle.com/cd/E26505_01/html/816-5165/sar-1.html

pacote SYSSTAT, que inclui também as ferramentas sadf, mpstat, iostat, tapestat, pidstat entre outras.

Figura 12 – Coleta de dados utilizando o SAR durante a sequência de eventos.



Fonte: Elaborado pelo autor (2016)

No caso do StackAct, o SAR é configurado para gerar informações de monitoramento de recursos a cada segundo de execução, ou seja, caso o processo todo (criação/inicialização, *snapshot*, desligamento e remoção da instância) durar 60 segundos, serão gerados 60 vezes o conjunto de informações de monitoramento. Essas informações são armazenadas em arquivo, em forma de *log* bruto, sem formatação.

Após a coleta dos dados, foi utilizado também o software KSAR[19], que é uma ferramenta de elaboração de gráficos e planilhas a partir dos resultados obtidos através da ferramenta SAR. O KSAR possibilita exportar os resultados de comandos SAR em gráficos PDF, JPEG ou exportar para planilhas CSV, este último utilizado na elaboração nos testes. O KSAR converte o arquivo gerado pelo SAR em planilha, para posterior análise.

Como a plataforma nativa para implementação do Apache CloudStack é o S.O. Linux, os passos para elaboração deste mecanismo foram contemplados em forma de *script* para

[19] https://sourceforge.net/projects/ksar/

automatização do processo de *benchmark*, desenvolvido em Shell Script[20]. Os requisitos básicos para funcionamento do mecanismo são:

- **Camada do Orquestrador:** Além da necessidade do Apache CloudStack, é imprescindível que tenha o CloudMonkey instalado e habilitado. Além deste, SAR e o KSAR devem estar instalados;
- **Camada do *Hypervisor*:** O mecanismo é independente do *hypervisor* utilizado, ou seja, pode ser utilizado com qualquer um que seja homologado para rodar com o Apache CloudStack. Porém, como o mecanismo utiliza SAR e KSAR, estes devem estar instalados nesta camada;
- **Camada de Armazenamento:** Da mesma forma que a camada do hypervisor, o mecanismo é independente do sistema utilizado, desde que este tenha instalado o SAR e o KSAR.

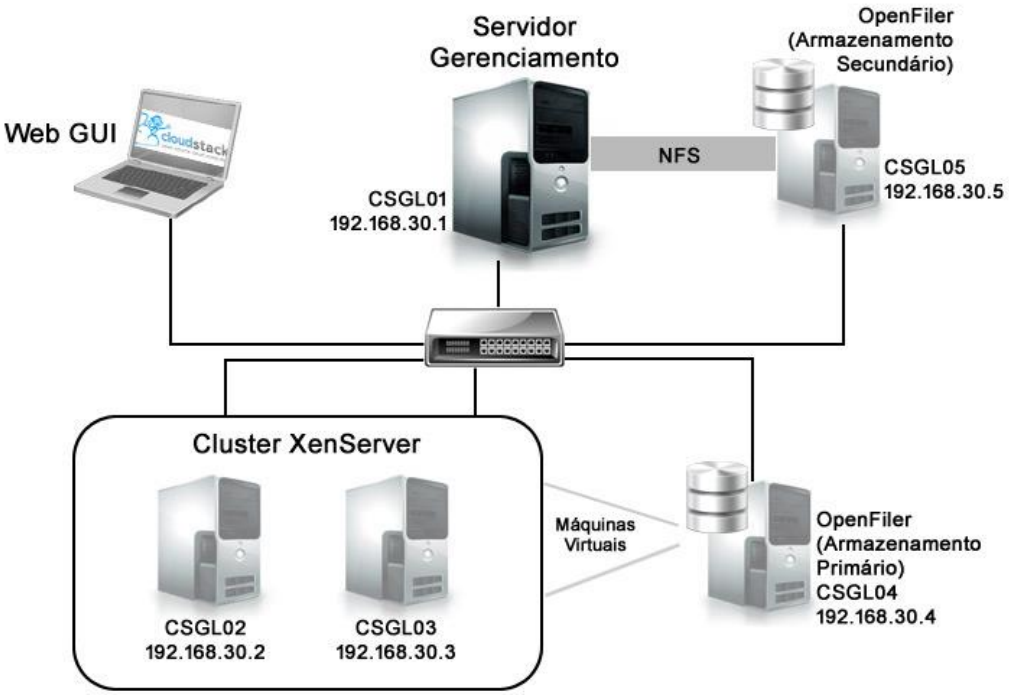
[20] Shell Script é uma linguagem de script usada em vários sistemas operacionais, com diferentes dialetos, dependendo do interpretador de comandos utilizado (http://www.shellscript.com.br).

CAPÍTULO 4 – Experimentos Realizados

4.1 Considerações Iniciais

Para validação do StackAct, foi criado um ambiente composto por 5 servidores físicos, conforme apresentado na Figura 13: o primeiro (CSGL01) é um servidor com processador Intel Core2 Quad com 2.66Ghz e 4GB de memória RAM, onde foi instalado S.O. Linux CentOS 6.6 e o orquestrador Apache CloudStack 4.4.2. O segundo e o terceiro servidores (CSGL02 e CSGL03) possuem as mesmas características do servidor CSGL01, com a exceção que foram adicionados mais 4GB de memória RAM, totalizando 8GB em cada um dos servidores. Foram instalados o hypervisor XenServer 6.2.0 e posteriormente criado um *cluster* entre os dois *hosts*.

Figura 13 – Nuvem privada criada para execução dos testes.



Fonte: Elaborado pelo autor (2016)

Fazendo a função de armazenamento primário e secundário para o orquestrador CloudStack, foram utilizados mais dois servidores, com as mesmas características dos demais, no qual foi instalado o Sistema Operacional OpenFiler 2.99 para armazenamento dos arquivos das máquinas virtuais, imagens de S.O., *snapshot*, entre outros. O Openfiler[21] (OpenFiler, 2016) é um sistema que é executado sobre um S.O. Linux que fornece armazenamento anexado à rede com base em arquivo e rede de armazenamento baseada em bloco. Para compartilhamento das pastas na rede, foi criado um compartilhamento em nível de arquivos utilizando NFS (*Network File System*), com acesso irrestrito e público as pastas compartilhadas na rede.

O StackAct foi executado no servidor CSGL01 (CloudStack), e como o objetivo foi coletar dados de consumo de recursos das 3 camadas definidas no escopo inicial, foi modificado apenas a sessão da coleta do consumo de recursos no servidor remoto - quando o objetivo foi mensurar o desempenho dos servidores CSGL02 (XenServer) e CSGL04 (OpenFiler) – para posterior avaliação de desempenho.

4.2 Definições e Execução dos Experimentos

Para execução dos testes e possibilidade de mensurar o desempenho utilizando diferentes cargas, foram criados previamente no Apache CloudStack, 3 ofertas de serviços com tamanhos diferentes de recursos alocados, conforme ilustrado no Quadro 4. É importante ressaltar que os tamanhos das instâncias foram limitados pelo ambiente disponível para os testes.

Quadro 4 - Características das ofertas de serviço utilizados nos testes.

	Cores	CPU (Mhz)	Memory (MB)
Pequena	1	500	512
Média	1	1000	1024
Grande	2	2000	2048

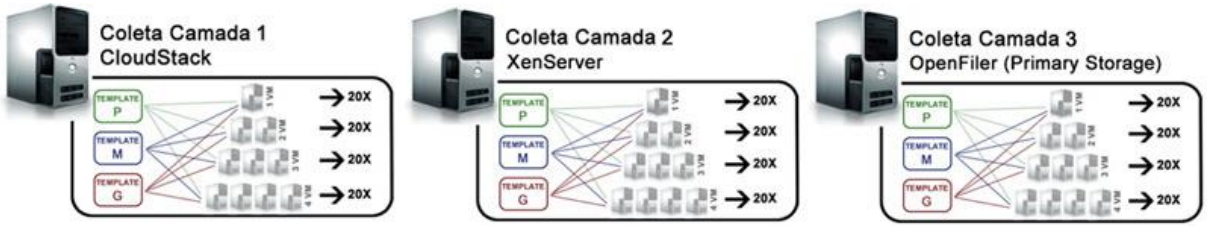
Fonte: Elaborado pelo autor (2016)

Foram realizados testes com até 4 rotinas completas (criação e inicialização, *snapshot*, *shutdown* e remoção da instância) realizadas de forma simultânea, e cada um dos perfis de teste

[21] https://www.openfiler.com/

criado foram executados 20 vezes. Esta rotina de testes foi realizada fazendo a coleta do consumo de recurso em cada um dos servidores proposto inicialmente: CSGL01 (CloudStack), CSGL02 (XenServer01) e CSGL04 (Openfiler, Armazenamento Primário), conforme apresentado na Figura 14.

Figura 14 – Rotina para coleta dos dados.



Fonte: Elaborado pelo autor (2016)

Após a execução de todos os testes, o StackAct gerou 20 arquivos CSV para cada uma das rotinas e *templates*, cada um destes arquivos com 179 colunas relativas a informações de consumo de recursos durante a execução da rotina, como por exemplo: consumo das interfaces de rede, *load average* (média de valores médios dos processos esperando execução ou em execução), diversas informações relativas a consumo de *CPU* e memória. Estas planilhas foram filtradas priorizando: (1) Total de *CPU* disponível, (2) Total de *CPU* consumida por processos de usuário, (3) Total de CPU aguardando E/S em disco, (4) E/S de Leitura de dados, (5) E/S de Gravação de dados e (6) Consumo de Memória.

Conforme JARGAS (2008), a descrição dos itens filtrados na ferramenta SAR está descrita no Quadro 5:

Quadro 5– Descrição dos recursos na ferramenta SAR.

Item	Descrição
Total de ***CPU*** disponível	Porcentagem que a *CPU* ou *CPUs* estavam ociosas.
Total de ***CPU*** consumida por processos de usuário	Porcentagem de utilização da *CPU* que ocorreu durante a execução no nível do usuário (aplicação). Este campo não inclui o tempo gasto executando processadores virtuais.
Porcentagem de ***CPU*** aguardando E/S em disco	Porcentagem de tempo em que a *CPU* fica aguardando uma requisição de E/S ser executada. Segundo IBM (2006), este ficará bloqueado e outros entrarão na fila para serem processados.
E/S de Leitura de dados	Número total de solicitações de leitura por segundo enviadas para os dispositivos físicos.
E/S de Gravação de dados	Número total de solicitações de gravação por segundo enviadas para os dispositivos físicos.
Consumo de Memória	Total de memória consumida excluindo o total de memória em *buffer*.

Fonte: JARGAS (2008)

Nas avaliações prévias que foram realizadas nos arquivos CSV ainda no estado original, ou seja, sem os agrupamentos e cálculos das médias aritméticas para posterior análise, foi constatado que não houve uma alteração significativa nos valores gerados pela rotina com 1 ou 4 instâncias sendo executadas simultaneamente - apenas o tempo de execução era maior quando haviam mais instâncias. Após análise, foi possível concluir que para o ambiente utilizado, algumas rotinas ficavam em fila devido as limitações de configuração de recursos físicos do ambiente.

Assim, para ter um valor único de comparação, a partir dos resultados obtidos nos arquivos CSV e posterior filtragem dos dados desejados para análise, foi tirado a média aritmética do total de consumo de cada recurso, já que a coleta foi realizada a cada segundo durantes os testes, estes resultavam em tempos diferentes, com número diferente de número de

execuções do SAR. Por exemplo, no teste realizado na menor instância (PEQUENA) com apenas 1 instância isolada, eram capturados em média 44 resultados, por executar em 44 segundos. Já no teste realizado com a maior instância (GRANDE) com 4 instâncias simultâneas, eram capturados em média 288 resultados, já que o tempo total para execução deste teste era em média de 288 segundos. Ou seja, para análise e comparação dos resultados, as métricas filtradas para cada processo (criação e inicialização de instância, *snapshot*, *shutdown* e exclusão da máquina) foi resumido a média aritmética de todo o processo.

4.3 Análise Geral dos Resultados

A média aritmética dos tempos obtidos nos testes realizados para obtenção dos resultados para cada configuração definida – oferta de serviço e quantidade de testes realizados simultaneamente – são apresentadas no Quadro 6. A quantidade total de testes por servidor ao final dos estudos foi de 240, com o número total de 600 instâncias criadas, inicializadas e eliminadas – 200 em cada perfil. Porém o total de arquivos CSV criados foi 720, já que cada teste resultou um arquivo CSV em cada uma das camadas analisadas (Orquestrador CloudStack, Hypervisor XenServer e sistema de armazenamento Openfiler).

Quadro 6 – Média de tempo para execução dos testes.

	1 Instância	2 Instâncias	3 Instâncias	4 Instâncias
Pequena	44 seg	132 seg	209 seg	263 seg
Media	48 seg	140 seg	218 seg	273 seg
Grande	102 seg	154 seg	228 seg	288 seg

Fonte: Elaborado pelo autor (2016)

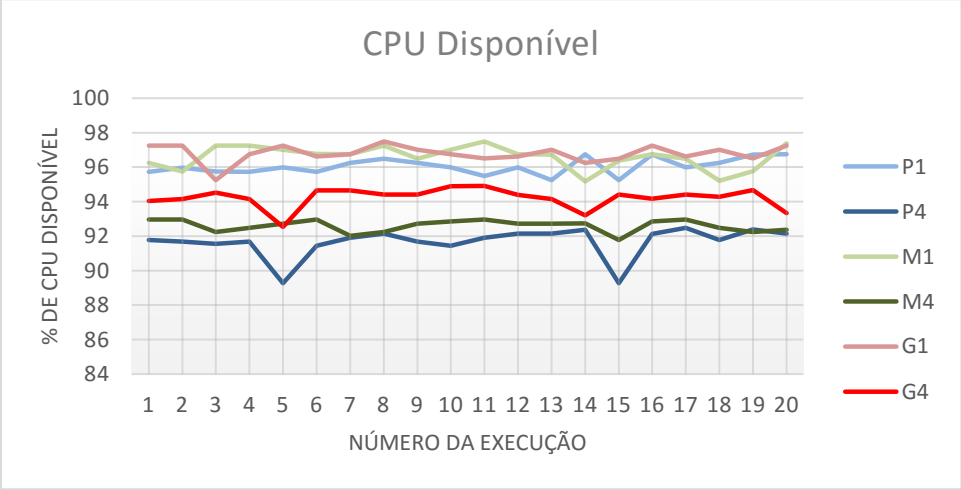
É importante evidenciar estes tempos, já que foi possível perceber que durante os processos com várias instâncias sendo criadas simultaneamente, o *hypervisor* coloca as ações de criação de instância, criação do snapshot, desligamento e exclusão da instância em fila. Isso fica evidente ao apresentar os resultados – figuras de 15 a 32 – do comparativo entre cada recurso avaliado com os diferentes perfis. Devido a esta constatação, optou-se por não exibir os valores dos testes com 2 e 3 instâncias sendo criadas simultaneamente – serão apresentados apenas com 1 e 4 instâncias.

Para apresentação dos resultados obtidos nos testes, foi definido que para cada um dos itens a ser analisado (*CPU* disponível, *CPU* de usuário, *CPU* aguardando E/S, E/S de gravação e leitura e consumo de memória) será exibido no eixo X o resultado da média aritmética de cada uma das execuções (de 1 a 20), e no eixo Y a métrica relativa ao item analisado.

4.3.1 Resultados Individuais da camada CloudStack

Nas Figuras 15 a 20 são apresentados os resultados referentes ao servidor CSGL01 (CloudStack). A Figura 15 apresenta a porcentagem de CPU disponível para este servidor, nos perfis definidos acima, e fica evidente que não há grande diferença como esperado, por exemplo, uma oferta de serviço pequena com apenas 1 instância (P1) da oferta de serviço grande rodando com 4 instâncias simultâneas (G4). Inclusive, como pode ser visualizado na Figura 15, a configuração que consumiu mais *CPU* – e consequentemente apresentou menores valores de CPU disponível – foi a configuração P4, mesmo considerando uma diferença mínima para as outras. Como este servidor tem função de apenas controlar os elementos da Nuvem, e apenas gera requisições para as outras camadas, pode ser considerado normal não haver grande consumo de *CPU*.

Figura 15 – CPU Disponível no servidor CSGL01, com diferentes configurações.

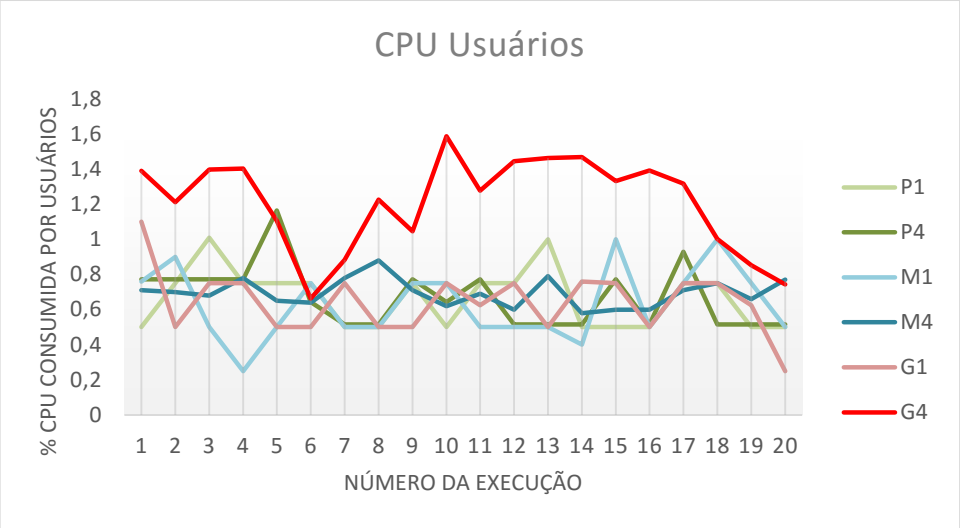


Fonte: Elaborado pelo autor (2016)

A Figura 16 apresenta o total de *CPU* consumido por processos do usuário no mesmo servidor. Processos de usuário, neste caso, são os processos recorrentes de aplicações, com exceção dos processos do próprio sistema – ou seja, as aplicações que estão sendo executadas.

Aqui também é possível constatar que não há grandes diferenças entre as configurações, além do baixo consumo de *CPU* pelas aplicações, já compreendido na análise anterior. Na configuração G4 foi possível detectar uma diferenciação com relação as outras configurações, que é justificada por outros processos de sistema sendo executados no servidor, já que a diferença no geral fica menor do que 1% de consumo.

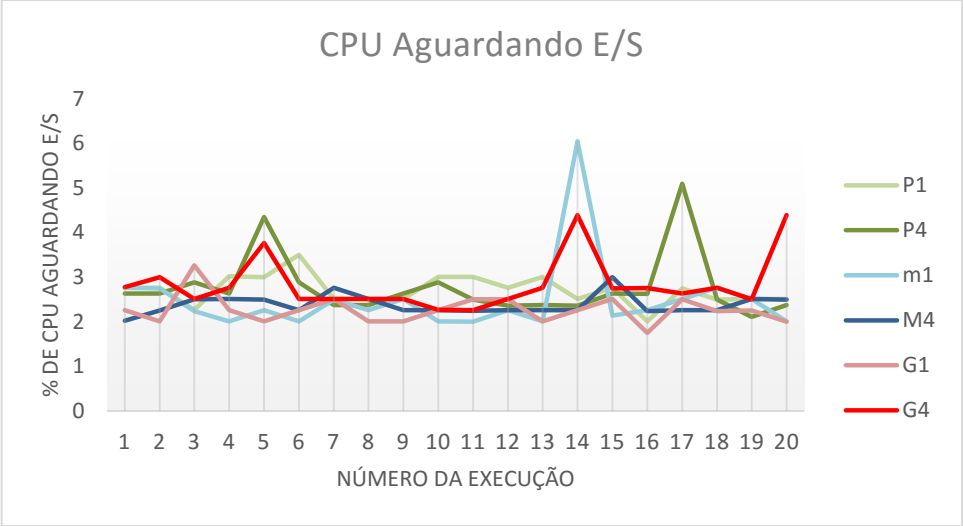
Figura 16 – *CPU* consumida por processos do usuário no servidor CSGL01, com diferentes configurações.



Fonte: Elaborado pelo autor (2016)

A Figura 17 apresenta o total de *CPU* consumido por eventos em espera de leitura e/ou gravação em disco, ou seja, o percentual *CPU* gastos aguardando processos do sistema de armazenamento. E como já era esperado para este servidor, o valor é baixo – já que o serviço mais relevante no consumo de recursos de armazenamento é o banco de dados, fazendo registros referentes ao controle do ambiente de Nuvem. Além disso, não há grandes variações entre as diferentes configurações.

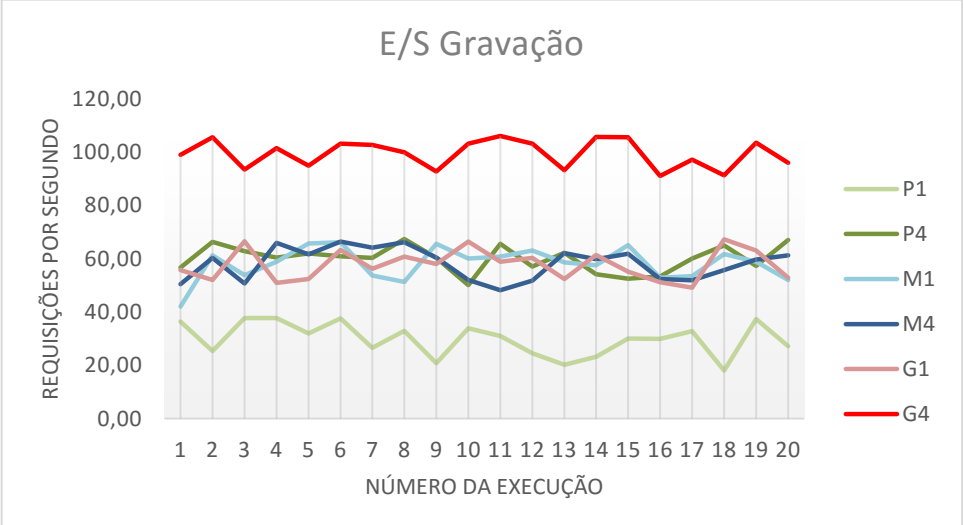
Figura 17 – Porcentagem de *CPU* aguardando disco no servidor CSGL01, com diferentes configurações.



Fonte: Elaborado pelo autor (2016)

Na figura 18 são apresentados os resultados referentes ao número total de pedidos de gravação emitido para o disco físico, por segundo. Com a função de orquestrador, a principal ação do servidor CSGL01 quanto a gravação em disco é registrar todas as ações do ambiente de Nuvem no banco de dados, além do registro de alguns *logs*. No gráfico, é possível verificar que realmente não há um grande número de requisições de gravação, como será possível visualizar em outros ambientes.

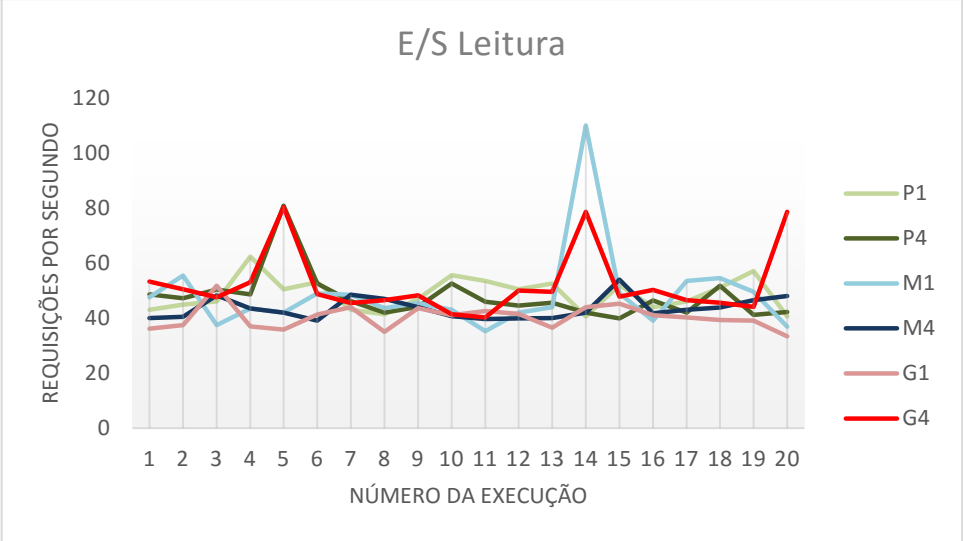
Figura 18 – Número de requisições de gravação por segundo no servidor CSGL01, com diferentes configurações.



Fonte: Elaborado pelo autor (2016)

Na figura 19 são apresentados os resultados referentes ao número total de pedidos de leitura emitido para o disco físico, por segundo. Foi possível verificar um número baixo de requisições de leitura, já justificado na análise do gráfico anterior - ou seja, não existem ações neste servidor que cause alto consumo no sistema de armazenamento.

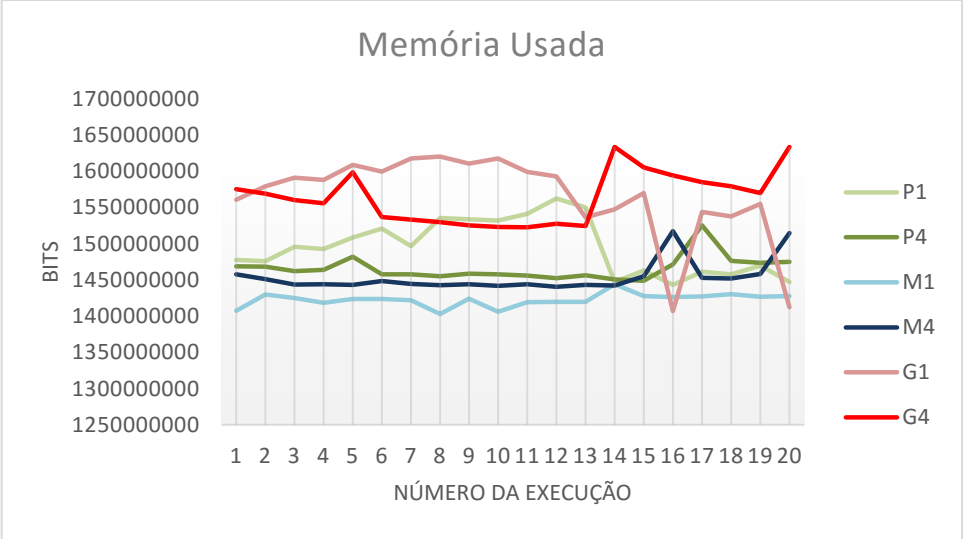
Figura 19 – Número de requisições de leitura por segundo no servidor CSGL01, com diferentes configurações.



Fonte: Elaborado pelo autor (2016)

A Figura 20 apresenta a quantidade de memória utilizada, não considerando a memória utilizada pelo próprio *Kernel*. É possível verificar o crescimento do consumo de memória conforme as configurações, mesmo que de forma discreta, associado as atividades que os serviços realizados neste servidor requerem – ainda que em quantidade pequena.

Figura 20 – Quantidade de memória utilizada no servidor CSGL01, com diferentes configurações.



Fonte: Elaborado pelo autor (2016)

O quadro 7 apresenta o desvio padrão referente a média dos resultados obtidos nas 20 execuções, em cada um dos itens avaliados e configuração da camada CloudStack (CSGL01).

Quadro 7 – Desvio padrão em cada item, referente aos resultados obtidos na camada CloudStack.

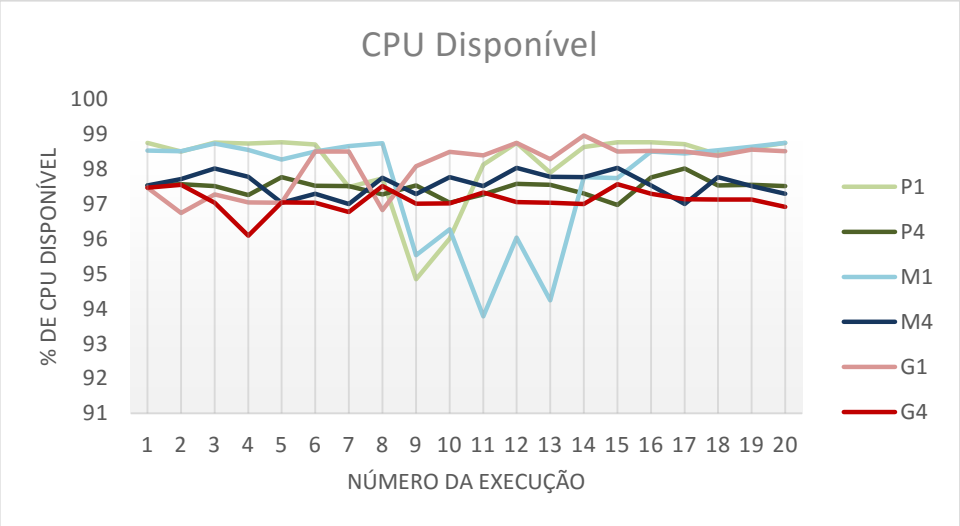
	P1	P4	M1	M4	G1	G4
CPU Disponível	0,47	0,87	0,67	0,35	0,49	0,58
CPU Usuários	0,16	0,17	0,20	0,07	0,17	0,26
CPU Aguardando E/S	0,32	0,71	0,87	0,22	0,32	0,61
E/S Gravação	6,28	5,00	6,15	5,91	5,73	5,26
E/S Leitura	6,01	8,68	15,58	3,95	4,30	12,06
Memória Usada	37,84	16,81	92,51	21,90	59,78	36,30

Fonte: Elaborado pelo autor (2016)

4.3.2 Resultados Individuais da camada XenServer

Nas Figuras 21 a 26 são apresentados os resultados referentes ao servidor CSGL02 (XenServer). A camada do *hypervisor* é responsável por processar as requisições enviadas pela camada do orquestrador, ou seja, é nesta camada onde, de fato, o consumo de processamento e memória é mais intenso. Os testes foram concentrados em apenas um dos *hosts*, procurando obter dados de consumo de recursos próximos a situação real - evitando que fossem distribuídos entre os nós computacionais do *cluster*. A Figura 21 apresenta a porcentagem de *CPU* disponível, e conforme já analisado na camada superior, não há grande diferença entre as configurações e número de instâncias – no caso desta camada entendidas como máquina virtual – são executadas.

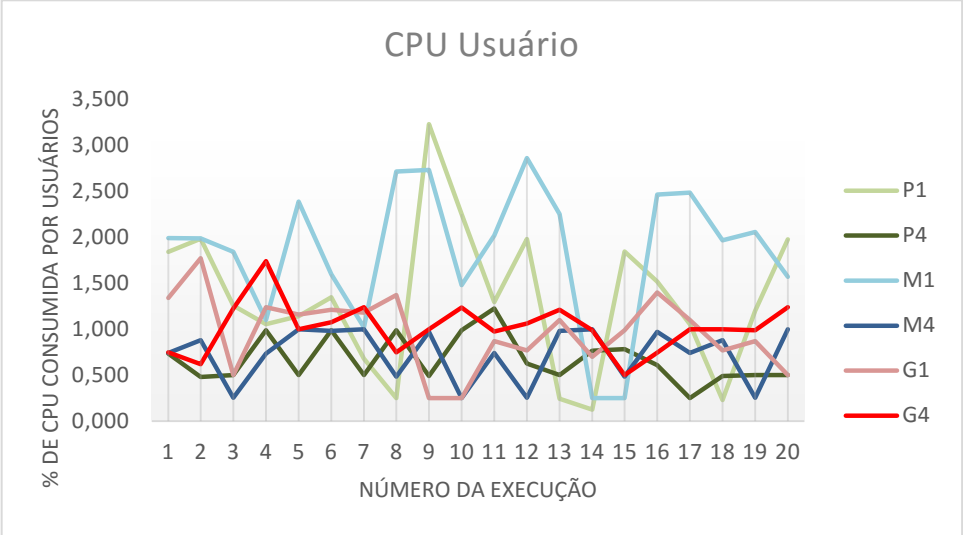
Figura 21 – CPU Disponível no servidor CSGL02, com diferentes configurações.



Fonte: Elaborado pelo autor (2016)

A Figura 22 apresenta o consumo na camada do *hypervisor* por processos do usuário, ou seja, os processos diversos referentes as aplicações. Mesmo tendo o *hypervisor* fazendo consumo de *CPU*, através da criação, inicialização e remoção das máquinas virtuais, não foi detectado um grande consumo de *CPU*. Além disso, não foi percebido diferenças de impacto no consumo segundo oferta de serviço e número de instâncias.

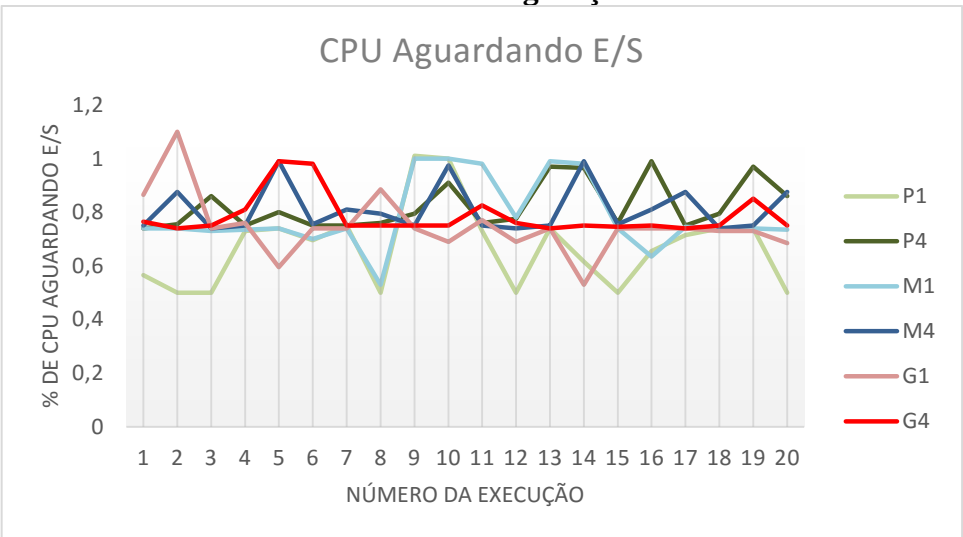
Figura 22 – CPU consumida por processos do usuário no servidor CSGL01, com diferentes configurações.



Fonte: Elaborado pelo autor (2016)

Conforme pode ser visualizado na Figura 23, a porcentagem de *CPU* aguardando Entrada/Saída no armazenamento é mínima, já que todos os arquivos referentes a cada máquina virtual criada estão na camada de armazenamento. Neste caso, o consumo ficou entre 0,4% e 1,1%.

Figura 23 – Porcentagem de *CPU* aguardando disco no servidor CSGL02, com diferentes configurações.

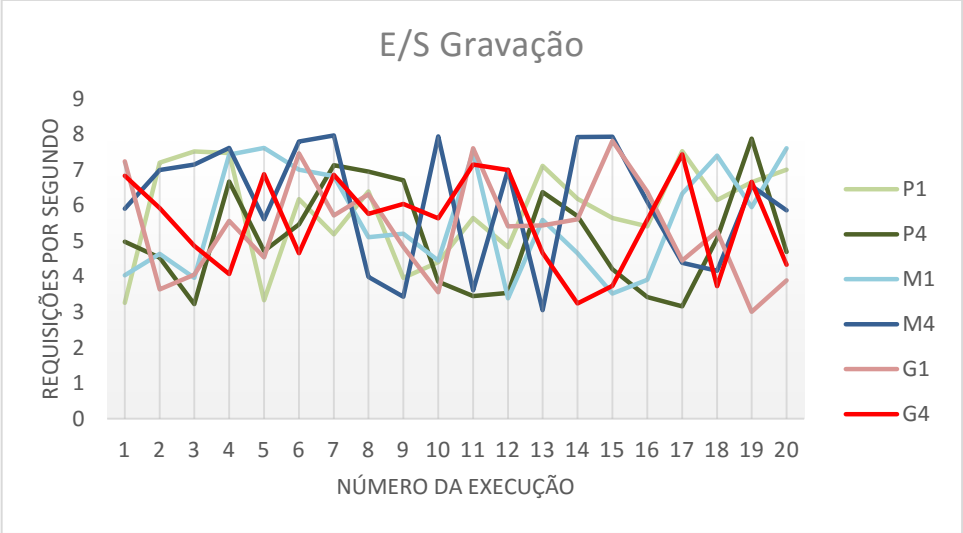


Fonte: Elaborado pelo autor (2016)

A Figura 24 apresenta o número de requisições de gravação por segundo, e seguindo os dados relativos a porcentagem de *CPU* aguardando disco, o valor foi pequeno. Nesta camada,

com a configuração proposta no trabalho, existem apenas gravações relativas a *logs* de registros das máquinas, além de gravações realizadas pelo próprio sistema.

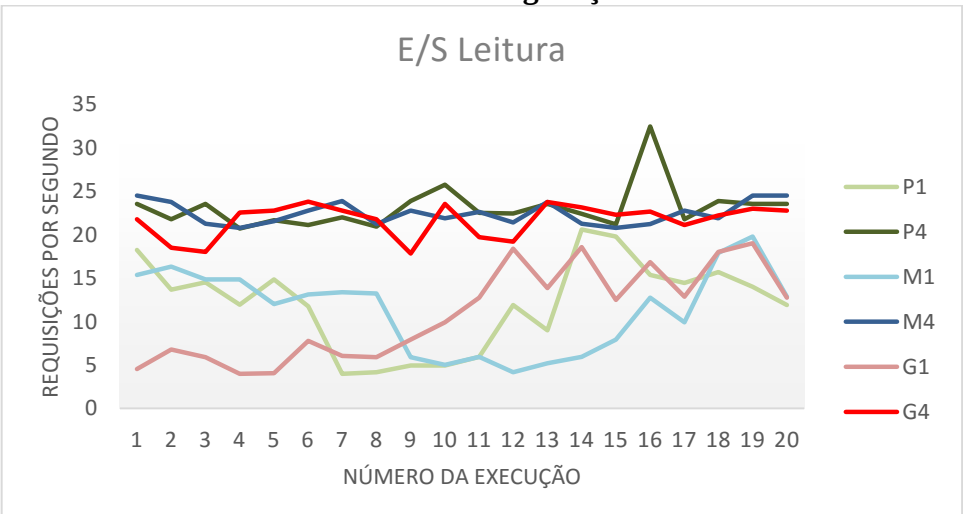
Figura 24 – Número de requisições de gravação por segundo no servidor CSGL02, com diferentes configurações.



Fonte: Elaborado pelo autor (2016)

A Figura 25 apresenta as requisições de leitura para a camada de *hypervisor*, sendo possível visualizar as mesmas características das requisições de gravação: pouca variação entre as diferentes ofertas de serviço e instâncias.

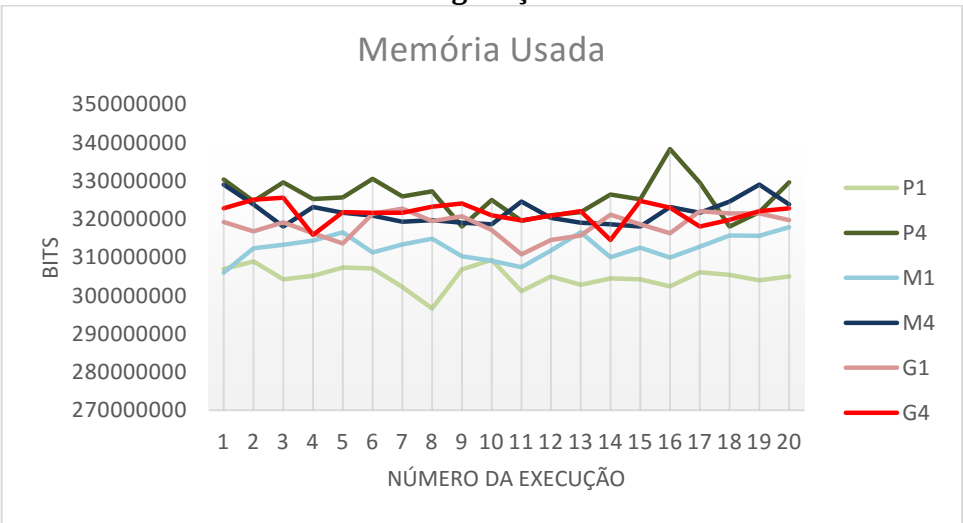
Figura 25 – Número de requisições de leitura por segundo no servidor CSGL02, com diferentes configurações.



Fonte: Elaborado pelo autor (2016)

Na Figura 26 é possível visualizar a quantidade de memória utilizada no servidor CSGL02, e os números obtidos nas execuções tiveram uma diferença mínima levando em consideração a oferta de serviço e instâncias simultâneas. Era esperado uma grande diferença entre todas as configurações possíveis, já que como haviam diferenças nas configurações de memória nas ofertas de serviços P (512MB), M (1024MB) e G (2048MB), aliados a várias instâncias sendo executadas ao mesmo tempo, a expectativa era que esse item apresentasse uma grande diferença. Porém foi identificado que o XenServer, assim que é iniciado, aloca uma parte da memória RAM para o uso das máquinas virtuais (DomU), ou seja, ele já deixa a memória com *status* de utilizada para o sistema. Desta forma, os valores obtidos foram similares entre as configurações.

Figura 26 – Quantidade de memória utilizada no servidor CSGL02, com diferentes configurações.



Fonte: Elaborado pelo autor (2016)

O quadro 8 apresenta o desvio padrão referente a média dos resultados obtidos nas 20 execuções, em cada um dos itens avaliados e configuração da camada XenServer (CSGL02).

Quadro 8 – Desvio padrão em cada item, referente aos resultados obtidos na camada XenServer.

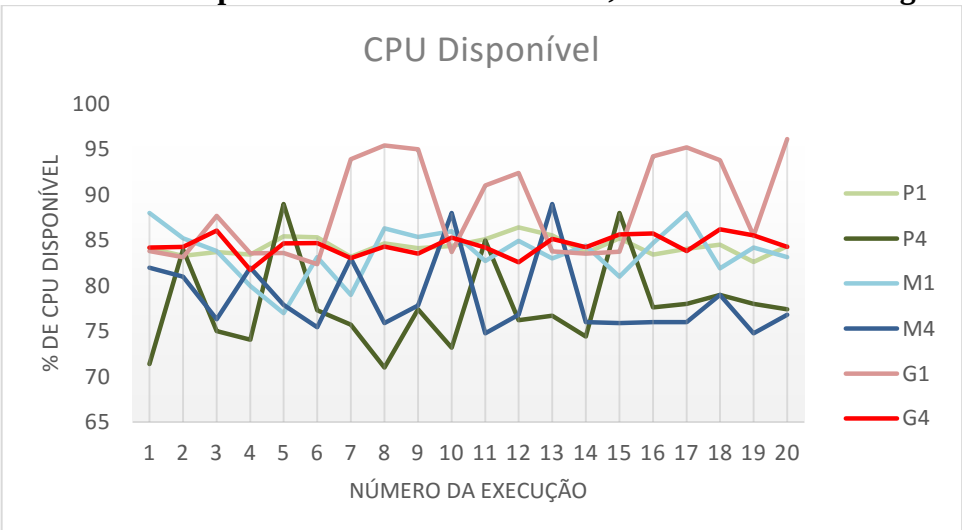
	P1	P4	M1	M4	G1	G4
CPU Disponível	1,03	0,24	1,57	0,33	0,70	0,32
CPU Usuários	0,79	0,25	0,74	0,29	0,39	0,27
CPU Aguardando E/S	0,15	0,08	0,13	0,08	0,11	0,07
E/S Gravação	1,35	1,46	1,50	1,71	1,42	1,31
E/S Leitura	5,11	2,54	4,74	1,30	5,30	1,93
Memória Usada	28,81	49,24	31,68	33,00	31,94	28,51

Fonte: Elaborado pelo autor (2016)

4.3.3 Resultados Individuais da camada OpenFiler

Nas Figuras 27 a 32 são apresentados os resultados referentes ao servidor CSGL04 (OpenFiler), camada responsável por armazenar todos os arquivos referentes as máquinas virtuais criadas pela camada do *hypervisor*. A Figura 26 apresenta a porcentagem de CPU disponível nas execuções, onde apresenta um consumo maior que as outras duas camadas. Mesmo com o serviço de armazenamento NFS não apresentando alto consumo, conforme pode ser percebido na Figura 27, o consumo é justificado pela espera de leitura e gravação em disco, visualizado na Figura 28.

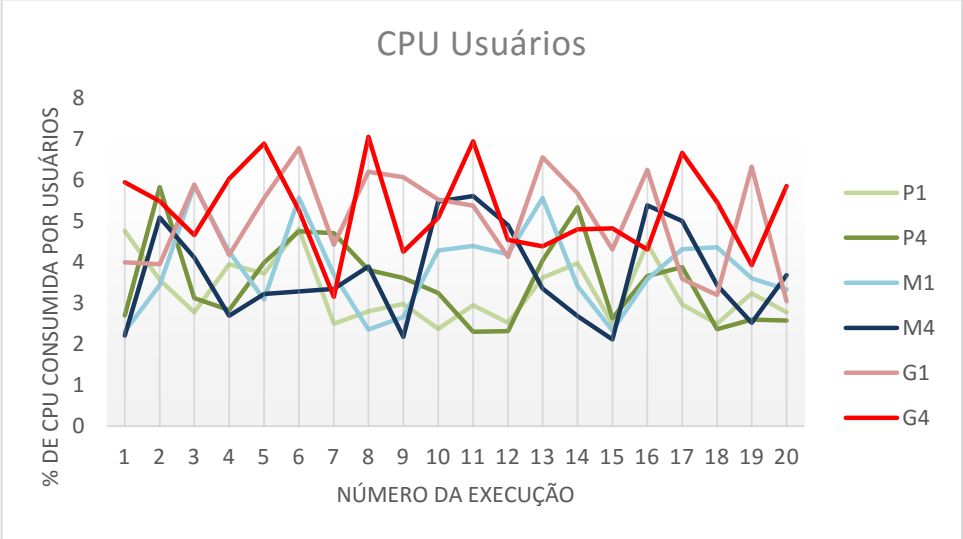
Figura 27 – CPU Disponível no servidor CSGL04, com diferentes configurações.



Fonte: Elaborado pelo autor (2016)

A Figura 28 apresenta o consumo de CPU por processos do usuário, e é possível visualizar que o serviço de armazenamento executado pelo OpenFiler neste servidor não apresenta grande consumo de CPU, já que a média ficou entre 2% e 7% de consumo.

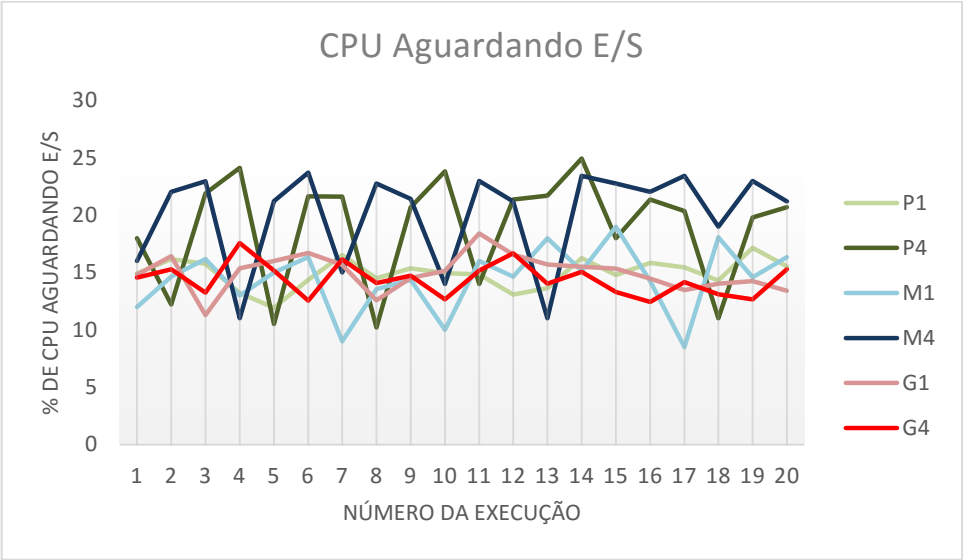
Figura 28 – CPU consumida por processos do usuário no servidor CSGL01, com diferentes configurações.



Fonte: Elaborado pelo autor (2016)

A Figura 29 apresenta o ponto mais relevante relacionado a consumo de *CPU* nesta camada, já que identifica o motivo de um consumo elevado de *CPU*, que é gerado pela espera da leitura e gravação em disco. Isto ocorre devido ao S.O. alocar outros processos para serem executados, enquanto existe processos bloqueando a Entrada/Saída de Leitura/Gravação em disco. Devido ao grande número de requisições que as camadas superiores solicitam para a camada de armazenamento realizar – como por exemplo, na criação de uma máquina virtual, onde o disco virtual é criado nesta camada – é compreensível que exista uma sobrecarga em disco que ocasiona um alto consumo de *CPU*.

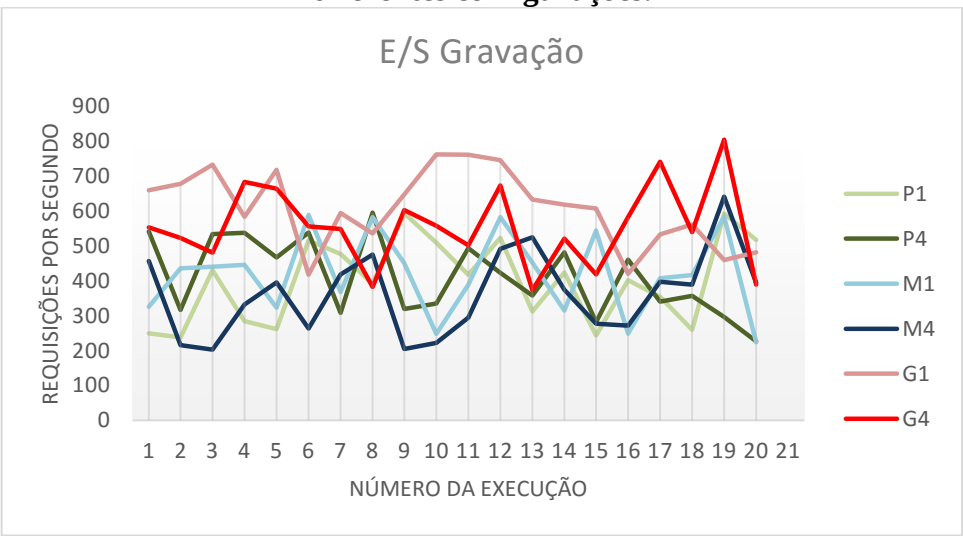
Figura 29 – Porcentagem de CPU aguardando disco no servidor CSGL04, com diferentes configurações.



Fonte: Elaborado pelo autor (2016)

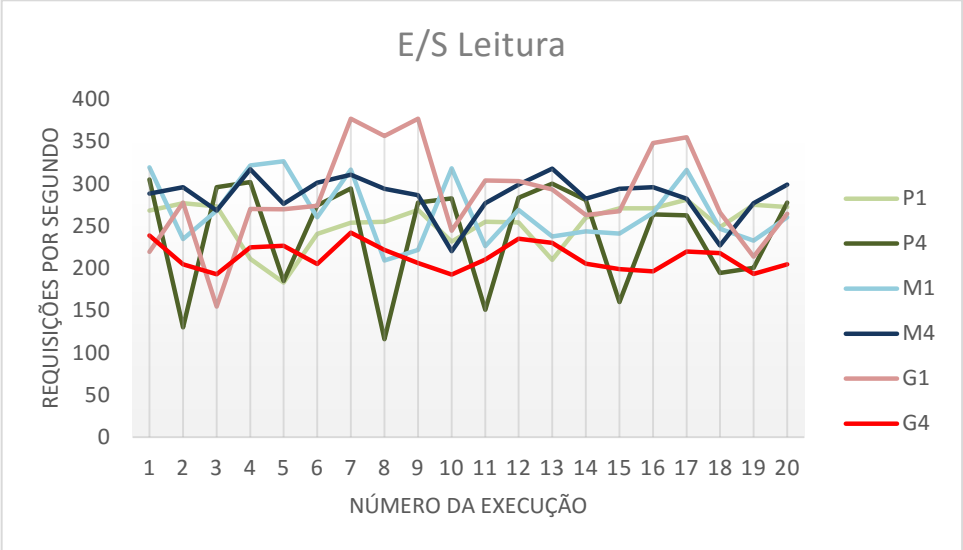
A Figura 30 apresenta o número de requisições por segundo de gravação em disco, demonstrando um valor superior as demais camadas neste item, devido ao fato que toda rotina relacionada as máquinas virtuais serem concentradas nesta camada, da mesma forma do consumo de Entrada e Saída de Leitura, visualizado na Figura 31.

Figura 30 – Número de requisições de gravação por segundo no servidor CSGL04, com diferentes configurações.



Fonte: Elaborado pelo autor (2016)

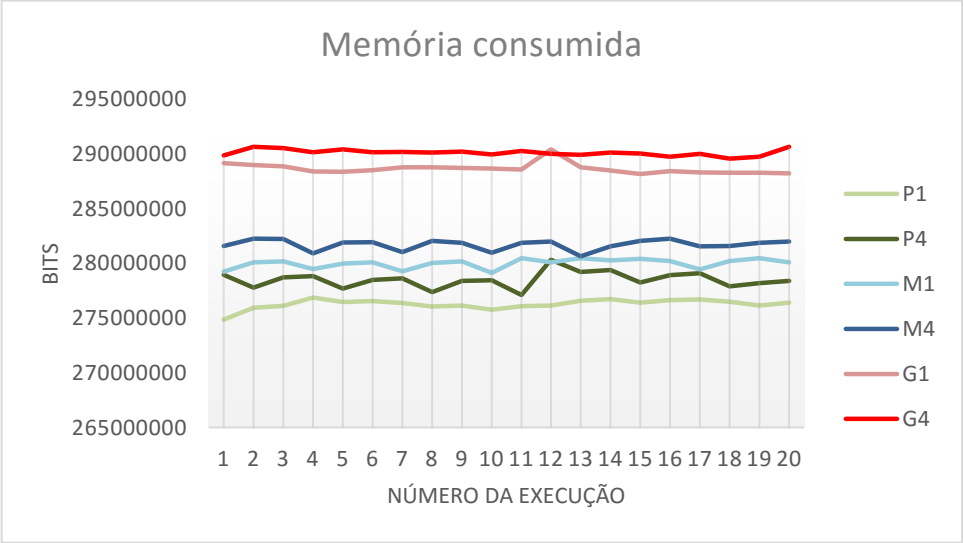
Figura 31 – Número de requisições de leitura por segundo no servidor CSGL04, com diferentes configurações.



Fonte: Elaborado pelo autor (2016)

A Figura 32 apresenta o consumo de memória na camada do *hypervisor*, e é possível verificar o crescimento do consumo de memória conforme as configurações, mesmo que de forma discreta. Este aumento no consumo de memória está associado as atividades que os serviços realizados neste servidor requerem – ainda que em quantidade pequena.

Figura 32 – Quantidade de memória utilizada no servidor CSGL04, com diferentes configurações.



Fonte: Elaborado pelo autor (2016)

O quadro 9 apresenta o desvio padrão referente a média dos resultados obtidos nas 20 execuções, em cada um dos itens avaliados e configuração da camada OpenFiler (CSGL04).

Quadro 9 – Desvio padrão em cada item, referente aos resultados obtidos na camada OpenFiler.

	P1	P4	M1	M4	G1	G4
CPU Disponível	0,98	4,86	2,68	4,21	5,38	1,18
CPU Usuários	0,71	1,04	1,01	1,14	1,19	1,10
CPU Aguardando E/S	1,30	4,82	2,86	4,19	1,63	1,47
E/S Gravação	117,4	107,0	117,6	120,89	111,71	120,62
E/S Leitura	26,5	31,3	37,8	25,71	56,42	15,01
Memória Usada	29,6	76,8	71,4	48,87	49,11	29,99

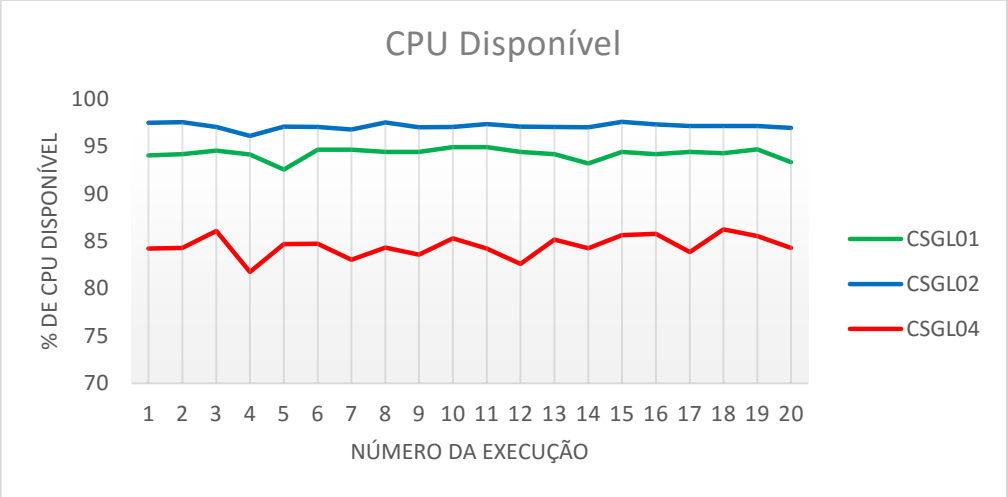
Fonte: Elaborado pelo autor (2016)

4.3.4 Comparação entre as 3 camadas

Os resultados mais relevantes obtidos nos testes são apresentados nas Figuras 33 a 38, com o comparativo entre as camadas envolvidas no estudo. Através da análise destes gráficos, foi possível avaliar os recursos mais consumidos, de acordo com as características dos serviços executados em cada camada. Para realizar este comparativo, foi utilizada apenas a configuração de oferta de serviço grande com 4 instancias simultâneas.

Conforme apresentado na Figura 33, o consumo de CPU durante os testes nas máquinas CSGL01 e CSGL02 foi inferior do que na máquina CSGL04. Isto significa que o equipamento que hospeda o serviço de armazenamento NAS Openfiler consumiu mais *CPU* durante os testes, pelo fato do serviço de armazenamento de dados realizar muita leitura e/ou gravação em disco, fazendo o consumo de *CPU* aumentar consideravelmente. É importante ressaltar ainda que o disco presente neste equipamento é um disco SATA 500*GB* de 7200Rpm.

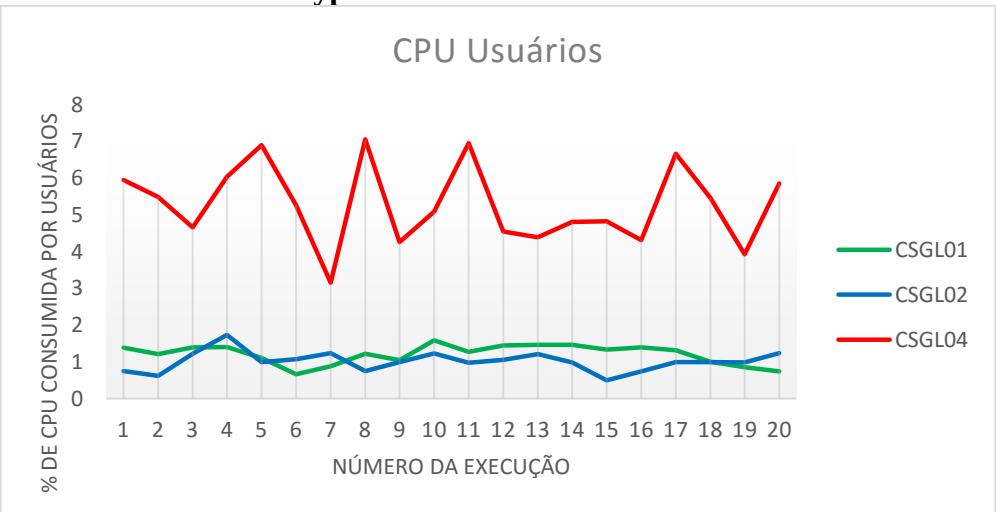
Figura 33 – CPU Disponível nas camadas do orquestrador, hypervisor e armazenamento.



Fonte: Elaborado pelo autor (2016)

A Figura 34 apresenta o consumo de processos de usuário, ou seja, exclusivamente a quantidade que os serviços propriamente consomem, e é possível identificar que, em todos os casos, são serviços com pouco consumo de *CPU*, com consumo inferior a 7%. Porém a camada de armazenamento apresenta um consumo ligeiramente superior aos demais.

Figura 34 – CPU consumida por processos do usuário nas camadas do orquestrador, hypervisor e armazenamento.

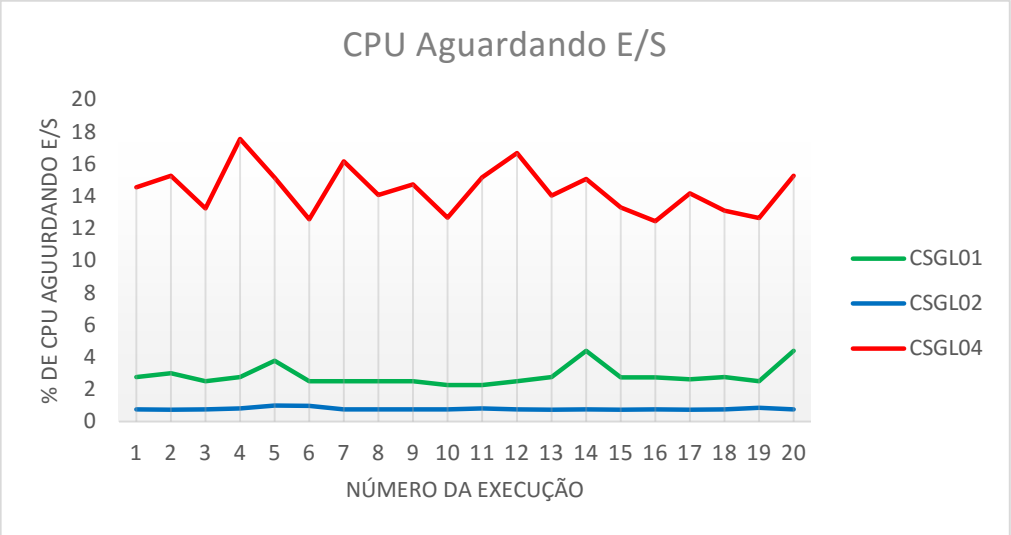


Fonte: Elaborado pelo autor (2016)

A Figura 35 expõe o motivo de maior consumo de *CPU* pela camada de armazenamento, já que ficou com média de aproximadamente 15% no que refere ao consumo de *CPU* devido a

espera por recursos de disco, enquanto a média das outras camadas ficou em aproximadamente 2,5%.

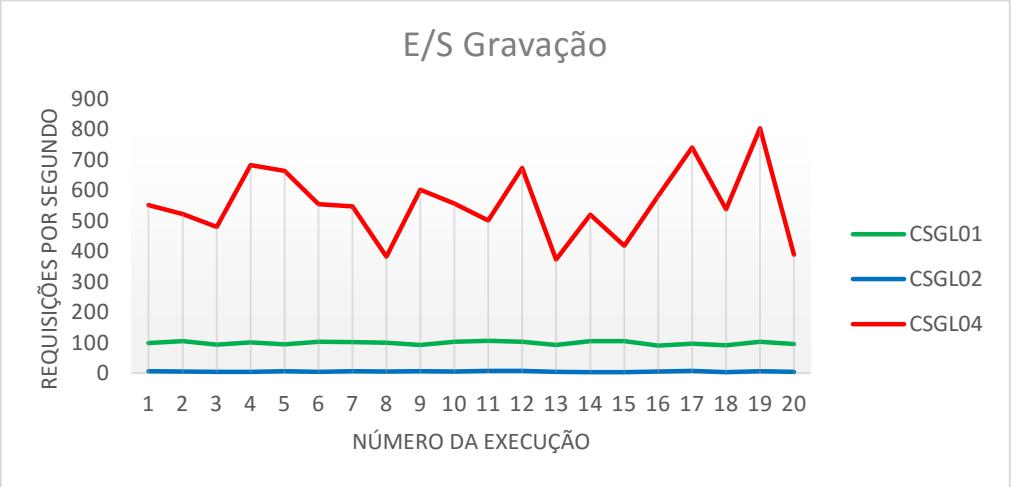
Figura 35 – Porcentagem de *CPU* aguardando disco nas camadas do orquestrador, *hypervisor* e armazenamento.



Fonte: Elaborado pelo autor (2016)

A figura 36 mostra o consumo de E/S de gravação para os equipamentos CSGL01, CSGL02 e CSGL04. Conforme esperado, o consumo deste recurso no servidor CSGL04 foi cerca de 7 vezes maior devido a quantidade de gravações que é requisitado ao disco. Neste ponto vale ressaltar que este servidor oferece serviço de armazenamento primário para o orquestrador Apache CloudStack, armazenando os arquivos referentes às instâncias (VMs). Ou seja, toda a vez que o orquestrador provisiona, cria um *snapshot*, inicializa, desliga ou elimina uma nova instância ele se comunica com a camada do *hypervisor* – no caso deste estudo, o XenServer – que envia as requisições para a camada de armazenamento, o que provoca um alto número de solicitações de E/S gravação nos discos internos presentes neste servidor.

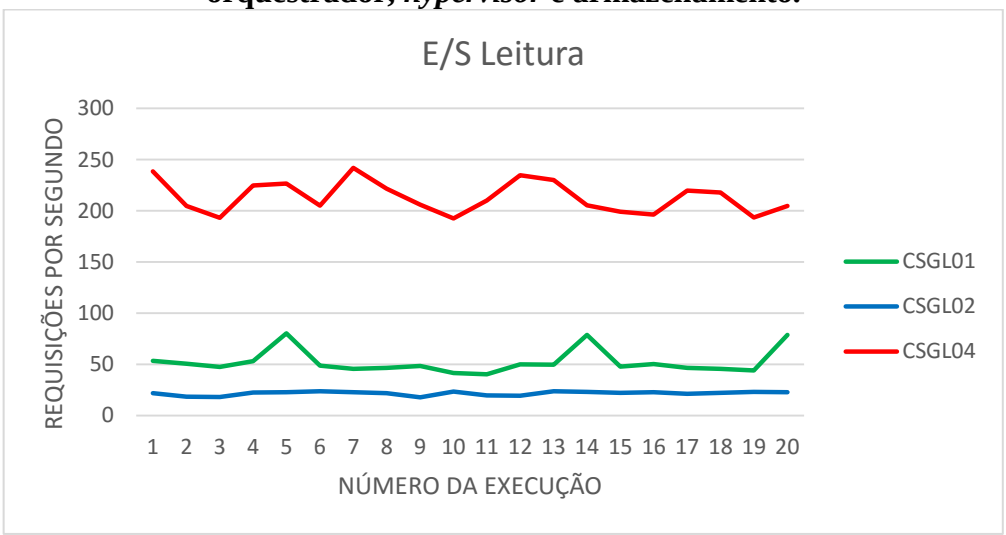
Figura 36 – Número de requisições de gravação por segundo nas camadas do orquestrador, hypervisor e armazenamento.



Fonte: Elaborado pelo autor (2016)

O consumo de E/S para leitura de disco segue a mesma tendência apresentada anteriormente, conforme pode ser visualizado na Figura 37.

Figura 37 – Número de requisições de leitura por segundo nas camadas do orquestrador, *hypervisor* e armazenamento.



Fonte: Elaborado pelo autor (2016)

Por fim, a figura 38 apresenta o consumo de memória nas 3 camadas, e conforme pode ser visualizado, o servidor CSGL02 (XenServer) consumiu mais memória que os demais servidores. Isto pode ser justificado por vários pontos, que serão analisados a seguir:

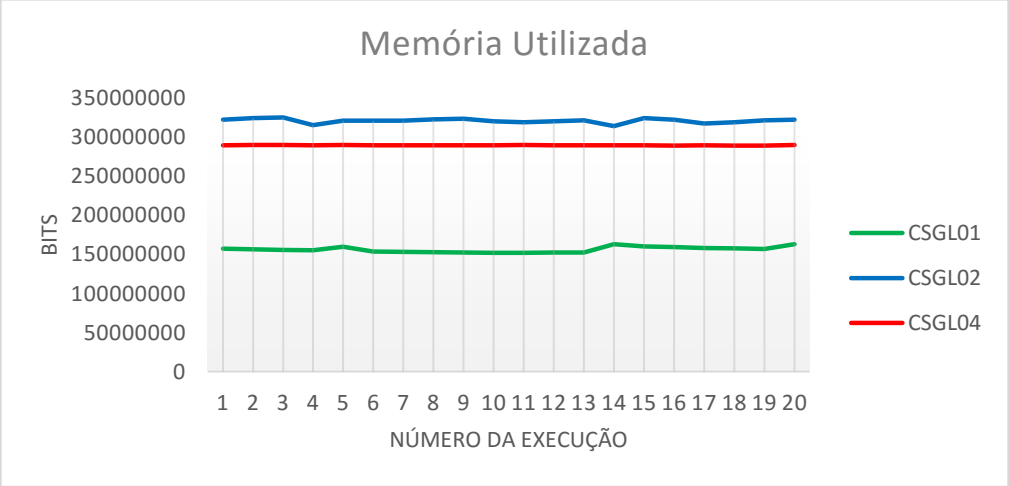
- Segundo Kwon (2013), três componentes contribuem para o consumo de memória de um *host* XenServer, sendo eles (a) a memória consumida pelo próprio *hypervisor*

XenServer, (b) pelo *Control Domain* (dom0) sobre o *host* e (c) XenServer Crash Kernel. A quantidade de memória utilizada pelo *hypervisor* será maior para os *hosts* com mais memória;

- Conforme são criadas mais instâncias, automaticamente é consumido mais memória do *host* físico (dom0), mas não de forma proporcional, devido a maneira que o XenServer aloca memória para as VMs;
- No ambiente criado para os testes, os *hosts* possuem 8GB de memória cada um, porém no momento do teste específico com este *host*, foi configurado para realizar carga de instâncias apenas no host CSGL02.

É possível observar também na figura 37 um alto consumo de memória no *host* CSGL04 (Openfiler), que pode ser justificado pelo alto consumo de E/S de gravação em disco, o que poderia ter consequência no consumo de memória deste *host*.

Figura 38 – Quantidade de memória utilizada nas camadas do orquestrador, *hypervisor* e armazenamento.



Fonte: Elaborado pelo autor (2016)

A síntese dos resultados obtidos no comparativo entre as três camadas pode ser visualizada no Quadro 10, que apresenta a média aritmética das 20 execuções para os itens analisados, em cada uma das camadas. Desta forma fica possível evidenciar o consumo elevado na E/S para gravação e leitura na camada de armazenamento – entre 5 e 10 vezes superior ao consumo do mesmo item nas outras camadas – que resulta também em um número elevado de *CPU* utilizada, devido à espera de E/S.

Quadro 10 – Síntese dos resultados obtidos no comparativo entre camadas.

	CSGL01 (CloudStack)	CSGL02 (XenServer)	CSGL04 (OpenFiler)
***CPU* Disponível (%)**	93,8	96,7	84,7
***CPU* Consumida pelo Usuário (%)**	1,3	1,2	5,8
***CPU* Aguardando E/S (%)**	3,2	1,1	15,5
E/S Gravação (RPS)	97,3	11,3	560
E/S Leitura (RPS)	54,1	22,8	223
Consumo de Memória (MB)	1540	3439	2833

Fonte: Elaborado pelo autor (2016)

4.4 Testes utilizando Disco de Estado Sólido na Camada de Armazenamento

Após realização dos testes definidos no escopo inicial do trabalho, observou-se a importância da camada de armazenamento para um bom desempenho no provisionamento de instâncias em uma Nuvem *IaaS*. Desta forma, optou-se por realizar novos testes, substituindo o disco HDD SATA 7200RPM alocado no servidor da camada de armazenamento por um disco de estado sólido (SSD[22]), possibilitando visualizar o ganho de desempenho para a solução como um todo. Foi realizado a configuração do sistema de armazenamento compartilhado OpenFiler com as mesmas configurações e características desenvolvidas anteriormente, procurando assim manter uma similaridade entre os testes.

Em testes preliminares de desempenho utilizando o software CrystalDiskMark[23] utilizando um disco HDD SATA2 7200RPM de 1TB e posteriormente um SSD de 120GB, foi possível perceber o desempenho de leitura até 12 vezes superior no SSD, e até 10 vezes superior para escrita, conforme pode ser visualizado no Quadro 11.

[22] Segundo ZACK (2012), a unidade de estado sólido (SSD), também conhecido como um disco de estado sólido, é um dispositivo de armazenamento que utiliza montagens de circuitos integrados como memória para armazenar dados persistentemente.
[23] CrystalDiskMark é um software de *benchmark* de disco (http://crystalmark.info/software/CrystalDiskMark/)

Quadro 11 – Testes de desempenho entre HDD e SSD – leitura e gravação.

	Leitura (MB/s)		Escrita (MB/s)	
	HDD	**SSD**	**HDD**	**SSD**
Sequencial	77,79	412,84	74,19	318,39
Aleatório	25,61	371,11	26,12	228,19

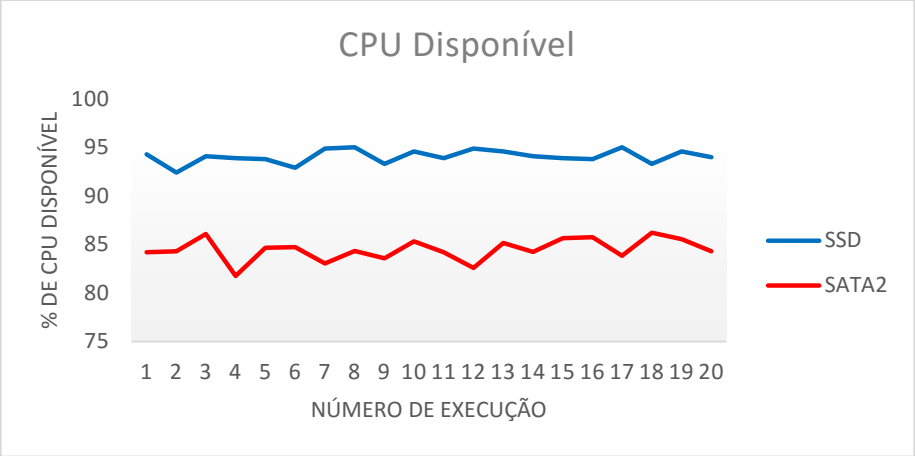
Fonte: Elaborado pelo autor (2016)

Após realizar a instalação e configuração do sistema OpenFiler no servidor CSGL04, responsável pelo armazenamento primário do Apache CloudStack, foram realizados novos testes utilizando o StackAct, que apresentaram um desempenho consideravelmente superior a configuração anterior, utilizando disco HDD SATA2. Para realizar um novo comparativo, foi utilizada apenas a configuração de oferta de serviço grande com 4 instâncias simultâneas, realizando a coleta de dados apenas na camada de armazenamento.

O tempo para execução da rotina completa do StackAct já foi um indicativo da melhora de desempenho, já que a média aritmética foi de 121 segundos, contra 288 segundos com a configuração anterior, ou seja, uma redução de 57,9%.

As Figuras 39 a 44 apresentam o comparativo dos mesmos itens selecionados anteriormente, entre a configuração anterior – com disco HDD SATA2 – e a nova configuração, utilizando disco sólido. A Figura 39 apresenta a porcentagem de *CPU* disponível no ambiente de armazenamento primário, onde o consumo de *CPU* no novo cenário apresentou em média 2,7 vezes inferior, confirmando que a leitura e gravação em disco é um fator que impacta diretamente no consumo de *CPU*.

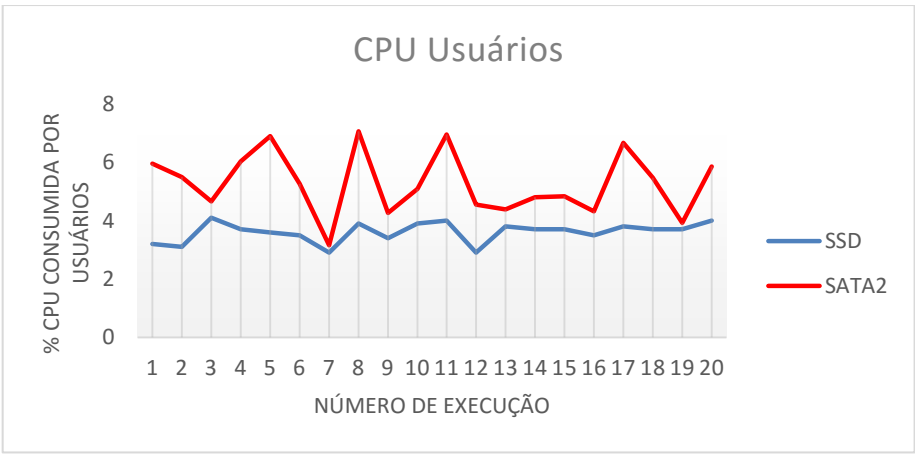
Figura 39 – CPU Disponível na camada de armazenamento, com disco HDD SATA2 e SSD.



Fonte: Elaborado pelo autor (2016)

A Figura 40 apresenta o consumo de *CPU* por processos de usuário, e como já era esperado, os valores são similares, já que os processos existentes nos 2 cenários são exatamente os mesmos: processos do sistema e processos executados pelos serviços inicializados pelo OpenFiler.

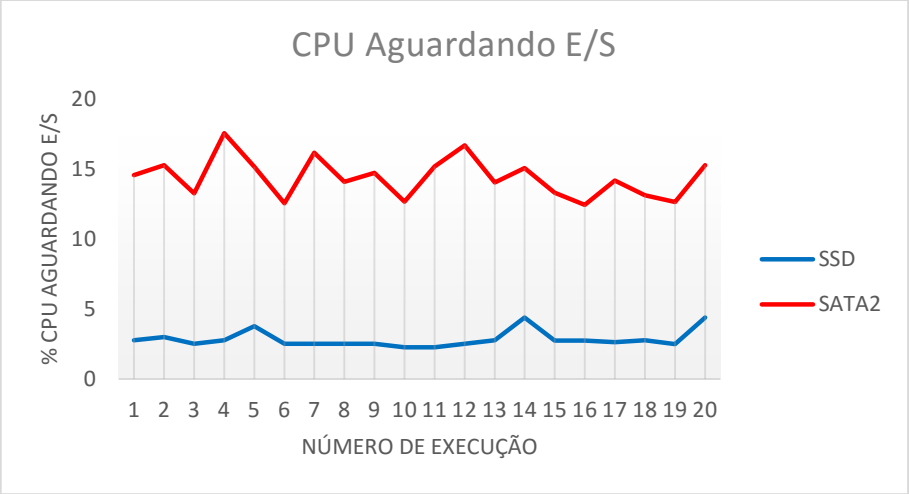
Figura 40 – CPU consumida por processos do usuário na camada de armazenamento, com disco HDD SATA2 e SSD.



Fonte: Elaborado pelo autor (2016)

Os valores mais representativos na realização dos novos testes são exibidos na Figura 41. O valor referente a porcentagem de *CPU* aguardando disco impacta diretamente ao desempenho da solução total da Nuvem *IaaS*, e este item apresenta um valor em média 5 vezes superior com a antiga configuração, justificado pela tecnologia de disco utilizada nos testes.

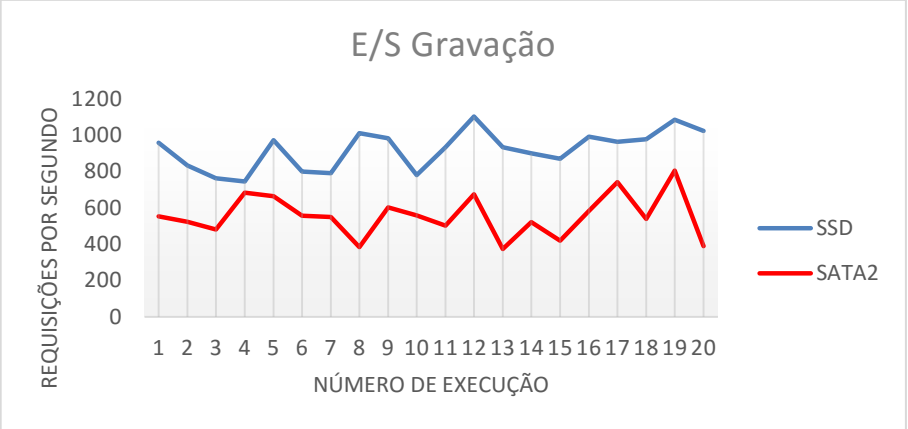
Figura 41 – Porcentagem de CPU aguardando disco na camada de armazenamento, com disco HDD SATA2 e SSD.



Fonte: Elaborado pelo autor (2016)

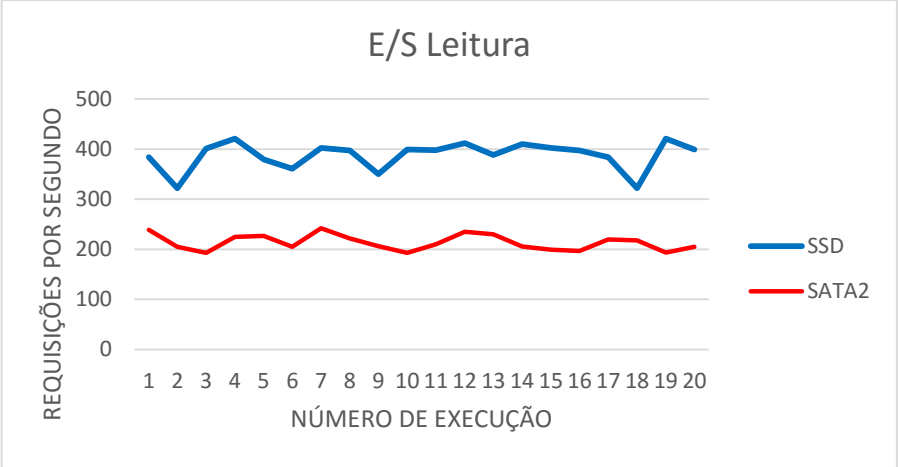
As Figuras 42 e 43 apresentam a diferença de requisições por segundo para leitura e gravação no disco, com crescimento de em média 83% nas requisições de gravação e 77% nas requisições de leitura em disco.

Figura 42 – Número de requisições de gravação por segundo na camada de armazenamento, com disco HDD SATA2 e SSD.



Fonte: Elaborado pelo autor (2016)

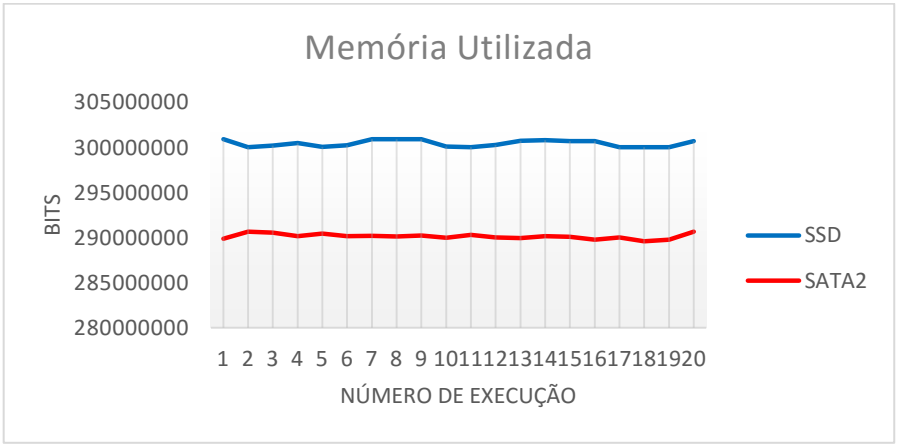
Figura 43 – Número de requisições de gravação por segundo na camada de armazenamento, com disco HDD SATA2 e SSD.



Fonte: Elaborado pelo autor (2016)

Finalmente a Figura 44 apresenta o consumo de memória, que em média foi considerado similar, já que a troca de disco não afeta diretamente o consumo de memória dos servidores.

Figura 44 – Quantidade de memória utilizada na camada de armazenamento, com disco HDD SATA2 e SSD.



Fonte: Elaborado pelo autor (2016)

Após realização dos testes complementares utilizando disco SSD, e realizando um comparativo com os resultados obtidos anteriormente, foi possível reforçar a importância da tecnologia de disco na camada de armazenamento, gerando uma melhora considerável no desempenho relacionado aos processos contemplados no StackAct.

4.5 Considerações Finais

Com o objetivo de validar o StackAct, este capítulo apresentou uma análise dos resultados obtidos com a execução do mecanismo. É importante reforçar que estes resultados podem não demonstrar com exatidão o consumo de recursos nas camadas envolvidas, devido a limitação do ambiente – grandes ambientes de Nuvem podem ter até algumas centenas de servidores distribuído entre orquestrador, *hypervisor* e armazenamento.

Na análise dos resultados, foi possível detectar que não houve alteração significativa no consumo de recursos entre os diferentes tipos de configurações propostos, devido ao enfileiramento de ações definidas pela camada do *hypervisor*, gerando assim um consumo similar e apenas um tempo de execução maior. Já no comparativo entre as camadas, foi possível evidenciar o alto consumo de recursos de armazenamento na camada do sistema OpenFiler e que onerou inclusive o consumo de CPU do servidor, fato que foi minimizado com a substituição do disco empregado nos testes – SATA– por uma tecnologia com melhor desempenho, como uma unidade de estado sólido (SSD).

CAPÍTULO 5 – Conclusão, Contribuições e Trabalhos Futuros

5.1 Considerações Iniciais

O crescimento na utilização da Computação em Nuvem torna inevitável uma concentração de esforços para tornar os ambientes cada vez mais rápidos e otimizados. Na modalidade *IaaS*, onde vários orquestradores surgem procurando trazer facilidades para os utilizadores, foi possível detectar que dois orquestradores aparecem com maior visibilidade: o OpenStack e o Apache CloudStack. Porém, ao mesmo tempo que diversos estudos já foram e estão sendo realizados com o primeiro, o Apache CloudStack ainda possui diversos aspectos relacionados a funcionalidades, módulos, desempenho, que podem e devem ser explorados, principalmente pelo destaque que recebeu após se tornar um TLP da Apache Software Foundation, em 2012.

Além disso, é importante ressaltar a importância das outras duas camadas para o desempenho de um ambiente *IaaS*: a camada do *hypervisor*, que neste estudo foi utilizado o XenServer, e a camada de armazenamento de dados, sendo utilizado o OpenFiler.

Este estudo apresentou um mecanismo de coleta de dados relacionados ao consumo de recursos computacionais em uma Nuvem *IaaS* multicamadas, denominado StackAct, utilizando o orquestrador Apache CloudStack, o *hypervisor* XenServer e o sistema de armazenamento Openfiler, além da execução de testes de desempenho utilizando o mecanismo criado em um ambiente de Nuvem privado, gerando importantes conclusões:

1. Através de ferramentas *opensource*, como SAR, KSAR e a ferramenta de linha de comando CloudMonkey – além da utilização de Shell Script - foi possível elaborar um mecanismo de coleta de desempenho que funciona em ambientes utilizando Apache CloudStack, e qualquer *hypervisor* e sistema de armazenamento de arquivos que seja

executado em ambiente que possibilite a execução do SAR, com o conjunto de pacotes SYSSTAT.

2. Utilizando deste mecanismo em ambiente Apache CloudStack, é possível realizar a coleta para posterior análise de consumo de diversos recursos computacionais – todos que a ferramenta SAR possibilitar o monitoramento, como *Load Average* (1, 5 e 10 minutos), atividade das interfaces de rede, consumo NFS, *Page Swap*, inicialização de processos do sistema, entre outros. Conforme definido no escopo inicial para elaboração deste artigo, foi realizado análise apenas em total de *CPU* disponível, total de *CPU* consumida por processos de usuário, total de *CPU* aguardando E/S em disco, E/S de leitura de dados, E/S de gravação de dados e consumo de memória, utilizando o limite de 4 processos de instância (criação e inicialização, *snapshot*, *shutdown* e exclusão da instância) sendo criados simultaneamente, devido às limitações impostas pelo ambiente utilizado nos estudos.
3. Quanto aos resultados dos testes realizados, mesmo considerando o ambiente disponível limitado em relação aos recursos computacionais, foi possível demonstrar com sucesso o mecanismo criado para avaliação de desempenho em um ambiente multicamadas utilizando Apache CloudStack. Foi possível constatar, por exemplo, o alto consumo de disco na camada de armazenamento de dados, em especial E/S de leitura e gravação de dados. Foi possível constatar também o alto consumo de memória na camada *hypervisor*, que pode ser justificada pela alocação do próprio *hypervisor* além das *VMs* que estão sendo criadas e utilizadas no processo. Além disso, foi constatado que no sistema multicamadas proposto neste estudo, a camada do orquestrador – no caso o Apache CloudStack é a que apresenta menor consumo de recursos computacionais, levando em consideração os elementos selecionados para análise.

5.2 Contribuições

Os estudos realizados para elaboração deste trabalho, o mecanismo desenvolvido e os resultados obtidos em ambiente de testes possibilitaram as seguintes contribuições para a área:

- **Computação em Nuvem e Virtualização:** Através do estudo nos conceitos explorados, foi possível reforçar as diferenças entre as funcionalidades de ambas as camadas, que são comumente confundidas. A definição de camadas utilizada neste trabalho, possibilitou

um melhor entendimento nas atividades de cada recurso existente em uma estrutura *IaaS*, sendo possível identificar a importância da camada de virtualização em um cenário de Computação em Nuvem;

- **StackAct:** Devido ao número reduzido de estudos encontrados para avaliação de desempenho e consumo de recursos em uma Nuvem *IaaS* com o orquestrador Apache CloudStack, o mecanismo pode-se tornar uma referência para estudos futuros. Possui características flexíveis quanto a camada alvo a ser analisada, a dimensão da Nuvem, além dos recursos que se deseja avaliar. Além disso, foram utilizadas apenas ferramentas *opensource*, o que tende a tornar o mecanismo ainda mais flexível;
- **Resultados:** Mesmo considerando o cenário construído com o propósito de validar o mecanismo, limitando o desenvolvimento de testes mais amplos, foi possível detectar um padrão de comportamento nos serviços propostos para os itens selecionados, com a camada do orquestrador realizando pouco consumo dos recursos monitorados, a camada do *hypervisor* fazendo alto consumo de memória, e por fim a camada de armazenamento com alto consumo de E/S de gravação e leitura.

Além disso, realizando um comparativo dos principais tópicos deste estudo com os trabalhos levantados no item 2.7 – Trabalhos Relacionados – é possível destacar:

- No artigo de Reddy e Rajamani (2014), os autores utilizaram um framework já disponível no mercado – SIGAR – que não possui uma função específica para levantamento de dados em Nuvens IaaS. Além disso, apesar de terem utilizado 3 diferentes *hypervisors* em *clusters* diferentes, não houve preocupação aparente com o consumo na camada do próprio orquestrador – focando exclusivamente na camada dos *hypervisors*. O StackAct foi proposto como um mecanismo que utiliza de várias ferramentas *opensource*, e que pode também ser multi-*hypervisor*, apesar de ser utilizado nos testes apenas com XenServer. Além disso, foram analisados os consumos nas 3 camadas envolvidas, possibilitando um entendimento de todo cenário proposto.
- No trabalho de Kai, Weiqin, Liping e Chao (2013), os autores enfatizam que o mecanismo foi elaborado para possibilitar o monitoramento a nível físico e virtual, porém não descrevem o funcionamento da ferramenta – linguagem, camada, forma de obtenção dos dados, entre outros. Diferente do trabalho anterior, este estudo foca apenas na captação de recursos computacionais na camada do orquestrador, ignorando a camada do

hypervisor e de armazenamento. O StackAct abrange as 3 camadas envolvidas no ambiente.

- No trabalho de Paradowski, Liu e Yua (2014), os autores focaram no comparativo entre as 2 plataformas de Nuvem IaaS, estruturando os testes de forma similar as encontradas neste estudo, porém separando os resultados para cada evento realizado (inclusão, inicialização e exclusão da instância). Os autores utilizaram ambiente virtualizado, ou seja, todo o ambiente CloudStack e Openstack foi construído sobre o software Virtualbox[24] – o que pode comprometer a captação dos dados relativos a consumo de recursos. O StackAct foi desenvolvido usando o CloudMonkey, ferramenta criada para automatizar processos para o Apache CloudStack – por isso não funcionaria com outro orquestrador. Porém no ambiente criado, foram utilizadas máquinas físicas, e a virtualização foi utilizada apenas na camada do *hypervisor*, da mesma forma que é realizado em um ambiente de Nuvem *IaaS* real.

5.3 Trabalhos Futuros

Como proposta para trabalhos futuros, é possível realizar as seguintes sugestões para desenvolvimento e evolução do trabalho realizado:

- Utilizar o mecanismo desenvolvido neste estudo para avaliar uma Nuvem *IaaS* multicamadas, selecionando outro *hypervisor*, como o KVM, e outros sistemas de armazenamento de dados, como o Ceph[25], possibilitando a comparação com os resultados obtidos neste estudo;
- Realizar a avaliação de outros itens disponíveis nos relatórios SAR que não foram selecionados para serem analisados neste artigo, como *Load Average*, atividade das interfaces de rede, consumo NFS, *Page Swap*, inicialização de processos do sistema, entre outros;

[24] VirtualBox é um software de virtualização da Oracle que visa criar ambientes para instalação de sistemas distintos (http://www.virtualbox.org).

[25] O Ceph é um sistema de arquivos distribuído que incorpora a replicação e a tolerância a falhas ao mesmo tempo em que mantém compatibilidade POSIX (http://www.ceph.com).

- Realizar a análise dos dados fazendo a separação de cada um dos eventos contemplados no teste. Desta forma, seria possível avaliar isoladamente o consumo de recursos no momento da criação e inicialização da instância, da criação do *snapshot*, e do desligamento e exclusão da instância, possibilitando uma avaliação individual de cada um dos eventos.
- Realizar integração do StackAct com sistema de escalonamento em um sistema de Nuvem *IaaS,* determinando uma atribuição de tarefas a elementos de processamento com o intuito de otimizar índices de desempenho, como tempo processamento, leitura e escrita em disco ou memória.
- Propõe-se também a utilização da ferramenta em ambiente com mais recursos, como por exemplo a Nuvem Computacional da Amazon, onde é possível testar a escalabilidade da Nuvem com uma configuração de oferta de serviço e número de instâncias maior.

REFERÊNCIAS

AMAZON. **Amazon Elastic Compute Cloud (Amazon EC2)**. Disponível em: <http://aws.amazon.com/pt/ec2/>. Acesso em: 30 outubro 2015.

ABOULNAGA, A., SALEM, K., SOROR, A., FAROOQ, U. **Deploying Database Appliances in the Cloud**, 2009. IEEE Data Eng. Bull., pp. 13–20.

ALI, M., **Exploring the Cloud Evaluating the Possibilities and Limitations of Cloud Computing for a Project Based Organization**, 2014. Master of Science Thesis in the Master's Programme, Chalmers University of Technology.

ANGELES, S. **Virtualization vs. Cloud Computing: What's the Difference?** Disponível em: <http://www.businessnewsdaily.com/5791-virtualization-vs-cloud-computing.html>. Acesso em: 03 Maio 2015.

APACHE FOUNDATION. **Open Source Cloud Computing**. Disponível em: <http://docs.cloudstack.apache.org/projects/cloudstack-administration/en/4.8/>. Acesso em: 02 de novembro de 2015.

BARKAT, A.; SANTOS, A., D. ; HO, T., T., N. **Open Stack and Cloud Stack: Open Source Solutions for Building Public and Private Clouds**. 22-25 Sept. 2014, 16th International Symposium, pp. 429-436, DOI 10.1109/SYNASC.2014.64

BJARDWAJ, S., JAIN, L., JAIN, S. **CLOUD COMPUTING: A STUDY OF INFRASTRUCTURE AS A SERVICE (IAAS)**, 2011, International Journal of Engineering and Information Technology, Vol2 , No. 1, pp. 60-63. ISSN 0975-5292.

BOSS, G., MALLADI, P., QUAN, D., LEGREGNI, **Cloud Computing**, 2013, American Journal of Information Systems. Vol. 2 No. 1, pp. 1-5. DOI: 10.12691/ajis-2-1-1

BUYYA, R. **Introduction to the IEEE Transactions on Cloud Computing**. IEEE TRANSACTIONS ON CLOUD COMPUTING, Vol. 1, N. 1, JAN 2013, pp. 3-21, DOI: 10.1109/TCC.2013.13

BUYYA, R., YEO, C. VNUGOPAL, S. **High Performance Computing and Communications, 2008. HPCC '08. 10th IEEE International Conference**, 2008, pp.: 5-13, DOI: 10.1109/HPCC.2008.172.

CASTILLO, J. A. ,L. ; MALLICHAN, M.; AL-HAZMI,Y. **OpenStack Federation in Experimentation Multi-cloud Testbeds**. Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on, 2013, pp: 51-56, DOI: 10.1109/CloudCom.2013.103

CHEN, G., RedHat. **KVM: Open Virtualization Becomes Enterprise Grade**, 2009. Disponível em <https://openvirtualizationalliance.org/sites/ova/files/resources/files/white_paper_kvm_-_the_rise_of_open_enterprise-class_virtualization.pdf >. Acesso em: 02 de abril de 2015.

CHEN, Q. ; XU, F. **Study on Water Information Cloud of Nanjing Based on CloudStack**. Distributed Computing and Applications to Business, Engineering & Science (DCABES), 2013 12th International Symposium on, pp. 132–136, DOI: 10.1109/DCABES.2013.31

CITRIX, **The Xen Project is Built for Cloud Computing**. Disponível em: <http://www.xenproject.org/users/cloud.html/>. Acesso em: 02 de novembro de 2014.

EUCALYPTUS. **Open source AWS compatible private cloud software**. Disponível em: <http://www.cloudbook.net/resources/stories/the-eucalyptus-open-source-private-cloud>. Acesso em: 02 de novembro de 2014.

GOYAL, M., AHHARWAL, A., GUPTA, P., KUMAR, P., **QoS based trust management model for cloud IaaS,** Proceedings of Second IEEE International Conference on Parallel Distributed and Grid Computing (PDGC), 2012, pp. 843-847, DOI: 10.1109/PDGC.2012.6449933

HU, B.; YU, H. **Research of Scheduling Strategy on OpenStack**. Cloud Computing and Big Data, 2013 International Conference on, 2013, pp. 191-196, DOI: 10.1109/CLOUDCOM-ASIA.2013.89

HUFFERD, J. **IP Storage Protocols: iSCSI**. Disponível em: <http://www.snia.org/sites/default/education/tutorials/2011/spring/networking/HufferdJohn-IP_Storage_Protocols-iSCSI.pdf>, Acesso em: 18 Março 2015.

IBM. **Easy system monitoring with SAR: SAR helps you pinpoint performance bottlenecks** Disponível em: <http://www.ibm.com/developerworks/aix/library/au-unix-perfmonsar.html>, Acesso em: 25 Junho 2016.

KAI, L.; WEIQIN, T.; LIPING, Z.; CHAO, H. **SCM: A Design and Implementation of Monitoring System for CloudStack.** 2013 International Conference on Cloud and Service Computing. 4-6 Nov. 2013, pp. 146-151, INSPEC: 14024443

Keith J., Burkhard N. **The future of cloud computing**. Disponível em: <http://cordis.europa.eu/fp7/ict/ssai/docs-cloudreport-final.pdf>, Acesso em: 18 de Abril de 2015.

KHAN, R., YLITALOT, J. **OpenID Authentication As A Service in OpenStack**, Information Assurance and Security (IAS), 2011 7th International Conference on, 2011, pp. 372-377, DOI: 10.1109/ISIAS.2011.6122782

LEI, Q., JIANG, Y., YANG, M. **Evaluating Open IaaS Cloud Platforms Based upon NIST Cloud Computing Reference Model**, Computational Science and Engineering (CSE), 2014 IEEE 17th International Conference on, pp. 1909-1914, DOI: 10.1109/CSE.2014.350

LENK, A., KLEMS, M., NIMIS, J., TAI, S. **What's Inside the Cloud? An Architectural Map of the Cloud Landscape**, Software Engineering Challenges of Cloud Computing, 2009. CLOUD '09. ICSE Workshop on, Pages: 23-31, DOI: 10.1109/CLOUD.2009.5071529

LIU, X., YANGM H. **Software Reuse in the Emerging Cloud Computing Era**, Edinburgh Napier University, UK, 2012. pp. 149, DOI: 10.4018/978-1-4666-0897-9

LONEA, A., POPESCU, D., PROSTEAN, O. **A Survey of Management Interfaces for Eucalyptus Cloud**, 2012 7th IEEE International Symposium on, pp: 261-266, DOI: 10.1109/SACI.2012.6250013

MELL,P; GRANCE,T. **The NIST definition of cloud computing**. Disponível em: <http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf>. Acesso em: 20 de agosto de 2014.

MONTERO, R., MORENO, R., LLORENTE, I., **IaaS Cloud Architecture: From Virtualized Datacenters to Federated Cloud Infrastructures,** Computer 2012, pp: 65-72, DOI: 10.1109/MC.2012.76

NIST - Programa de Cloud Computing. Disponível em: <http://www.nist.gov/itl/cloud/index.cfm>. Acesso em: 01 Setembro 2015.

NIST - Cloud Computing Service Metrics Description. Disponível em: < http://www.nist.gov/itl/cloud/upload/RATAX-CloudServiceMetricsDescription-DRAFT-20141111.pdf>. Acesso em: 07 de junho de 2016.

OPENNEBULA. **Cloud and data center virtual infrastructure management**. Disponível em: < http://opennebula.org/documentation/tutorials/ >. Acesso em : 02 de novembro de 2015.

OPENSTACK. **What is OpenStack?** Disponível em: <http://docs.OpenStack.org/cactus/OpenStack-compute/admin/content/what-is-OpenStack.html>. Acesso em: 01 de setembro de 2015.

PARADOWSKI, A.; LIU, L.; YUA, B. **Benchmarking the Performance of OpenStack and CloudStack**. 2014 IEEE 17th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing. 10-12 Jun. 2014, pp. 405-412, DOI:10.1109/ISORC.2014.12

PRASAD, A.; RAO, S. **A Mechanism Design Approach to Resource Procurement in Cloud Computing**, in IEEE Transactions on Computers, vol. 63, no. 1, pp. 17-30, Jan. 2014. doi: 10.1109/TC.2013.106

REDDY, P.; RAJAMANI, L. Reddy. **Evaluation of Different Hypervisors Performance in the Private Cloud with SIGAR Framework.** International Journal of Advanced Computer Science and Applications (IJACSA), Volume 5 Issue 2, 2014. DOI 10.14569/IJACSA.2014.050210

REGOLA, N., DUCOM, C. **Recommendations for Virtualization Technologies in High Performance Computing**, Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on, Indianapolis, IN, 2010, pp. 409-416. doi: 10.1109/CloudCom.2010.71

SALAPURA, V., **Cloud computing: Virtualization and resiliency for data center computing**, Computer Design (ICCD), 2012 IEEE 30th International Conference on, Montreal, QC, 2012, pp. 1-2. doi: 10.1109/ICCD.2012.6378606

SALESFORCE. **What is "the cloud"?.** Disponível em: <http://www.salesforce.com/uk/cloudcomputing/>. Acesso em: 18 de maio de 2015

SCHIMIDT, E., JONATHAN, R., EAGLE, A. **How Google Works**, Grand Central Publishing 2014.

SERRANO, N., GALLARDO, G. HERNANTES, **Infrastructure as a Service and Cloud Technologies**, in IEEE Software, vol. 32, no. 2, pp. 30-36, Mar.-Apr. 2015. doi: 10.1109/MS.2015.43

SHAIKH, F. B.; HAIDER, S., **Security threats in cloud computing**, Internet Technology and Secured Transactions (ICITST), 2011 International Conference for, Abu Dhabi, 2011, pp. 214-219.

JADEJA, Y., MODI, K., **Cloud computing - concepts, architecture and challenges**, Computing, Electronics and Electrical Technologies (ICCEET), 2012 International Conference on, Kumaracoil, 2012, pp. 877-880. doi: 10.1109/ICCEET.2012.6203873

SOMANI, G., CHAUDHARY, S. **Load Balancing in Xen Virtual Machine Monitor, Third International Conference, IC3 2010**, pp 62-70, DOI 10.1007/978-3-642-14825-5_6

SOUSA, E.,SILVA, E.,LINS,F., TAVARES, E., MACIEL, P., **Dependability evaluation of cloud infrastructures**, 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC), San Diego, CA, 2014, pp. 1282-1287. doi: 10.1109/SMC.2014.6974091

STOKELY, M. **The FreeBSD Handbook 3rd Edition, Vol. 1: User Guide, FreeBSD Mall**, FreeBSD Mall; 3 edition (March 1, 2004)

VMWARE. **Server Consolidation - Documentation.** Disponível em: <https://www.vmware.com/support/vsphere-hypervisor.html>. Acesso em: 12 Setembro 2015.

WEN,X.,GU, G., LI, Q., GAO, Y., ZHANG, X., **Comparison of open-source cloud management platforms: OpenStack and OpenNebula**, Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on, Sichuan, 2012, pp. 2457-2461. doi: 10.1109/FSKD.2012.6234218

XIAODONG, L,; FOWLEY, F, **A Classification and Comparison Framework for Cloud Service Brokerage Architectures**, in IEEE Transactions on Cloud Computing , vol.PP, no.99, pp.1-1 doi: 10.1109/TCC.2016.2537333

APÊNDICE A – Código do StackAct

```
#!/bin/bash
#Bloco desenvolvido para utilização do StackAct COM interação com o usuário
if [ $# -eq 0 ]; then
	clear
	echo "######  Script de automaticação de processo CloudStack  ######"
	echo "Informe:"
	echo "(1) Para oferta de serviço pequena (1 core/512Mhz/512 RAM):"
	echo "(2) Para oferta de serviço média (2 cores/1Ghz/1024 RAM):"
	echo "(3) Para oferta de serviço grande (2 cores/2Ghz/2048 RAM):"
	echo "(3) Sair!"

	read ofertaservico

	clear
	echo "Informe quantas sessões simultâneas deseja realizar a rotina (de 1 a 4):"
	read sessões
	clear
	echo "Informe o teste desejado:"
	echo "(1) Apenas criação da(s) VM(s):"
	echo "(2) Criação e Snapshot da(s) VM(s):"
	echo "(3) Criação, Snapshot e Exclusão da(s) VM(s):"
	read tipoteste
	clear

#Bloco desenvolvido para utilização do StackAct SEM interação com o usuário
else

	ofertaservico=$1
	sessoes=$2
	tipoteste=$3

fi

#Limitação imposta pelo ambiente desenvolvido
if [ $sessoes -gt 5 ]; then
	echo "Numero de sessões maior que o permitido!"
	exit
fi

#Variaveis para registro do sistema
NOW=$(date +"%m-%d-%Y-%T")
datainicial=`date +%s`
RAND=$RANDOM
```

```
echo "INICIO AS $NOW"
logrotina=/home/gustavo/benchCS3/logrotina.$NOW-$RAND.log
logerro=/home/gustavo/benchCS3/logerro.$NOW.log
logsar=/home/gustavo/benchCS3/sar.$NOW.log

#Definição do alvo a ser executado
ssh 192.168.30.1 '/root/rotina/rotina.sh'&

echo "<<<<<<<<<<    INICIO DA ROTINA - CLOUDSTACK - CSGL     >>>>>>>>>>" >> $logrotina
date >> $logrotina
c=1

#Início da rotina
echo "#####CRIA NOVA INSTANCIA#####"  >> $logrotina
if [ $ofertaservico -eq 1 ]; then
      while (( $c <= $sessoes ))
      do
            echo "Oferta de serviço pequena!"  >> $logrotina
            echo "Oferta de serviço pequena!"
            #Comando para criação da VM – todas as variáveis devem ser setadas baseadas nos códigos
obtidos no ambiente em produção.
            cloudmonkey deploy virtualmachine hostid=b2f37f01-3ad2-4471-9c3b-ae6462f5cb0b
zoneid=2c1fbf53-cb95-4d72-96d3-beb5c7bc73cb serviceofferingid=a3ebc4f9-9431-4ebe-8245-
8a72602609e7 templateid=1ff6c20e-62c0-11e5-82bb-6cf049f92951  name=InstTesteDeployP-$c-
$RAND displayname=TesteDeployP-$c-$RAND >> $logrotina &

            echo "INSTANCIA ($c) PEQUENA CRIADA!"
            sleep 2
            #Comando necessário para registrar o ID da instância criada
            vmid[$c]=$(cloudmonkey list virtualmachines name=TesteDeployP-$c-$RAND filter=id
|grep "id ="|cut -f3 -d' ' |head -n 1)

            echo "######ID da VM registrada no script $c: ${vmid[$c]}"  >> $logrotina
            echo "######ID da VM registrada no script $c: ${vmid[$c]}"
            c=$(( c+1 ))
      done

elif [ $ofertaservico -eq 2 ]; then
      while (( $c <= $sessoes ))
      do
            echo "Oferta de serviço média!"  >> $logrotina
            echo "Oferta de serviço média!"
            cloudmonkey deploy virtualmachine hostid=b2f37f01-3ad2-4471-9c3b-ae6462f5cb0b
zoneid=2c1fbf53-cb95-4d72-96d3-beb5c7bc73cb serviceofferingid=babe83fb-8b38-4364-a3b9-
17b003897ec4 templateid=1ff6c20e-62c0-11e5-82bb-6cf049f92951  name=InstTesteDeployM-$c-
$RAND displayname=TesteDeployM-$c-$RAND >> $logrotina &

            echo "INSTANCIA ($c) MÉDIA CRIADA!"
            sleep 2
```

```
        vmid[$c]=$(cloudmonkey list virtualmachines name=TesteDeployM-$c-$RAND filter=id
|grep "id ="|cut -f3 -d' ' |head -n 1)

        echo "######ID da VM registrada no script $c: ${vmid[$c]}"  >> $logrotina
        echo "######ID da VM registrada no script $c: ${vmid[$c]}"

        c=$(( c+1 ))
    done

elif [ $ofertaservico -eq 3 ]; then
    while (( $c <= $sessoes ))
    do
        echo "Oferta de serviço grande!"  >> $logrotina
        echo "Oferta de serviço grande!"
        cloudmonkey deploy virtualmachine hostid=b2f37f01-3ad2-4471-9c3b-ae6462f5cb0b
zoneid=2c1fbf53-cb95-4d72-96d3-beb5c7bc73cb serviceofferingid=fc55be0a-4e1c-4e80-9063-
290f9cf2f7ad templateid=1ff6c20e-62c0-11e5-82bb-6cf049f92951  name=InstTesteDeployG-$c-
$RAND displayname=TesteDeployG-$c-$RAND >> $logrotina &

        echo "INSTANCIA ($c) GRANDE CRIADA!"
        sleep 2

        vmid[$c]=$(cloudmonkey list virtualmachines name=TesteDeployG-$c-$RAND filter=id
|grep "id ="|cut -f3 -d' ' |head -n 1)

        echo "######ID da VM registrada no script $c: ${vmid[$c]}"  >> $logrotina
        echo "######ID da VM registrada no script $c: ${vmid[$c]}"

        c=$(( c+1 ))

    done

else
    exit
fi


c=1
while (( $c <= $sessoes ))
do
    echo "#####CRIAR SNAPSHOT $c VM#####"  >> $logrotina
    cloudmonkey create vmsnapshot name=testesnap-$c snapshotmemory=yes
virtualmachineid=${vmid[$c]}& >> $logrotina &
    echo "SNAPSHOT $c CRIADO!"
    c=$(( c+1 ))


done
sleep 2
c=1
while (( $c <= $sessoes ))
```

```
do
	echo "#####APAGAR INSTANCIAi $c#####"  >> $logrotina
	cloudmonkey destroy virtualmachine id=${vmid[$c]} >> $logrotina
	cloudmonkey expunge virtualmachine id=${vmid[$c]} >> $logrotina
	echo "INSTANCIA $c APAGADA!"
	c=$(( c+1 ))

done

#Encerra captação de dados no alvo
ssh 192.168.30.1 'killall sadc >& /dev/null'&

echo "<<<<<<<<<<    FIM DA ROTINA - CLOUDSTACK - CSGL     >>>>>>>>>>" >> $logrotina
NOWF=$(date +"%m-%d-%Y-%T")
datafinal=`date +%s`
soma=`expr $datafinal - $datainicial`
resultado=`expr 10800 + $soma`
tempo=`date -d @$resultado +%H:%M:%S`
echo "FIM AS $NOWF"
echo "TEMPO TOTAL GASTO: $tempo "
```

APÊNDICE B – Comandos utilizados no CloudMonkey

Comando	Descrição
cloudmonkey deploy virtualmachine	Cria uma nova instância, inicializando-a em seguida. Possui parâmetros como identificação da oferta de serviço, *hypervisor*, entre outros.
cloudmonkey destroy virtualmachine	Remove instância. Possui parâmetro de identificação da instância.
cloudmonkey expunge virtualmachine	Limpa todos os registros da instância removida. Possui parâmetro de identificação da instância.
cloudmonkey create vmsnapshot	Cria uma cópia do estado atual da instância. Possui parâmetros como descrição do snapshot, nome do snapshot, entre outros.