UNIVERSIDADE FEDERAL DO PARÁ
CAMPUS UNIVERSITÁRIO DE MARABÁ
FACULDADE DE COMPUTAÇÃO
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Aline Santana Oliveira
Príscila Alves Lima

INTEGRAÇÃO DO FRAMEWORK DE MONITORAMENTO ZABBIX COM UMA NUVEM PRIVADA

Marabá
2013

UNIVERSIDADE FEDERAL DO PARÁ
CAMPUS UNIVERSITÁRIO DE MARABÁ
FACULDADE DE COMPUTAÇÃO
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Aline Santana Oliveira
Príscila Alves Lima

INTEGRAÇÃO DO FRAMEWORK DE MONITORAMENTO ZABBIX COM UMA NUVEM PRIVADA

Trabalho de Conclusão de Curso, apresentado para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Prof. Msc. Warley Muricy Valente Junior.

Marabá-PA
2013

UNIVERSIDADE FEDERAL DO PARÁ
CAMPUS UNIVERSITÁRIO DE MARABÁ
FACULDADE DE COMPUTAÇÃO
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Aline Santana Oliveira
Príscila Alves Lima

INTEGRAÇÃO DO FRAMEWORK DE MONITORAMENTO ZABBIX COM UMA NUVEM PRIVADA

Trabalho de Conclusão de Curso, apresentado para obtenção do grau de Bacharel em Sistemas de Informação.

Data da defesa: 12 de Agosto de 2013.
Conceito: Excelente

Banca Examinadora

_____
Prof. Msc. Warley Muricy Valente Junior
Faculdade de Computação/UFPA - Orientador

_____
Prof. Esp. Rangel Filho Teixeira
Faculdade de Computação/UFPA - Membro

_____
Dir. Esp. Sebastião de Sousa Mesquita
Processamento de Dados do Estado do Pará/PRODEPA Marabá - Membro

DEDICATÓRIA

Primeiramente a Deus, pois sem Ele não somos nada, e graças a Fé que tenho Nele, fui capaz de seguir em frente com os meus objetivos. Aos meus amados pais, Adão e Alaide, pelas orações por mim e conselhos, eles são minha rocha firme por onde posso andar. Aos meus irmãos Athos e Wilame pela paciência e carinho e ao meu noivo por sempre me dar o apoio que preciso.

Aline Santana Oliveira

À Deus por tudo quanto tem realizado em minha vida, a Ele toda honra, glória e louvor. A minha querida mãe Merian Alves Lima, por seu amor, dedicação, sempre estando ao meu lado me orientando e encorajando a lutar pelos meus sonhos. E aos meus irmãos Rafael Alves Lima e Sara Alves Lima pelo carinho e força.

Príscila Alves Lima

AGRADECIMENTOS

Á Universidade Federal do Pará.

Ao professor Rangel Teixeira, coordenador do Curso de Sistemas de Informação, pela longa empreitada no decorrer do trabalho.

Ao professor Warley Muricy Valente Junior pela orientação competente, dedicando-se de forma marcante na elaboração deste trabalho.

E aos demais profissionais da FACOM que modelaram nosso conhecimento durante nossa jornada acadêmica.

EPÍGRAFE

"Deleita-te também no Senhor, e Ele te concederá o que deseja o teu coração. Entrega o teu caminho ao Senhor; confia Nele e Ele tudo fará."

Salmos *37. 4-5*

SUMÁRIO

LISTA DE QUADROS

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

CC	*Cluster Controller* ou Controlador de Cluster
CLC	*Cloud Controller* ou Controlador da Nuvem
NC	*Node Controller* ou Controlador de Nó
NIST	*National Institute of Standards and Technology* ou Instituto Nacional de Padrões e Tecnologia
SLA	*Service Level Agreement* ou Acordo de Nível de Serviço
VM	*Virtual Machine* ou Maquina Virtual
CPU	*Central Processing Unit* ou Unidade Central de Processamento
CN	Computação em nuvem
SaaS	*Software as a Service* ou Software como Serviço
PaaS	*Platform as a Service* ou Plataforma como Serviço
IaaS	*Infrastructure as a Service* ou Infraestrutura como Serviço
VMM	*Virtual Machine Monitor* ou Monitor de Maquina Virtual
KVM	*Kernel Based Virtual Machine* ou Maquina Virtual Baseada em Núcleo
BSA	*Business Software Alliance* ou Aliança Empresarial de Software
Brasscom	Associação Brasileira das Empresas de TICs
PNBL	Plano Nacional de Banda Larga
KPIs	*Key Performance Indicator* ou Chave de Controle de Desempenho
QoS	*Quality of Services* ou Qualidade de Serviço
TCP/IP	*Transmission Control Protocol/ Internet Protocol* ou Protocolo de Internet
SNMP	*Simple Network Management Protocol* ou Protocolo simples de gerenciamento de rede
PHP	*Personal Home Page* ou Processador de Hipertexto
LAN	*Local Area Network* ou Rede Local
WAN	*Wide Area Network* ou Ampla Área de Trabalho
PCMONS	*Private Cloud Monitoring System* ou Sistema de Monitoramento para Nuvem Privada
CA	Computação Autônoma
API	*Application Programming Interface* ou Interface de Programação de Aplicativos

RESUMO

O monitoramento do ambiente de nuvem privada possui a finalidade de garantir que os componentes da mesma tenham um bom funcionamento, visto que este ambiente está em constante alteração, pois a percepção que o usuário possui com relação à nuvem, é que ela é um aglomerado de informações, dessa forma ela também se torna propícia a ataques e a instabilidades. Realizou-se então o estudo da arte sobre computação em nuvem privada a fim de entender os conceitos e aplicações da mesma, além das funcionalidades e de seus principais modelos de implantação e de serviço. Investigou-se o *Framework* Zabbix, pois se trata de uma ferramenta de monitoramento robusta, porém de fácil manipulação, que possui uma interface simples e concisa. Dessa maneira, o presente trabalho realizou o monitoramento de uma nuvem privada utilizando o *framework* de monitoramento Zabbix. A fim de analisar o comportamento de uma nuvem privada, efetuando troca de serviços e dados, foi possível coletar informações como o Ping e SSH das MV's (Máquinas Virtuais) na nuvem e assim verificar a disponibilidade das mesmas na rede, além disso, foram monitoradas outras métricas nas maquinas físicas como CPU, memória e processos ativos. Foram feitos testes de sobrecarga nas MVs a fim de avaliar o comportamento das maquinas físicas em um ambiente de várias requisições. Como resultados obtidos, percebeu-se que além das métricas coletadas pelo Zabbix, foi possível observar a capacidade que a nuvem privada possui de se adaptar em meio à heterogeneidade das tecnologias e também o quanto a mesma é segura com relação as suas informações.

Palavras – chaves: Computação em Nuvem, Nuvem Privada, Monitoramento.

ABSTRACT

The monitoring of private cloud environment has the purpose of ensuring that the same components have a smooth operation, since this environment is constantly changing as the perception that the user has with respect to the cloud is that it is a cluster of information, so it also becomes prone to attacks and instabilities. The next step was the study of the art private cloud computing in order to understand the concepts and applications of it, beyond its core functionality and deployment models and service. Investigated the Framework Zabbix because it is a robust monitoring tool, but easy to handle, it has a simple interface and concise. Thus, this paper carried out the monitoring of a private cloud using the framework Zabbix monitoring. In order to analyze the behavior of a private cloud, making exchange of services and data, it was possible to collect only information such as Ping and SSH of VM's (Virtual Machines) in the cloud and thus verify the availability of the same network, in addition, were monitored physical machines in other metrics such as CPU, memory, and active processes. Tests were made on overload of VMs in order to assess the behavior of the physical machines in an environment of multiple requests. As results, it was noticed that besides the metrics collected by Zabbix, it was possible to observe the ability of the private cloud has to fit through the heterogeneity of technologies and also how much it is safe with respect to your information.

Key - Words: Cloud Computing, Private Cloud, Monitoring.

1 INTRODUÇÃO

Recentemente surgiu um novo paradigma para distribuição de serviços de computação, trata-se de um modelo semelhante à distribuição de serviços básicos como o fornecimento de eletricidade, água, telefone, etc., onde o provisionamento desses serviços utiliza o modelo de pagamento baseado no uso, e toda sua infraestrutura entrega os serviços de forma transparente ao usuário. A esse novo paradigma se convencionou chamar de *Cloud Computing* ou Computação em Nuvem. De acordo com Velte *et al* (2010), o termo é uma metáfora a forma como a Internet é comumente representada nos diagramas de rede (como uma nuvem), abstraindo a infraestrutura e sua complexidade.

Segundo Vaquero *et al* (2009), a computação em nuvem é um conjunto de recursos virtualizados facilmente utilizáveis e acessíveis, tais como hardware, software, plataformas de desenvolvimento e serviços. Estes recursos podem ser dinamicamente reconfigurados para se ajustarem a uma carga de trabalho variável, permitindo a otimização do seu uso. Esse conjunto de recurso é explorado através de um modelo pague-pelo-uso, com garantias oferecidas pelo provedor através de Acordos de Nível de Serviços (*Service Level Agreement* - SLA).

O NIST (2011) (*National Institute of Standards and Technology* - Instituto Nacional de Padrões e Tecnologia), afirma que para um modelo de computação ser considerado uma nuvem, o mesmo deve possuir as seguintes características essenciais: *Self-Service*[1] sob demanda, amplo acesso, *pool* [2]de recursos, elasticidade rápida e serviço medido.

Ainda segundo o NIST (2011), a computação em nuvem pode ser classificada de acordo com as características do serviço disponibilizado. Para isso consideram-se os recursos que o usuário tem acesso, estes são classificados em: Plataforma como Serviço *(Platform as a Service* - Paas*)*, Software como Serviço (*Software as a Services* - SaaS) e Infraestrutura como Serviço (*Infrastructure as a Service* - IaaS). Além da classificação por modelo de distribuição de serviços, a computação em nuvem é categorizada de acordo com o modelo de implantação, que são: Nuvem Pública, Nuvem Privada, Nuvem Híbrida e a Nuvem Comunitária.

[1] Termo usado para designar a capacidade que o usuário tem para provisionar recursos computacionais sem a interação humana com o provedor de serviço.
[2] Conjunto de recursos computacionais.

Este novo modelo de distribuição de serviços está se tornando atraente para as organizações uma vez que ele possibilita diversos benefícios para os seus negócios, especialmente para organizações que não dispõe de um alto poder aquisitivo. Segundo Taurion (2009) esses benefícios consistem em: Eliminar a necessidade de adquirir recursos antecipadamente; Criar uma ilusão de disponibilidade de recursos infinitos, acessáveis sob demanda; Oferecer elasticidade permitindo que as empresas utilizem os recursos na quantidade que forem necessários, aumentando e diminuindo a capacidade de processamento de forma dinâmica; Oferecer serviços em nuvem e que o pagamento seja efetuado conforme a quantidade de recursos utilizados.

Esse paradigma computacional vem com a intenção de melhorar a infraestrutura de TI (Tecnologia da Informação), pois os hardwares ficaram obsoletos rapidamente, e a nuvem fará com que o computador, se torne somente um dispositivo que ligue o usuário a web. Não existirá mais a preocupação de fazer *backups*, os próprios servidores de nuvem serão responsáveis por isso.

De acordo com Gonçalves (2011), este assunto é relativamente novo, porém alguns recursos oferecidos baseados nesse novo paradigma de distribuição de serviços já se encontram disponíveis aos usuários, embora muitas vezes estes não tenham conhecimento dos fatos, a exemplo pode-se citar o *Google Docs, Google Drive, DropBox, Sky Drive,* entre outros.

A implementação de uma nuvem privada no âmbito de uma empresa pode agregar diversos benefícios à mesma, como: melhorias no aproveitamento dos recursos, redução dos custos com manutenção, redução do consumo de energia, permitir maior controle das configurações da nuvem; possibilitar que o gerenciamento se adeque as necessidades da organização, além de permitir que ela adquira experiência sobre o funcionamento de uma nuvem.

De acordo com Canedo (2012) a computação em nuvem, pela sua característica de agrupar diversas tecnologias e não se tratar apenas de um paradigma computacional, mas também de um modelo de negócios, requer a necessidade de atividades gerenciais de mais alto nível como o monitoramento de seu adequado funcionamento.

Desta forma, a realização do monitoramento dos aspectos técnicos da nuvem é válida para garantir o bom funcionamento da mesma, uma vez, que essa prática objetiva averiguar o

desempenho do sistema, diagnosticar um potencial problema e prover informações precisas para a realização da manutenção.

Quanto a uma nuvem privada, a qual é utilizada exclusivamente por uma organização ou empresa que armazena informações e documentos sigilosos na mesma, o monitoramento exerce um papel fundamental no acompanhamento deste modelo de nuvem, não apenas para garantir que a nuvem tenha um bom funcionamento, como também para servir de base para atividades de planejamento, melhorias e segurança.

Considerando a importância de uma nuvem privada e a necessidade de seu monitoramento, o objetivo principal deste trabalho é realizar a implantação, configuração e adaptação do *framework* Zabbix no monitoramento de uma nuvem privada. Além deste objetivo, este trabalho também visa:

- Investigar o estado da arte da computação em nuvem privada e soluções livres para este modelo de implantação;

- Definir e modelar a topologia da rede local, onde será realizado o estudo de caso;

- Implantar, com caráter experimental (acadêmico), uma nuvem privada na sala de pesquisa da FACOM;

- Realizar a instalação e configuração de software de infraestrutura para nuvem privada;

- Realizar a instalação, configuração e adaptação do *framework Zabbix* para o monitoramento;

- Avaliar o comportamento, desempenho, facilidade de uso e disponibilidade do *framework* dentro de uma nuvem privada através de métricas objetivas tanto para máquinas virtuais, como para máquinas reais.

Como resultado deste projeto, a comunidade científica e até mesmo empresas de médio e pequeno porte, terão como referência uma arquitetura de nuvem privada de baixo custo, escalável, e com *framework* de monitoramento adaptável as regras de negócio e ao administrador local.

Quanto à estrutura deste trabalho o mesmo está organizado da seguinte forma:

Capítulo 1 - Este é o atual capítulo que apresenta a temática, a motivação, e os principais objetivos para a realização deste trabalho.

Capítulo 2 - Apresenta os conceitos gerais e classificações da Computação em Nuvem.

Capítulo 3 - Aborda o assunto de monitoramento, tanto para ambientes em grade como também para ambiente em nuvem.

Capítulo 4 - Apresenta os trabalhos correlatos encontrados na literatura que serviram de base para a fundamentação deste projeto.

Capítulo 5 - Neste capítulo serão apresentados os métodos e as ferramentas utilizadas para a implementação do presente trabalho.

Capítulo 6 - Trata do estudo de caso para a realização do projeto. Este capítulo tem por objetivo detalhar os procedimentos realizados para validar a proposta.

Capítulo 7 - Apresenta a conclusão que se tem a cerca do trabalho, as dificuldades enfrentadas e sugere atividades para trabalhos futuros.

2 COMPUTAÇÃO EM NUVEM

Neste capítulo serão apresentados os conceitos gerais da Computação em Nuvem (CN), suas características essenciais, assim como suas classificações quanto ao modelo de distribuição de serviços e os modelos de implantação, além de abordar a tecnologia de virtualização e o futuro da computação em nuvem no Brasil.

2.1 Definição

De acordo com Uriarte (2012) a computação em nuvem não tem um início bem definido. Segundo ele em 1961, John MacCarthy sugeriu que o poder da computação e aplicações específicas poderiam ser entregues utilizando o modelo de fornecimento de serviços básicos. A partir de então, diversas tecnologias e paradigmas foram desenvolvidos, mais apenas com a popularização da Internet e da banda larga, em meados dos anos 90, a idéia começou a se desenvolver.

Segundo Chaves (2010) diversos trabalhos de pesquisa viabilizaram o que se convencionou chamar de computação em nuvem, entre eles pode-se citar a computação utilitária, computação em grade, computação em cluster, virtualização e a computação distribuída de modo geral.

Várias definições têm sido propostas pelo meio acadêmico e pela indústria para computação em nuvem ao longo do amadurecimento do assunto. Também há organizações que trabalham no desenvolvimento de pesquisas sobre o tema no sentido de melhor defini-la, a exemplo pode-se citar o NIST (National Institute of Standards and Technology - Instituto Nacional de Padrões e Tecnologia), do governo estadunidense, que conceitua a computação em nuvem da seguinte forma:

> Computação em nuvem é um modelo que permiti o acesso, de modo conveniente e sob demanda, a um pool compartilhado de recursos computacionais configuráveis (por exemplo, redes, servidores, armazenamento, aplicativos e serviços) que podem ser rapidamente provisionados e liberados com mínimo esforço de gerenciamento ou interação com o provedor de serviços (MELL; GRANCE, 2011, p. 2).

Para Canedo (2012) a computação em nuvem faz referência ao uso de diversas aplicações por meio da Internet com a mesma facilidade de tê-las instaladas no computador dos usuários. “Uma definição mais simples do termo seria o provisionamento de recursos

computacionais para um cliente a partir de uma demanda. O fornecedor desses recursos abstrai as tecnologias envolvidas e a procedência do recurso para os usuários finais" (VITTI, 2012, p.5).

Ainda segundo Vitti (2012) a computação em nuvem vem despertando interesse em grandes empresas que passaram a investir em pesquisas sobre o tema na tentativa de conquistar espaço e tornar-se referência no mercado de CN, a exemplo pode-se citar a *Google, IBM, Amazon, Microsoft* e *Apple*. O objetivo da computação em nuvem é fornecer serviços sob demanda, desta forma os usuários não precisam realizar grandes investimentos em hardware, uma vez que grande parte do poder de processamento encontra-se na nuvem e não no computador do usuário. A Figura 1 ilustra de modo simples a arquitetura genérica da computação em nuvem.

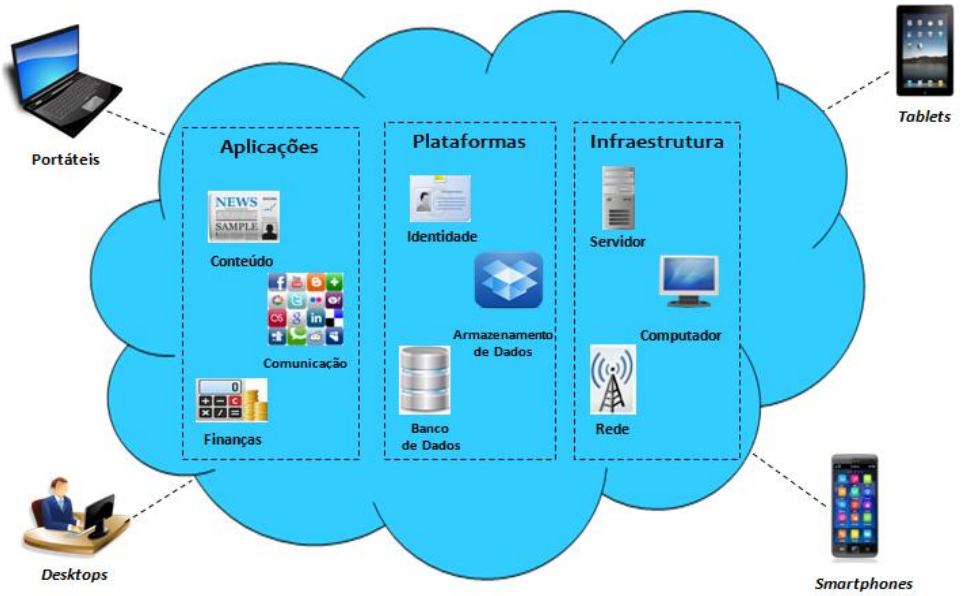

Figura 1- Arquitetura da Computação em Nuvem.
Fonte: Lima, 2011.

Conforme apresentado na figura acima os serviços disponibilizados na nuvem, são: aplicações, plataformas e infraestrutura. Os usuários, por sua vez, precisam apenas de um dispositivo conectado a Internet para ter acesso a esses serviços, vale ressaltar que os serviços da nuvem são fornecidos ao usuário sob demanda, ou seja, o mesmo paga apenas pelo que consome.

2.2 Características Essenciais

De acordo com o NIST (2011), para que um modelo de computação seja considerado uma nuvem o mesmo deve possuir as seguintes características essenciais:

- **Autoatendimento sob demanda** – O modelo deve permitir que o usuário possa prover funcionalidades computacionais (tais como tempo de servidor, capacidade de armazenamento, entre outros) automaticamente, sem interação com o provedor de serviço.
- **Amplo acesso a serviços de rede** – A nuvem deve prover mecanismos para que os serviços disponibilizados sejam acessíveis por vários dispositivos móveis e portáteis.
- ***Pool* de recursos** – de acordo com Veras (2012) são recursos computacionais disponibilizados por um provedor que é utilizado para servir a múltiplos usuários, onde os recursos podem ser alocados e realocados dinamicamente conforme a demanda.
- **Elasticidade Rápida** – Os recursos computacionais devem ser rapidamente provisionados, assim como liberados. Passando a impressão ao usuário de que os recursos são ilimitados.
- **Serviços mensuráveis** – Os sistemas de gerenciamento usados devem controlar e monitorar automaticamente os serviços (processamento, armazenamento, contas de usuário, etc.) disponibilizados ao usuário. E esse controle deve ser transparente tanto para o provedor de serviços como para o consumidor.

Além das características mencionadas, Sousa, Moreira e Machado (2009) afirmam que a CN foi desenvolvida com o objetivo de oferecer serviços de fácil acesso, baixo custo e garantir características tais como disponibilidade e escalabilidade. Segundo eles esse modelo visa fornecer basicamente três benefícios: redução de custo na aquisição e composição da infraestrutura para atender as necessidades das empresas; Oferecer flexibilidade na adição e troca de recursos computacionais, permitindo escalar tanto em nível de recurso de hardware como de software para atender as demandas dos consumidores; E por último, mais não menos importante, prover abstração e facilitar o acesso aos serviços, uma vez que os usuários não precisam saber dos aspectos de localização física e de entrega dos resultados destes serviços.

2.3 Classificações para Computação em Nuvem

Ainda segundo o NIST (2011) além das cinco características essenciais que uma nuvem deve possuir a mesma também é composta por três modelos de serviços e quatro modelos de implementação, que serão abordados nas próximas seções.

2.3.1 Modelo de Serviços

Nesta classificação são considerados as funções e os recursos que o usuário tem acesso. Existem três modelos de serviços para computação em nuvem listados abaixo, e em seguida é apresentada a Figura 2 que ilustra os modelos de serviços e seus respectivos consumidores.

- **Software como Serviço (SaaS):** Segundo Veras (2012) esse modelo refere-se aos aplicativos disponibilizados pelo provedor como serviços e acessados pelos usuários através de um *browser*. O público alvo deste serviço são usuários finais, que poderão ter acesso aos softwares sem se preocuparem com a aquisição de licenças. Como exemplo de SaaS temos o *SalesForc*e ou um editor de texto como o *Google Docs.*
- **Plataforma como Serviço (PaaS):** De acordo com Uriarte (2012) é o provisionamento de toda uma plataforma para desenvolvimento de aplicativos, abstraindo dos desenvolvedores os requisitos de hardware e possivelmente de outras camadas de software necessárias como banco de dados, servidor web e o suporte a linguagem de programação agilizando e reduzindo a complexidade do desenvolvimento. Como exemplos desse tipo de serviços pode-se citar a *AppEngine* do Google e o *Windows Azure* da Microsoft, onde o público alvo deste serviço são desenvolvedores de aplicações.
- **Infraestrutura como Serviço (IaaS):** Os serviços oferecidos pelo provedor nesse modelo é a infraestrutura de processamento e armazenamento necessária para o PaaS e o SaaS. Segundo Sousa, Moreira e Machado (2009) o IaaS é responsável por prover toda a infraestrutura para a PaaS e o SaaS, onde seu principal objetivo é viabilizar o fornecimento de recursos, tais como servidores, rede, armazenamento e outros recursos de computação fundamentais para construir um ambiente de aplicação sob demanda, que podem incluir sistemas operacionais e aplicativos. Como exemplo de IaaS temos a Amazon EC2 e a Gogrid. Os usuários deste serviço, de acordo com Uriarte (2012), são principalmente administradores de rede, engenheiros de rede e administradores de setores de tecnologia.

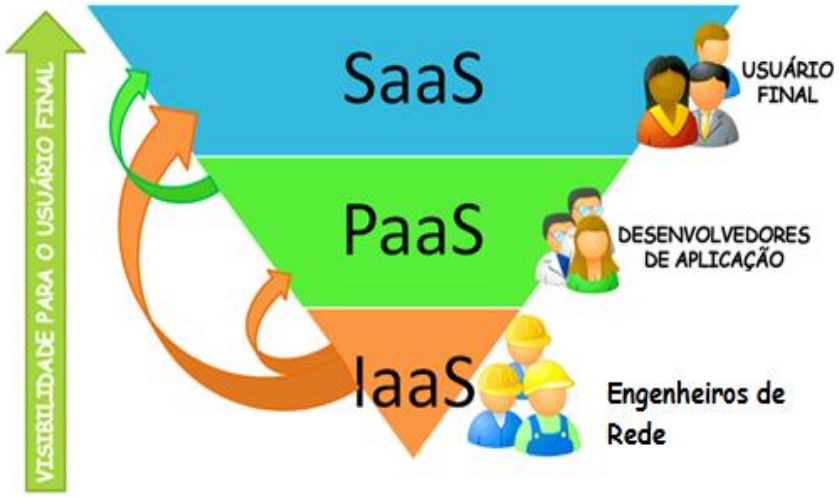

Figura 2 - Modelo de Serviços e seus respectivos usuários
Fonte: Uriarte, 2012.

2.4 Modelo de Implantação

Na categorização conforme o modelo de implantação é considerado o público-alvo e a abrangência da nuvem, que podem ser classificadas, conforme a Figura 3, em pública, privada, comunitária e híbrida:

- **Pública** – Segundo Mell e Grance (2011) nuvem pública é a infraestrutura disponibilizada para o público em geral, podendo ser acessada por qualquer usuário. Neste caso a nuvem está hospedada no datacenter[3] do provedor e este disponibiliza os serviços aos usuários de modo geral.
- **Privada –** Mell e Grance (2011) afirmam que é uma infraestrutura de nuvem usada exclusivamente por uma organização, sendo uma nuvem local ou remota gerenciada pela própria empresa ou por terceiros, onde a nuvem é operada exclusivamente pelo cliente e todos os seus dados estão localizados no seu próprio datacenter.
- **Comunitária -** Uriarte (2012) afirma que este modelo é baseado no compartilhamento de uma nuvem por diversas organizações (universidades ou indústrias), utilizada por uma comunidade específica que possui interesses semelhantes no que diz respeito à segurança, tais como missão e área de atuação.
- **Híbrida –** Para Mell e Grance (2011), neste modelo existe uma composição de duas ou mais nuvens (privada, comunitária ou pública) que permanecem como entidades

[3] Conjunto integrado de componentes de alta tecnologia que permite fornecer serviços de infraestrutura de TI (Tecnologia da Informação) de valor agregado, tipicamente processamento e armazenamento de dados, em larga escala, para qualquer tipo de organização.

únicas, ligadas por uma tecnologia padronizada ou proprietária que permite a portabilidade de dados e aplicações. Os diferentes tipos de nuvem envolvidos neste modelo podem estar hospedados tanto na infraestrutura do provedor quanto do cliente.

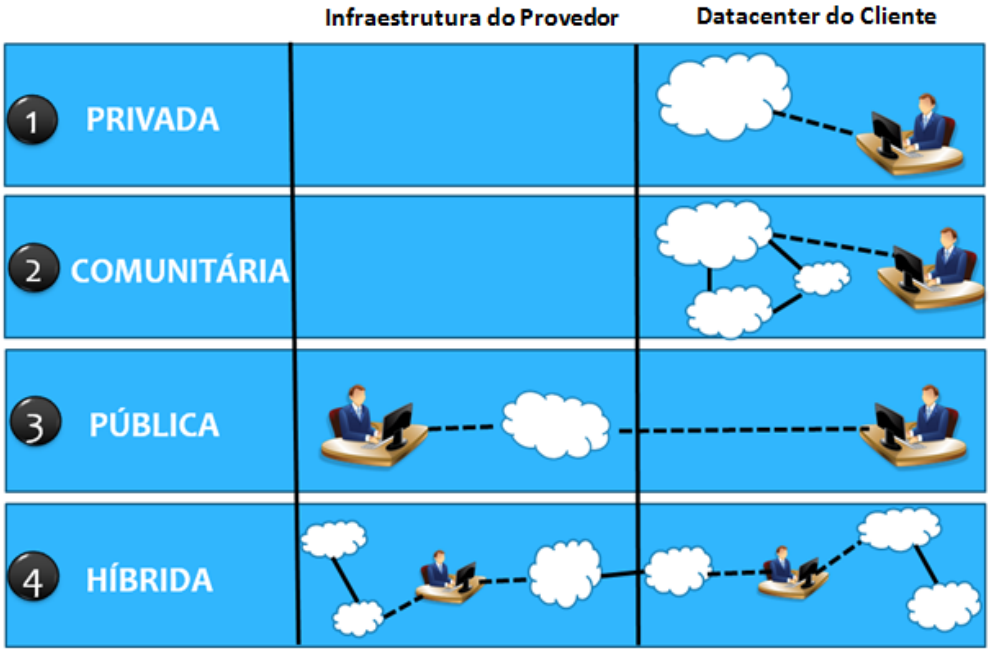

Figura 3 - Modelo de Implantação da Nuvem.
Fonte: Uriarte (2012 apud Cloud Security Alliance, 2009).

2.5 Virtualização

De acordo com Veras (2011) a virtualização é o componente central da computação em nuvem, na medida em que possibilita aprimorar o uso dos recursos e viabilizar o modelo de computação sob demanda. Uriarte (2012, p. 34) define a virtualização como:

> Um processo que abstrai logicamente o sistema operacional do hardware criando uma nova camada de software entre os mesmos, a qual fica responsável pela comunicação com o hardware e permite a execução simultânea de vários sistemas operacionais (chamados de Máquinas Virtuais ou MV's).

Ainda segundo Uriarte (2012) a camada de software criada entre o hardware e o sistema operacional é chamada de monitor de máquinas virtuais, também denominada por alguns autores como "sistema operacional para sistemas operacionais" ou ainda de hipervisor. "O hipervisor ou monitor de máquina virtual é quem permite que seja executado o sistema operacional hóspede, construindo as interfaces virtuais para as interfaces reais do sistema hospedeiro" (CHAVES; URIARTE; WESTPHALL, p 34, 2010).

De acordo com Carissimi (2008) existe duas técnicas para a implementação de máquinas virtuais de sistemas ou Monitores de Máquinas Virtuais (VMM – *Virtual Machine Monitor*) que são a Virtualização Total (completa) e a Paravirtualização, definidas por ele da seguinte forma:

- **Virtualização Total –** Técnica que prover uma réplica (virtual) do hardware subjacente de modo que o sistema operacional e as aplicações podem executar como se estivesse executando diretamente sobre o hardware original. A grande vantagem dessa técnica é que o sistema operacional hóspede não precisa ser modificado para executar sobre a VMM.
- **Paravirtualização –** Ainda segundo Carissimi (2008) trata-se de uma abordagem onde o sistema hóspede é modificado para chamar a VMM no momento em que for executado uma instrução. A Paravirtualização é uma "técnica de virtualização que permite que o sistema operacional esteja ciente de que está sendo executado sob um hipervisor e não diretamente sob o hardware" (Xen.org, 2010).

Um exemplo de hipervisor pode-se citar o KVM *(Kernel-based Virtual Machine* – Maquina Virtual Baseada em Núcleo) que de acordo com Vitti (2012) é uma solução de virtualização completa para Linux em hardware x86[4] contendo extensões de virtualização (Intel VT ou AMD-V). É composto por um módulo de *kernel*[5] carregável, que fornece a infraestrutura de virtualização do núcleo e um módulo de processador específico. O uso deste hipervisor possibilita executar diversas máquinas virtuais rodando Linux ou Windows sem a necessidade de modificação do *kernel*.

Além do KVM pode-se citar também o Hipervisor Xen que segundo seu fabricante é "uma camada de software que roda diretamente no hardware do computador, substituindo o sistema operacional e assim possibilitando que o mesmo hardware rode diferentes sistemas operacionais convidados de forma concorrente" (Xen.org, p 1, 2010).

Chaves (2010), afirma que a tecnologia de máquinas virtuais ganha destaque no ambiente de computação em nuvem, não só por possibilitar a melhor utilização de recursos pela disponibilidade de novas máquinas a partir de um mesmo hardware, como também pela maior facilidade para implantação de software através do uso de imagens virtuais, essencial para se provisionar serviços sob demanda de modo rápido.

[4] Nome dado a arquitetura de processadores desenvolvidos pela empresa Intel Corporation .
[5] É o núcleo do sistema operacional, é ele que permite a comunicação entre o software e o hardware.

2.6 O Futuro da Computação em Nuvem no Brasil

De acordo com Sobragi (2012) a computação em nuvem é considerada um paradigma computacional da atualidade, e sua adoção vêm crescendo nos últimos anos. A NMC (*New Media Consortium* – Consórcio de Novas Mídias) em sua primeira edição brasileira, publicada no fim do ano de 2012, mostra quais tecnologias irão tornar-se populares e a computação em nuvem é uma delas. No Brasil estima-se que no prazo de um ano ou menos ela irá tornar-se muito importante para os usuários de celulares e *tablets*. A NMC acredita que talvez o Brasil seja um dos maiores países no mundo onde esta tecnologia esteja mais forte.

Segundo a revista convergência Digital (2013), estudos realizados pela BSA (*Business Software Alliance* – Aliança Empresarial de Software) no ano de 2013, verificou-se que quanto ao uso da computação em nuvem o Brasil subiu no ranking e está no 22º lugar, sendo que no estudo anterior o mesmo estava em último lugar. Toda essa mudança ocorreu por causa da legislação contra os crimes cibernéticos e ao progresso no Plano Nacional de Banda Larga (PNBL). O Brasil possui um grande potencial para se tornar um centro utilizador de nuvem, onde segundo a Brasscom (Associação Brasileira das Empresas de TICs) a perspectiva é que a nuvem no Brasil irá movimentar cerca de R$ 81 bilhões ate 2017.

Com relação aos empregos na área de computação em nuvem, a revista Convergência Digital (2013) afirma que em 2015 irão surgir 798 mil vagas de emprego no Brasil. Dessa forma algumas áreas da computação estarão em ascensão, entre elas podemos citar: analistas de informação, arquitetos de software e os especialistas em segurança. As que entrarão em declínio serão: especialistas em infraestrutura e administradores de sistemas.

2.7 Resumo do Capítulo

Este capítulo apresentou os conceitos da computação em nuvem, suas principais características e definições. Abordou também as classificações para computação em nuvem que ocorre de acordo com a distribuição de serviço (IaaS, PaaS e SaaS), e conforme o modelo de implantação (privada, pública, híbrida e comunitária), bem como o futuro da computação em nuvem no Brasil, os empregos que ela irá possibilitar, e as perspectivas de crescimento do uso desse novo paradigma em dispositivos móveis.

No próximo capítulo iremos discorrer sobre o monitoramento em CN, e algumas ferramentas para monitoramento de rede existente no mercado, além de apresentar o *framework* Zabbix que será utilizado no estudo de caso deste trabalho.

3 MONITORAMENTO EM COMPUTAÇÃO EM NUVEM

Este capítulo abordará alguns conceitos sobre monitoramento de maneira geral, e no ambiente de computação em nuvem, além de enfatizar sua relevância em um ambiente heterogêneo característico da nuvem. Em seguida será apresentada a arquitetura de monitoramento e suas camadas. Por fim serão exibidas algumas ferramentas existentes na literatura que são utilizadas no monitoramento de redes de computadores.

3.1 Definição, Como e o Que Monitorar

De acordo com Uriarte (2012, p. 57) “monitoramento é a ação de coletar informações relacionadas com as características e o status dos recursos que são de interesse”. Para Ferreira (2011) o monitoramento diz respeito a todas as atividades, métodos, procedimentos e ferramentas utilizadas na operação, administração, manutenção e controle de um sistema de rede. Segundo ele existem diversas razões que justificam a necessidade do monitoramento como: diagnosticar problemas rapidamente; garantir que os sistemas de segurança estejam funcionando corretamente; Gerar informações do estado da rede; Planejar mudanças a ser implantadas na rede, etc.

Para Rhoden (2002), por menor e mais simples que seja uma rede de computadores, ela precisa ser gerenciada, a fim de garantir, aos seus usuários, a disponibilidade de serviços a um nível de desempenho aceitável. Isso aumenta a complexidade de seu gerenciamento, forçando a adoção de ferramentas automatizadas para sua monitoração e controle.

Na computação em nuvem, o monitoramento desempenha um papel fundamental para garantir um bom funcionamento da mesma, conforme Sousa, Moreira e Machado (2011, p. 09):

> O monitoramento é essencial para o ambiente de computação em nuvem, visto que este ambiente está se alterando constantemente de forma a atingir os objetivos. Para garantir a qualidade do serviço, utiliza-se a abordagem baseada em acordo de nível de serviço (SLA), que contêm informações sobre os níveis de disponibilidade, desempenho e penalidades em caso de violação destes níveis. No ambiente em nuvem, o objetivo é minimizar a quantidade de recursos necessários para garantir a qualidade do serviço, o que reduz os custos.

No ambiente de nuvem também existe diversas razões que justificam a necessidade do monitoramento. De acordo com Elmroth e Larsson (2009) os essenciais são: Garantir que as

MV (Máquinas Virtuais) obtenham capacidade de estipular em SLAs, e Coletar dados que serão utilizados, por exemplo, para contabilizar os recursos utilizados. Além dessas razões, Elmroth e Larsson (2009) também indicam dois tipos complementares de monitoramento que devem ser realizados pelo sistema de monitoramento: Medição da Infraestrutura (medição da disponibilidade de acesso a MV's), Indicadores-chaves de desempenho específico da aplicação (Ex. número de usuários logados no servidor).

De acordo com Chaves (2010) *o que* monitorar em um ambiente de nuvem, está relacionado aos serviços contratados e aos níveis de serviços do acordo de um SLA. E *como* monitorar sofre influência do que está sendo monitorado. Segundo a autora em CN as definições de *o que* e *como* monitorar nem sempre são atividades triviais, uma vez que além de depender do modelo de implantação utilizado (IaaS, PaaS e SaaS), existem também aspectos legais e a própria dinamicidade do paradigma.

Para Taurion (2009) mesmo com a multiplicidade de serviços prestados pelos provedores de Computação em Nuvem, que são: e-mails, desenvolvimento de aplicativos personalizados, armazenamento de dados e gestão de infraestrutura, podemos considerar que esses são concentrações maciças de recursos e dados. A percepção que se tem é de que a nuvem é um aglomerado de informações, e isso pode caracterizá-la como sendo um alvo propício a ataques por potenciais invasores. Ameaças como esta podem afetar diretamente os pilares da segurança da informação: disponibilidade, confidencialidade e integridade, e consequentemente comprometer toda a nuvem. Segundo o autor, a garantia do cumprimento desses princípios está diretamente relacionada à escolha do modelo de implantação da nuvem, a exemplo pode-se citar o modelo de Nuvem privada que possibilita a restrição de acessos visto que a nuvem está localizada atrás do *Firewall* da empresa, mantendo assim o controle do nível de serviço e aderência às regras de segurança da empresa.

3.2 Arquitetura de Monitoramento para Computação em Nuvem Privada

De acordo com Chaves (2010) a computação em nuvem não se trata apenas de um paradigma computacional, mas também de um modelo de negócios, pois o mesmo traz consigo um amplo espectro de atividades gerenciais de mais alto nível, no qual podemos citar, por exemplo, a escolha do tipo de provedor de serviço. Dessa forma, é importante que seja estabelecida uma arquitetura de monitoramento, que possa englobar esse fator e outros

também, permitindo assim, que seja possível exercer as atividades usuais de planejamento, provisionamento, escalonamento e outras, de forma que seja uma agregação importante ao desenvolvimento e utilização do paradigma de computação em nuvem. Segundo Menascé (2013, p 3):

> Monitoramento de nuvem é uma tarefa de suma importância para ambos os provedores e consumidores. Por outro lado, é uma ferramenta fundamental para o controle e gerenciamento de infraestruturas de hardware e software, pois ele fornece informações e indicadores chave de desempenho (KPIs) para ambas as plataformas, e o monitoramento contínuo de aplicações.

Uma das tarefas mais desafiadoras para a aplicação e desenvolvimento de serviços, antes da opção de adoção em larga escala da computação em nuvem, sempre foi o de recursos e capacidade de planejamento. De acordo com Menascé (2013, p 3):

> Para garantir o do desempenho exigido por aplicações e serviços os desenvolvedores é preciso: (i) quantificar a capacidade e os recursos (e.g.CPU, memória, armazenamento, etc) a serem comprados, dependendo de como essas aplicações e serviços são projetados e implementados, (ii) determinar a carga de trabalho prevista. Enquanto que a estimativa pode ser obtido através de uma análise estática, testes e monitoramento. Para este fim, monitoramento é essencial para os fornecedores de serviço de computação em nuvem para prever e acompanhar a evolução de todos os parâmetros envolvidos no processo de garantia de QoS , a fim para planejar adequadamente sua infraestrutura e recursos respeitando os SLAs.

Para Shao e Wang (2013), o primeiro passo para gerenciar um sistema complexo como a nuvem, consiste em ter um sistema de controle capaz de capturar seu estado. E ao longo dos anos, percebeu-se que a virtualização tornou-se esse componente-chave para implementar a computação em nuvem. A virtualização possui a capacidade de esconder a grande heterogeneidade de recursos físicos da infraestrutura, a partir dela introduziu-se outro nível de complexidade para o provedor de infraestrutura, que possui a função de gerir os recursos físicos e virtuais.

Os serviços em nuvem são fornecidos através de dados em larga escala, cuja gestão é uma atividade muito importante. Essa é a proposta para arquitetura de monitoramento de nuvem. De acordo com Chaves (2010), essa análise possibilita visualizar melhor as ferramentas que serão utilizadas na implementação.

A figura 4 ilustra a arquitetura de monitoramento de uma nuvem privada, a mesma divide-se em 3 camadas: de visualização, integração e infraestrutura.

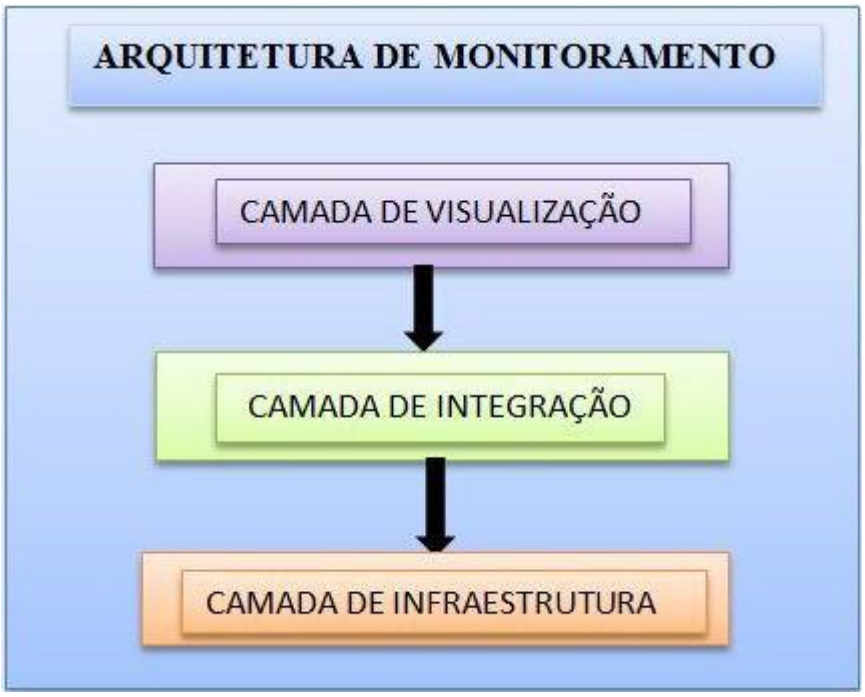

Figura 4 - Arquitetura de Monitoramento para computação em nuvem privada.
Fonte: Chaves, 2010.

1- Camada de Visualização

Chaves (2010) afirma que essa camada é a de maior nível hierárquico. Ela é utilizada como a interface de gerenciamento de alto nível, através da qual é possível fazer a verificação de informações como o cumprimento de políticas organizacionais e SLAs. Significa dizer que os usuários dessa camada geralmente vão estar interessados, principalmente na verificação das seguintes informações:

- Imagens de máquinas virtuais disponíveis para instanciamento;
- Níveis de serviço disponíveis e/ou negociáveis para as MV's instanciadas;
- Dados disponíveis de monitoramento das MV's em execução.

2- Camada de Integração

Segundo Chaves (2010) em um ambiente de computação em nuvem, a camada de integração será composta por hardware e software heterogêneos, os quais precisarão de uma interface comum de comunicação ou de acesso. Ao fazer uma solicitação qualquer para a nuvem, o usuário geralmente não conhece detalhes da implementação dessa infraestrutura e nem deve ter que se preocupar com as medidas necessárias na hora da utilização do paradigma para que se tenha informações de monitoramento disponíveis posteriormente.

3- Camada de Infraestrutura

Nessa camada se encontram o software e o hardware disponíveis na nuvem (em utilização ou que possam vir a ser utilizados):

- Hardware: dispositivos de armazenamento, processamento, de rede, entre outros.
- Software: sistema operacional (tanto das máquinas físicas que compõem o hardware citado acima, com as imagens para MV's), aplicativos diversos, licenças, hipervisores, entre outros.

Para Chaves (2010, p.73) um ponto importante nessa camada é a granularidade da informação, uma vez que quanto mais detalhes se têm sobre essa camada, maior gama de organização do monitoramento é possível. Por exemplo, na camada de visualização, poderia se organizar a visualização das informações de monitoramento, por equipamento, por cluster, entre outros.

3.3 Frameworks de Monitoramento

Segundo Vitti (2012 apud Verma, 2009) a partir do processo de monitoramento, é possível obter informações sobre os elementos do sistema, essas informações facilitam o entendimento das situações que ocorrem na nuvem tais como: estatística de uso, informações sobre erros e topologia do sistema. O processo de monitoramento irá depender de técnicas para coletar, armazenar, processar e disponibilizar estas informações.

Para Benítez (2011, p. 1) "um papel importante é o do monitoramento de recursos considerando a necessidade de se ter informações sobre a disponibilidade dos mesmos". Diversas ferramentas estão disponíveis para o monitoramento de servidores de serviços, sistemas distribuídos, clusters[6] e grids[7].

Nesta seção serão apresentadas as ferramentas mais utilizadas na literatura que podem ser implementadas para paradigmas da computação distribuída que são: Cacti, Zabbix, Nagios, EXEHDA-RM e OpManager. De acordo com Carvalho (2010) "muitas destas ferramentas trazem rotinas prontas para o monitoramento de serviços amplamente utilizados, restando para o administrativo apenas compor sua ferramenta de monitoramento como blocos de construção".

[6] Conjunto de computadores interligados entre si com o objetivo de ganho de performance e disponibilidade.
[7] É um modelo emergente de computação distribuída que permite o compartilhamento de recursos computacionais entre usuários conectados através de uma rede de computadores.

3.3.1 Nagios

De acordo com Ferreira (2011), o Nagios é uma ferramenta de código fonte aberto que foi desenvolvida por Ethan Galstad. Essa ferramenta permite efetuar a monitorização de um sistema, e dessa forma, vai apresentando todos os resultados obtidos a partir de uma interface Web. Isto significa que a aplicação possui a capacidade de verificar constantemente o estado das máquinas presentes na rede, como também os respectivos serviços. O primeiro nome que foi atribuído ao Nagios foi "NetSaint", esse *framework* foi desenvolvido para ambiente Linux, mas com o decorrer do tempo, o autor do programa e a sua equipe de colaboradores fizeram às mudanças necessárias para que esta aplicação se tornasse compatível com outros sistemas operacionais.

Ferreira (2011) afirma que os dois principais requisitos para utilização do Nagios são uma máquina com um sistema operativo UNIX e um compilador C. Além disso, é necessário que a pilha TCP/IP (*Transmission Control Protocol/ Internet Protocol* – Protocolo da Internet) esteja devidamente configurada, uma vez que a grande maioria das verificações é feita pela rede. A arquitetura do *framework* Nagios é baseada no conceito cliente e servidor. O servidor funciona como se fosse uma máquina responsável por efetuar o processo de monitoramento, enquanto o cliente constitui a máquina remota que é monitorada.

Benítez (2011) diz que o Nagios possui a capacidade de gerenciar recursos: de equipamentos, carga de CPU, ocupação de disco e memória dentre outros aspectos. Alguns recursos do Nagios segundo Vitti (2012) são:

- Monitoramento de serviços de rede (SMTP, POP3, HTTP, NNTP, PING);
- Monitoramento dos recursos das máquinas (carga do processador, uso de disco);
- Monitoramento de hosts executando diversos sistemas operacionais como Microsoft Windows, Unix/Linux, Novell NetWare, e outros;
- Sistema simples de plug-ins que permite aos usuários desenvolver suas próprias verificações de serviços;
- Verificação dos serviços paralelamente;
- Habilidade para definir hierarquias da rede, permitindo a detecção e a distinção entre as máquinas que estão inativas e as que são inalcançáveis;

- Notificações quando problemas de serviços ou de máquinas ocorrerem ou são resolvidas (por e-mail, SMS, *pager* ou algum método definido pelo usuário);
- Capacidade para definir manipuladores de eventos a serem executados durante eventos de serviços para uma resolução pró-ativa do problema;
- Capacidade de utilizar uma variedade de protocolos de rede, incluindo HTTP, SNMP, e SSH, para realizar o monitoramento.

3.3.2 Cacti

Segundo Benítez (2011), o Cacti é uma ferramenta de monitoramento, que exibe informações sobre o estado de uma rede de computadores através de gráficos. Consiste em um *front end* para o *RRDTool*[8], o qual é o responsável pelo armazenamento de dados e geração de gráficos. Black (2008), afirma que o Cacti gera gráficos referentes a uso de memória física, memória virtual, quantidade de processos, processamento, tráfegos de rede, quantidade não só dos sistemas operacionais Linux, mas também de Windows e de dispositivos de rede como roteadores e *switches*, bem como qualquer dispositivo que suporte SNMP (*Simple Network Management Protocol* - Protocolo Simples de Gerenciamento de Rede*).*

O Cacti busca informações via SNMP de um dispositivo que tem efeito direto nas opções SNMP disponíveis nele. Essa ferramenta foi escrita em PHP (*Hypertext Preprocessor ou Personal Home Pager* - Processador de Hipertexto) e suas principais características são:

- Distribuído sob os termos da GNU/GPL;
- Suporte ao protocolo SNMP;
- Número ilimitado de gráficos por host;
- Apresenta uma interface amigável ao usuário;
- Envio de alertas via e-mail e script personalizado;
- Gerenciamento via web.

Entretanto, ela possui alguns pontos fracos, comparado com outros *frameworks* de monitoramento, pois seu desempenho com relação a armazenamento, exibição de dados e gráficos em uma LAN (*Local Area Network* - Rede Local) menor será mais satisfatório do que em uma rede complexa.

[8] É a abreviação de *Round Robin Database*, sistema cujo objetivo é armazenar e monitorar dados em série obtidos durante um período de tempo pré-determinado.

3.3.3 Zabbix

Segundo Ferreira (2011), o Zabbix apresenta-se como uma boa alternativa ao Nagios. Ele é considerado um sistema de monitorização semi-distribuído com uma gestão centralizada que oferece uma solução para monitorizar a disponibilidade e o desempenho de servidores, equipamentos de rede e aplicações.

De acordo com Benítez (2011), o Zabbix foi desenvolvido pela Zabbix SIA. É um software distribuído sob termos da GNU (*Generel Public License).* Contempla suporte comercial que é fornecido pela Zabbix *Company.*

Existem alguns requisitos mínimos para a utilização do Zabbix, dessa forma podemos cita-los como: sistema de compilação GCC (*GNU Compiler Collection* - Coleção de Compiladores), o *automake*[9] e o sistema de gestão de base de dados MySQL. Porém, dependendo da distribuição que for utilizada e das funcionalidades, poderá ser necessária a instalação de pacotes adicionais.

Segundo Ferreira (2011), da mesma forma como o Nagios, Zabbix também funciona segundo o modelo de servidores e agentes. Os agentes são instalados nas máquinas remotas e possuem o objetivo de recolher informações, enquanto os servidores reúnem um conjunto de componentes que constituem o núcleo do programa. No Zabbix, o monitoramento das máquinas pode ocorrer da forma ativa ou passiva. Na forma ativa o servidor envia ao *host*[10] que está controlando a rede um pedido sobre determinado recurso, e dessa forma, ele obtém depois a respectiva resposta. Já no caso da monitorização passiva, o agente é quem toma a iniciativa, de enviar ao servidor um pedido sobre todos os comandos disponíveis.

A principal vantagem do Zabbix é a facilidade de manipulação dos objetos, o que agiliza muito o trabalho do dia-a-dia. As características do Zabbix são:

- Possui suporte a maioria dos sistemas operacionais: Linux, Solaris, HP-UX, AIX, FreeBSD, OpenBSD, NetBSD, Mac OS X, Windows, entre outros;
- Monitora serviços simples (http, pop3, imap, ssh) sem o uso de agentes;
- Suporte nativo ao protocolo SNMP;

[9] É uma ferramenta parágrafo que automaticamente gera arquivos Makefile.in a partir de Arquivos Makefile.am. Ele serve para a compilação de software.

[10] É um hospedeiro de qualquer computador ligado a rede.

- Interface de gerenciamento Web, de fácil utilização;
- Integração com banco de dados (MySQL, Oracle,PostgreSQL ou SQLite);
- Geração de gráficos em tempo real;
- Fácil instalação e customização;
- Agentes disponíveis para diversas plataformas: Linux,Solaris, HP-UX, AIX, FreeBSD, OpenBSD,SCO-OpenServer, Mac OS X, Windows 2000/XP/2003/Vista;
- Agentes para plataformas 32 bits e 64 bits;
- Integração com os Contadores de Performance do Windows;
- Software Open Source distribuído pela Licença GPL v2;
- Excelente Manual (Em Inglês) – Possui licenciamento próprio – Não GPL;
- Envio de alertas para: e-mail, Jabber, SMS;
- Scripts personalizados.

3.3.4 EXEHDA-RM

De acordo com Benítez (2011), o EXEHDA-RM é um *framework* responsável por gerenciar a conectividade global e o deslocamento dos dispositivos em um ambiente ubíquo, pois em ambientes ubíquos os recursos devem estar compartilhados para que possam ser acessados de qualquer lugar, e a qualquer momento. O EXEHDA-RM é um *middleware*[11] adaptativo e o seu contexto é baseado em serviços, esses serviços criam um ambiente ubíquo e fazem o gerenciamento de diferentes aplicações sobre o mesmo.

O EXEHDA-RM poderá ser acessado pelo administrador via uma interface Web. Isso possibilitara ao administrador ter uma visão geral dos recursos disponíveis (dispositivos e serviços) integrantes do sistema. O administrador defini os mecanismos de notificação que serão disparados automaticamente caso algum dos recursos do ambiente pare de funcionar. Essas notificações podem ser enviadas tanto aos usuários quanto ao próprio administrador.

Principais funcionalidades do EXEHDA-RM:

- Monitoramento de aplicações, serviços, sistemas operacionais e componentes de infraestruturas computacionais;

[11] A camada de *middleware* concentra serviços como identificação, autenticação, autorização, diretórios, certificados digitais e outras ferramentas para segurança.

- Suporte a agentes SNMP, NRPE, NSClient;
- Permite definir a hierarquia de hosts na rede e exibe um grafo com a topologia;
- Visão centralizada de todos os sistemas monitorados;
- Informações detalhadas dos componentes monitorados através de interface Web;
- Notificações customizáveis via e-mail, SMS e Gtalk, para o envio diretamente a pessoa responsável;
- Pré-configuração para ações de eventos, como reiniciar um serviço que ficou indisponível;
- Histórico de envios de alertas e notificações;
- Multiusuário Web com níveis de acessos.

3.3.5 OpManager

Segundo Black (2008) o *OpManager* é um software completo de gerenciamento de rede. Ele é desenhado para oferecer a integração entre *help-desk*, WAN (*Wide Area Network ou Ampla área de Trabalho*), servidores, aplicações, gerenciamento de ativos e análise de tráfego da WAN, bem como monitoramento *real-time* de *firewalls*, servidores Windows/Linux/Unix, servidores de e-mail Exchange, servidores Active Directories, roteadores, impressoras entre outros. Escrito em Pearl e Python, o OpManager automatiza varias tarefas de monitoramento e remove a complexidade associada ao gerenciamento da rede, com um complexo mas eficiente sistema de alertas e gatilhos, notificando instantaneamente os administradores quando e como ocorrem erros.

Ainda segundo Black (2008), a interface e *design* do Opmanager são simples e que a instalação e configuração do software, podem ser coordenadas pelo administrador. Suas características são:

- Monitoramento da infraestrutura de rede Servidor, *switches*, roteadores, impressoras, eventos, URL, serviços, aplicações e outros;
- Suporte a SBMP, WMI, CLI, NMAP;
- Monitoramento de aplicações – MS SQL Server, *Active Directory*, MS Exchange;
- Gráficos de utilização de CPU, Memória e Disco;
- Ferramentas de diagnóstico de rede;
- Monitoramento remoto;
- Alertas configuráveis e escalonáveis;

- Integrável com o *ServiceDesKPlus* e outras ferramentas de *Help Desk.*

3.4 Resumo do Capítulo

Neste capítulo foram apresentados alguns conceitos de monitoramento, sua importância em especial no ambiente de nuvem, além de explanar sobre as camadas da arquitetura de nuvem e apresentar algumas ferramentas existentes para o monitoramento de rede.

O próximo capítulo apresentará os trabalhos correlatos encontrados na literatura, voltados para o monitoramento de nuvem privada.

4 TRABALHOS CORRELATOS

Como forma de realçar a contribuição e a relevância deste trabalho, este capítulo analisa o estado da arte com o foco em monitoramento de nuvem privada e servidores virtuais, destacando alguns trabalhos relacionados ao tema. O assunto comum aos trabalhos é o monitoramento de ambientes computacionais, especialmente em nuvem privada. No Quadro 1 é apresentado um resumo dos assuntos abordados de modo a fornecer uma visão geral do objetivo de cada trabalho.

4.1 Monitoramento de Nuvem Privada e Servidores Virtuais

No trabalho de Chaves (2010) é proposta uma arquitetura para monitoramento de nuvem privada. Para validar essa arquitetura a autora desenvolveu um sistema modular e extensível para o monitoramento de nuvens privadas, chamado PCMONS (*Private Cloud Monitoring System* – Sistema de Monitoramento de Nuvem Privada). Esse sistema foi capaz de monitorar máquinas virtuais e dar suporte à plataforma de software para computação em nuvem *Eucalyptus* e para o software de monitoramento *Nagios*.

Outro trabalho correlato encontrado na literatura foi o de Uriarte (2012) que propõe o uso dos conceitos da Computação Autônoma (CA) para o gerenciamento do ambiente de computação em nuvem. O autor utilizou a propriedade de autoproteção da CA em ambiente de nuvens privadas. Para validar sua proposta, Uriarte desenvolveu uma arquitetura e um arcabouço (*open source*) chamado de PANOPTES para realizar o monitoramento de uma nuvem privada. O PANOPTES possui uma arquitetura multi-agente, e se dividi em dois principais módulos, o de segurança e o de monitoramento, uma vez separados funcionam de forma autônoma, além de ser um software distribuído e escalável. O PANOPTES surgiu com o intuito de ser um primeiro passo para a automação em nuvens privadas.

Em Gonçalves (2011), é feita a implementação de uma nuvem privada utilizando o *OpenNebula* (ferramenta de código aberto utilizada para implementar nuvens privadas, híbridas, comunitárias e até nuvens públicas), com o propósito de averiguar a sua confiabilidade e viabilidade na entrega dos serviços aos clientes. O autor afirma que o desconhecimento da computação em nuvem e um relacionamento confiável de seus serviços torna esse assunto um tanto nebuloso e por vezes retarda a migração natural da tecnologia.

Nesse sentido, o autor desenvolveu um sistema web implementando algumas métricas de acordo com o nível de serviços (Número de MV, Carga média de uso da CPU, Estabelecimento de conexão segura e *Ping*) onde esse sistema se comunica com a nuvem e gera relatórios para os usuários e para o provedor de serviços, permitindo assim a realização de ajustes futuros no SLA.

Em Vitti (2012), foi proposto o desenvolvimento de uma extensão para integração do *OpenNebula* com o PCMONS, uma vez que este era compatível somente com nuvens *Eucalyptus*. Para isso o autor realizou algumas adaptações no *script* do PCMONS tornando-o compatível com a nuvem *OpenNebula*, possibilitando assim o monitoramento da nuvem privada implementada no experimento.

Carvalho (2010) propõe a realização de adaptações no monitor Nagios para que esta ferramenta se adeque a dinamicidade do ambiente virtualizado sem a intervenção dos administradores. Com essas adaptações, foi possível integrar as informações de monitoramento de máquinas físicas com as informações de máquinas virtuais e inserir na ferramenta de monitoramento escolhida, as informações do relacionamento entre ambas.

Os trabalhos correlatos abordados anteriormente são apresentados resumidamente no Quadro 1, e este tem como objetivo fornecer uma visão geral de suas propostas e respectivos autores.

Quadro 1- Resumo dos Trabalhos Correlatos.

Referência	Proposta
Carvalho (2010)	Adaptação da Ferramenta Nagios para o Monitoramento de Servidores Virtuais.
Chaves (2010)	Arquitetura e um sistema denominado PCMONS para o Monitoramento de Nuvem Privada.
Gonçalves (2011)	Implementação de um sistema Web para averiguar a confiabilidade na estrega dos serviços em um ambiente de nuvem privada.
Uriarte (2012)	Desenvolvimento de um arcabouço chamado de PANOPTES para o monitoramento e autoproteção de nuvens privadas.
Vitti (2012)	Desenvolvimento de uma extensão do PCMONS para integração com o OpenNebula.

Fonte: Própria.

Após análise dos trabalhos mencionados, foi possível verificar que os autores empenharam-se em desenvolver arquiteturas, sistemas, adaptações de software para a realização do monitoramento de ambientes virtualizados e de nuvem privada. No entanto também foi possível observar que o único monitor de rede utilizado nas propostas dos trabalhos correlatos foi o *Nagios* o que restringe o monitoramento da nuvem a este *framework*. Vale ressaltar, que o *Nagios* possui uma interface com poucos recursos de gerenciamento e um tanto arcaica se comparado com outras soluções de código aberto disponíveis atualmente, tal como o *Zabbix*. Além da interface, o *Nagi*os apresenta lentidão na varredura da rede e requer um bom conhecimento técnico por parte do administrador, pois o processo de configuração do parque computacional é bastante trabalhoso.

A presente proposta, diferente das demais, faz o uso do Zabbix, que é uma ferramenta bastante difundida na literatura e agrega características importantes para o monitoramento, o que a torna um *framework* robusto. Esta solução é provavelmente a mais completa entre as opções de código aberto para o monitoramento de rede atualmente, uma vez que ela possui uma interface avançada que permite o controle total do sistema, desde cadastramento de ativos, inserção de serviços a serem monitorados bem como criação de cenários para monitoramento dinâmico, é também compatível com diversos sistemas de gerenciamento de banco de dados, gera gráficos em tempo real e é de fácil manipulação visto que qualquer administrador de rede com pouco conhecimento é capaz de configurar ativos devido a sua interface intuitiva.

Assim, nenhum dos trabalhos pesquisados visou integrar o *Zabbix* a uma nuvem privada, para realizar o monitoramento de máquinas virtuais e máquinas reais que a compõe, no qual o presente projeto se propõe a fazer.

4.2 Resumo do Capítulo

Este capítulo teve como objetivo apresentar alguns trabalhos relacionados ao tema deste projeto. Com eles, foi possível verificar o interesse do meio científico em desenvolver arquiteturas, técnicas e adaptações para o monitoramento de nuvem privada.

No capítulo seguinte serão apresentados detalhadamente os métodos e as ferramentas utilizadas no estudo de caso do presente trabalho.

5 MATERIAIS E MÉTODOS

Este capítulo abordará os procedimentos e as ferramentas utilizadas para o desenvolvimento e validação da proposta do projeto. Ele tem por objetivo, detalhar as soluções implementadas, justificar o seu uso, além de apresentar outras soluções com fins semelhantes disponíveis atualmente.

5.1 Metodologia

A primeira etapa foi o estudo do estado da arte sobre a computação em nuvem, onde foi possível averiguar o conceito, funcionamento, classificações destes novos modelos de distribuição de serviços computacionais. Nesta etapa também foi realizado um estudo bibliográfico sobre *framework* para o monitoramento de rede, além da realização de pesquisas para escolha das ferramentas utilizadas no experimento e a modelagem do cenário do estudo de caso.

Em seguida, foi planejada a topologia da rede local, e a definição das tecnologias de software e hardware a serem adotados. A topologia estrela foi utilizada no presente estudo de caso. Com a topologia definida, foi configurada uma LAN (*Local Area Network* – Rede Local) na sala de pesquisa da FACOM da Universidade Federal do Pará - Campus Marabá. Nesta LAN, foram utilizados uma máquina Desktop, um Notebook e um Roteador, todos com acesso a Internet.

Na segunda etapa, foi realizado o processo de instalação e configuração da nuvem privada, onde foi instalado o *frontend*[12] da nuvem na máquina Desktop e o Nó controlador[13] no Notebook. Após esses procedimentos foram instaladas e configuradas as instancias virtuais na nuvem privada.

A terceira etapa envolveu o processo de instalação, configuração e adaptação de um *framework* de monitoramento em ambiente de nuvem privada, para realizar o monitoramento das máquinas físicas e virtuais.

Por fim foram definidas as métricas dos itens a serem monitorados de modo a verificar o desempenho e disponibilidades das máquinas, a fim de se coletar e avaliar os resultados obtidos.

[12] É o nome da máquina que contém o Controlador de Nuvem (CN), Controlador de Cluster (CC) e o Walrus.
[13] Nome da máquina que possui conexão com o *Frontend*, é nele que as máquinas virtuais estarão instaladas.

5.2 Ferramentas

Nesta seção serão apresentadas as soluções utilizadas no experimento. Destacamos que foi priorizado o uso de soluções de código aberto, uma vez que o objetivo deste projeto também foi de utilizar os recursos de hardware e software acessíveis.

5.2.1 Definição da Topologia da Rede Local

Para a modelagem da topologia e cenário do estudo de caso, foi utilizada à ferramenta Microsoft Visio versão 2013. Que de acordo com seu fabricante é uma solução que permite a criação de "diagramas profissionais para simplificar informações complexas com formas atualizadas" (MICROSOFT, 2013).

Esta ferramenta foi escolhida, pois ela também permite a criação de diagrama de redes, é de fácil manuseio, além disso, ela orquestra outras funções, como diagramas UML, porém foi utilizada para realizar somente a topologia de rede.

Existem outras ferramentas de modelagem no mercado, entre as quais podemos citar a StarUML, que é uma ferramenta de código aberto utilizada para um desenvolvimento rápido e extensível de diagramas e topologias de rede, possui vários recursos, um deles é a instalação e criação de plug-ins, possibilitando assim, que o usuário manipule sua arquitetura, adequando a forma que deseja (StarUML, 2013).Porém essa ferramenta de modelagem não foi utilizada porque apresenta-se a necessidade de conhecer bem a arquitetura da aplicação para utiliza-la e principalmente o fato de possuir uma versão muito antiga e que nunca mas foi atualizada desde 2005.

O produto Rational Modeler da IBM (*International Business Machines*) é outra ferramenta de *designer* gratuita. Oferece um sistema customizável e eficiente para o usuário, o mesmo pode escolher uma plataforma de linguagem aberta e extensiva para modelar os aplicativos e topologia de redes de forma muito mais produtiva e menos cansativa, além de possuir suporte ao usuário em caso de falhas (RATIONAL, 2013). Não foi utilizada essa ferramenta porque ela necessita de um conhecimento mais aprofundado, o que requer mais tempo de estudo.

5.2.1 Implementação da Infraestrutura de Software

A solução usada para a implantação da infraestrutura da nuvem foi o *Eucalyptus FastStart* Versão 3.3.0.1. Esta é uma ferramenta que permite a criação de nuvem de pequeno porte. Segundo o Eucalyptus (2013) essa versão é voltada para usuários que queiram criar uma nuvem com poucos recursos de hardware, desenvolver e testar suas aplicações, além de adquirir habilidade em nuvens. Essas características foram um fator preponderante para a escolha da ferramenta, uma vez que as mesmas se enquadram aos objetivos do projeto, pois o mesmo necessita de um ambiente totalmente controlado e possui recursos computacionais limitados.

Além do *Eucalyptus* existem outras soluções para implantação de nuvens na qual pode-se citar: *OpenNebula, Nimbus e OpenStack*. O *OpenNebula* é uma ferramenta que "possui um alto nível de centralização, flexibilidade e customização, é uma plataforma adaptável para a construção de nuvens privadas e híbridas num datacenter pré-existente" (OpenNebula, 2013). Ela possui uma arquitetura flexível, possibilitando assim a portabilidade com vários *DataCenters*, porém não foi utilizada essa ferramenta pois a mesma é voltada para aquelas pessoas que possuem um bom conhecimento técnico, possibilitando assim que o usuário possa manipular e obter maiores funcionalidades, adequando-o de acordo com a sua necessidade.

Já a ferramenta *OpenStack* é uma colaboração global de desenvolvedores e tecnólogos de computação em nuvem que produzem a plataforma de computação ubíqua em nuvem para nuvens públicas e privadas (OpenStack, 2013). O *OpenStack* não foi utilizado porque é usada para corporações, provedores de serviços, pesquisadores e centros de dados globais, que possuem a perspectiva de implantar nuvens públicas ou privadas em grande escala.

A solução *Nimbus* "é um kit de ferramentas de código aberto focada no fornecimento de Infraestrutura como Serviço (IaaS) e recursos para a comunidade científica" (Nimbus 2013). Ela permite que seus provedores construam nuvens privadas ou comunitárias, além da integração de uma nuvem pública com a privada e que os mesmos possam fazer as configurações que desejarem na infraestrutura. Esta solução não foi utilizada porque sua função é criar nuvens em grande escala.

O sistema operacional utilizado para acomodar o *Eucalyptus FastStart*, foi o CentOS Versão 6.4, pois o mesmo além de ser nativo do próprio *Eucalyptus* possui uma interface agradável para que o usuário possa manipular as máquinas virtuais, além disso, possui um

atalho que se conecta na interface de administrador e no console do *Eucalyptus*, possibilitando que as máquinas sejam instanciadas em uma interface gráfica e legível. Ele também é compatível com as arquiteturas i386 e x86_64. Resumindo, CentOS é um sistema que possui performance e segurança (CentOS, 2013).

Existem outros sistemas operacionais de código aberto que podem ser utilizados em uma nuvem privada, tais como o *Ubuntu Server Cloud* e *Debian.* O *Ubuntu Server Cloud*, por exemplo, possui suporte nativo para o *Eucalyptus*, porém ele é um servidor e não possui interface gráfica, assim, todas as atualizações e pacotes são feitos manualmente. Um usuário menos experiente pode ter problemas utilizando essa solução. Além disso, as versões que possuem suporte para o *Eucalyptus* são 10.04, 10.10, 11.04 e 11.10, dessa forma, quando o *Ubuntu* atualiza suas versões de sistemas operacionais (hoje estamos na 13.01), os pacotes de atualização das antigas versões são retirados dos servidores da Canonical (distribuidora do *Ubuntu*), então alguns dados não são atualizados, causando problemas na infraestrutura da nuvem (UBUNTU, 2013).

O sistema operacional *Debian* possui interface gráfica e pode ser instalado em quase todos os computadores existentes, até mesmo os mais antigos. Uma de suas qualidades é a interface limpa e de fácil entendimento. Porém, o *Debian* não possui suporte nativo para o *Eucalyptus*, o que forçaria o usuário a instala-lo no sistema operacional, e esta instalação ocorre via comandos no terminal, assim um usuário pouco experiente, poderia ter que enfrentar alguns problemas na instalação (DEBIAN, 2013).

O hipervisor utilizado no projeto foi o KVM (*Kernel Virtual Machine* - Máquina Virtual Baseada em Núcleo). Este é uma maquina virtual baseada em kernel, e que possui uma solução de virtualização completa para o Linux com hardware x86, visto que ele contém extensões de virtualização para os processadores Intel VT ou AMD-V. Ao utilizar o KVM, é possível realizar a execução de uma série de máquinas virtuais que possuem o Linux, sem modificações em outros sistemas operacionais, como por exemplo, o Windows (KVM, 2013).

Outro hipervisor de código aberto, disponível atualmente é o Xen. De acordo com Xen (2013), esta solução é usada como base para uma série de aplicações comerciais, tais como: virtualização de servidores, Infraestrutura como Serviço, virtualização de *desktop*, aplicações de segurança e dispositivos de hardware. Pode ser instalado em qualquer arquitetura de processador, até os que não possuem suporte para virtualização, além disso, possui virtualização total, ou seja, toda a estrutura de hardware é virtualizada. Como o presente projeto faz o uso de máquinas com poder computacional limitado e o XEN visa a

virtualização total, as máquinas virtuais ficariam bastante lentas o que prejudicaria no processo de monitoramento, ou seja, foi preferível utilizar o KVM por utilizar pouco recurso de hardware e não prejudicar o desempenho das mesmas.

Como ferramenta de gerenciamento das MV's, o *Euca2ools* 2.0.1, foi utilizado para gerenciar imagens; executar, parar e terminar uma instância; gerenciar IP estático e gerenciar pares de chaves das mesmas. Ele foi o escolhido, por ser gratuito e por estar nativo do *Eucalyptus FastStart* (EUCA2OOLS, 2013).

Entretanto existe outra ferramenta chamada *Hybridfox*. Segundo Hybridfox (2013), ela possui uma interface gráfica de fácil manipulação. Diferente do *Euca2ools* que é utilizado a partir de comandos no terminal, o *Hybridfox* possui uma interface mais robusta e de fácil entendimento. O único problema dessa ferramenta é que ela é compatível somente com o navegador Firefox, e se o mesmo for atualizado, o *plugin* do *Hybridfox* também terá que ser, o que na realidade nem sempre acontece, e consequentemente acarretará em problemas.

5.2.2 Implementação do *Framework* de Monitoramento

Foi utilizado como *framework* de monitoramento o Zabbix versão 2.0.6. Segundo Zabbix (2013), essa ferramenta de monitoramento de código aberto, no momento tem se mostrado uma das ferramentas mais robustas do mercado, porque ele oferece monitoramento preciso, emite alertas sobre a situação da rede em tempo real, possui uma interface de fácil manipulação, além de monitorar diversos dispositivos (catraca, câmera de segurança, nobreak, etc.), e gerar gráficos de seu desempenho em tempo real.

Esse *framework* permite o monitoramento através de dois tipos de recursos bastante utilizados: o Agente Zabbix que é responsável por coletar informações dos ativos monitorados e reportar ao servidor Zabbix que se encarrega de gerar relatórios e enviar alertas aos administradores sobre o funcionamento da rede; e o *Simple Check* que permite o monitoramento sem agente instalado nos ativos, onde o servidor Zabbix realiza a checagens desses ativos e estes por sua vez retornam "sim" ou "não" (0 ou 1) representando sua disponibilidade na rede. A princípio no experimento proposto por este trabalho foi utilizado o Agente Zabbix para monitorar as MV's, no entanto verificou-se que não houve comunicação entre as camadas de integração e de visualização. Por isso utilizou-se o *Simple Check* para monitorar a disponibilidades das instancias virtuais, uma vez que ele foi capaz de se comunicar com as MV's e verificar seu *status* na rede.

Outro *framework* de monitoramento é o Nagios. De acordo com Ferreira (2011), essa solução permite efetuar a monitorização de um sistema, lançando todos os resultados obtidos a partir de uma interface Web. Entretanto, ele possui uma interface arcaica e não fornece gráficos em tempo real.

Existe também a ferramenta Cacti, *esse monitor* busca informações via SNMP (*Simple Network Management Protocol* - Protocolo Simples de Gerência de Rede) dos ativos, possui um numero ilimitado de gráficos por host e apresenta uma interface amigável ao usuário. Porém, ela não é uma ferramenta robusta e seus gráficos não são exibidos em tempo real.

5.3 Resumo do Capítulo

Este capítulo objetivou abordar as técnicas e as soluções de software utilizadas para a realização do estudo de caso do presente trabalho.

O próximo capítulo tratará do estudo de caso proposto por este trabalho com o intuito de implementar e analisar o desempenho do *framework* Zabbix no monitoramento de uma nuvem privada.

6 ESTUDO DE CASO

Este capítulo tem por objetivo descrever o desenvolvimento do estudo de caso proposto neste trabalho. Nele serão especificados os softwares e hardwares utilizados, as métricas utilizadas para monitoramento da nuvem privada, e os resultados coletados após a execução dos cenários. Por fim é dada uma análise conclusiva.

6.1 Topologia da Rede e Configuração de Hardware e Software

Inicialmente foi necessário definir a topologia da rede, e assim foi realizada sua modelagem topológica, onde nesta há uma LAN composta por dois computadores físicos interligados em rede através de um roteador D-Link, onde um destes acomoda três máquinas virtuais, conforme se pode verificar na Figura 5. A partir dessa modelagem foram definidos os softwares que cada máquina receberá e o número de MV's instanciadas. A distribuição dos IP's (*Internet Protocol* - Protocolo de Internet) entre as máquinas físicas foi realizada de forma manual, onde o computador Desktop recebeu o IP 192.168.0.14 e o Notebook o IP 192.168.0.13. Já a atribuição de IP's entre as MV's foi realizada pelo *Eucalyptus*, uma vez que este foi configurado no modo *Static*, e nesse caso o administrador da nuvem reserva uma faixa de IP. Assim, neste experimento, foi utilizada a seguinte faixa de IP: 192.168.0.15 a 192.168.0.30 em que será utilizada pelo *Eucalyptus* para atribuir as MV's.

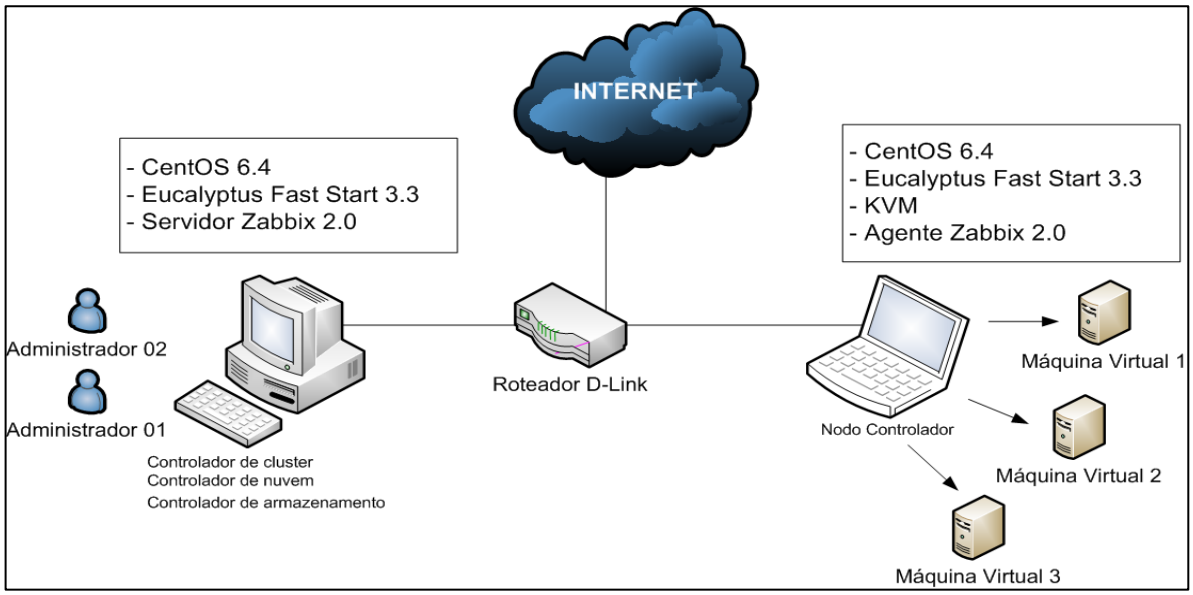

Figura 5 - Topologia do Estudo de Caso.
Fonte: Própria.

Com a topologia definida, foi realizada a implementação do ambiente de computação em nuvem, a fim de se realizar o experimento onde optou-se por configurar uma nuvem do tipo privada, uma vez que seu monitoramento exerce papel importante para verificação dos serviços que ela oferece aos seus usuários, garanti a segurança dos dados, visto que pode haver documentos sigilosos e analisa o comportamento da nuvem quando houver requisição excessiva de recursos, no caso de uma sobrecarga por exemplo, o tempo de resposta entre os dados e serviços e etc. Outro fator que contribuiu para a escolha de uma nuvem privada é que ela fornece maior controle para sua configuração, além de permitir o aproveitamento de recursos de hardware disponíveis na sala de pesquisa, conforme ilustrado na Figura 6.

Figura 6 - Hardware utilizado no experimento.
Fonte: Própria.

Vale ressaltar que o hardware utilizado nesse trabalho são equipamentos com configurações simples conforme as especificações técnicas no Quadro 2, assim como são citados os softwares utilizados, onde serão apresentados com maiores detalhes na próxima seção.

Possuir um ambiente de nuvem controlado é importante para manter a integridade dos serviços oferecidos por ela, além da segurança dos dados dos usuários na nuvem. Dessa forma, foi proposto utilizar um *framework* de monitoramento que pudesse monitorar uma nuvem, e a partir disso coletar os dados da mesma e observar seu comportamento.

Quadro 2 - Especificações Técnicas dos Hardwares Utilizados.

Hardware	Software	Papel no Ambiente
Computador HP Compaq 6005 Pro SFF, Processador: AMD Phenom™ II X4 B97 Processor 3.20 GHz, Cache do Processador: 6 MB L3 cache, Memória RAM: 4GB DDR3, Disco Rígido (HD): 500GB SATA (o sistema operacional utilizado foi instalado em uma partição de 96,13 GB).	CentOS 6.4 Eucalyptus *FastStart* 3.3.0.1 Servidor Zabbix 2.0.6	Controlador da Nuvem Controlador de Cluster Controlador de Armazenamento
Notebook Semp Toshiba Infinity IS1442, Processador: Intel Core i5-2410M 2,3 GHz, Cache do Processador: 3072 KB, Memoria RAM: 4GB DDR3, Disco Rígido (HD): 500 GB.	CentOS 6 Eucalyptus *FastStart* 3.3.0.1 KVM Agente Zabbix	Nó Controlador
Roteador D-Link- Ethernet Broadband Router. Modelo Dl-604.	-	Atribuir IP's e acesso a Internet para as maquinas do experimento

Fonte: Própria.

6.2 Arquitetura Proposta

Visando monitorar o desempenho e a disponibilidade das máquinas na nuvem privada, foi planejada uma arquitetura que conforme ilustrada na Figura 7, está dividida em três camadas, onde na camada de visualização está alocado o Zabbix que é responsável por monitorar o funcionamento das máquinas, tanto virtuais como físicas, e disponibilizar, através de sua interface, gráficos demostrando os seus desempenhos e disponibilidades de acesso, possibilitando ao administrador, a detecção e correção de um potencial problema e melhor gerenciamento dos recursos de hardware. A camada de integração é composta por hardware e software heterogêneo, que precisam de uma interface de comunicação e acesso, assim nessa camada encontra-se o KVM, pois é ele que faz a comunicação entre a camada de infraestrutura e de visualização. E a ultima camada é a de Infraestrutura, que contém os

softwares e os hardwares utilizados para criação da nuvem privada, ela engloba o *Eucalyptus* e o sistema operacional CentOS.

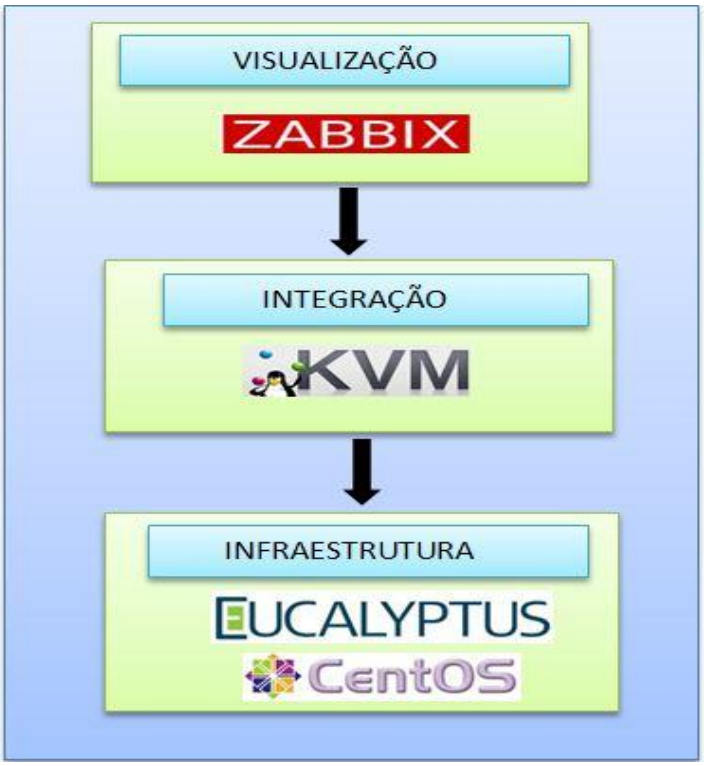

Figura 7 - Arquitetura Proposta.
Fonte: Própria.

Durante o processo de instalação do sistema operacional CentOS, se fez obrigatório definir uma faixa de IP's, pois eles serão atribuídos as máquinas virtuais, quando forem instanciadas. O arquivo/caminho que irá ser modificado para a inserção dessa faixa de IP's *"Eucalyptus.conf"* está sendo referenciado do na Quadro 3, na segunda linha.

Depois de inserir a faixa de IP, foi realizado o instanciamento das MV's através do console, em um atalho no CentOS no qual acessamos uma interface web para instanciar as mesmas. Depois de elas estarem funcionando, foi inserida uma chave: "*ssh –i ./credentials/admin/admin.private ec2-user@192.168.0.16*" que possibilita a entrada na máquina virtual instanciada. Com as máquinas funcionando, foi realizada a instalação do Zabbix Server no *Frontend*, e é lá que estarão expostos os gráficos de monitoramento das MV's, Nó Controlador e do próprio *Frontend*.

Em seguida, o agente no Nó Controlador foi inserido. O arquivo/caminho que foi modificado para que o agente pudesse se comunicar com o *Frontend*, foi o *"zabbix_agentd.conf*" do Quadro 3. Esse arquivo irá possuir os dados, ou seja, informações

como arquivos de log e etc., do servidor *Frontend*, no qual o agente irá levar as informações de CPU, memória e entre outras do Nó Controlador.

Entretanto, com o Zabbix monitorando as máquinas, o *Eucalyptus* acreditava estar sendo espionado e por questões de segurança, o mesmo modifica as chaves de acesso as MV's, impossibilitando o acesso dos usuários. Porém, no arquivo/caminho *"known_hostss"* do Quadro 3, é possível que o usuário possa modificar a chave de acesso outra vez.

Quadro 3 - Arquivos de configuração modificados.

Software	Arquivo/Caminho	Alteração
Eucalyptus FastStart	/root/.ssh/known_hosts	Alteração nas chaves de acesso as Máquinas Virtuais.
	/etc/eucalyptus/eucalyptus.conf	Neste arquivo foi definida a faixa de IPs que o Eucalyptus deveria utilizar para atribuir as MV's.
Zabbix Agente	/etc/zabbix/zabbix_agentd.conf	Atribuição do IP da máquina que contem o Servidor Zabbix.

Fonte: Própria.

Conforme as alterações realizadas no arquivo "*known_hosts*", a qual está sendo ilustrado com detalhes no APÊNDICE A, as chaves de acesso as MV's, precisaram ser alteradas, pois o *Eucalyptus* identificou uma atividade suspeita o que o levou a associar a uma tentativa de invasão, no entanto essa atividade tratava-se apenas da instalação do Zabbix. A modificação do arquivo *"eucalyptus.conf"* foi a inserção da faixa de IP 192.169.0.15 - 192.168.0.30, pra que eles fossem atribuídos pelo *Eucalyptus* as instancias virtuais. Ressaltamos que foi através dos IP's atribuídos as MV's que fora possível realizar o monitoramento simples das mesmas. Já as modificações realizadas no arquivo "*agentd.conf"* foram necessárias para definir o IP da máquina onde estava instalado o Servidor Zabbix, uma vez que o Agente Zabbix precisa conhecer o endereço do Servidor Zabbix para enviar os dados coletados na máquina. Para mais detalhes a cerca das modificações realizadas nos três arquivos mencionados, consulte os APÊNDICE A, B e C.

Como resultado de todo este processo de instalação, configurações e adaptação, a Figura 8 abaixo ilustra a tela do *Framework* Zabbix, e nela estão as três maquinas virtuais, o *Frontend* e o Nó Controlador, sendo monitorados.

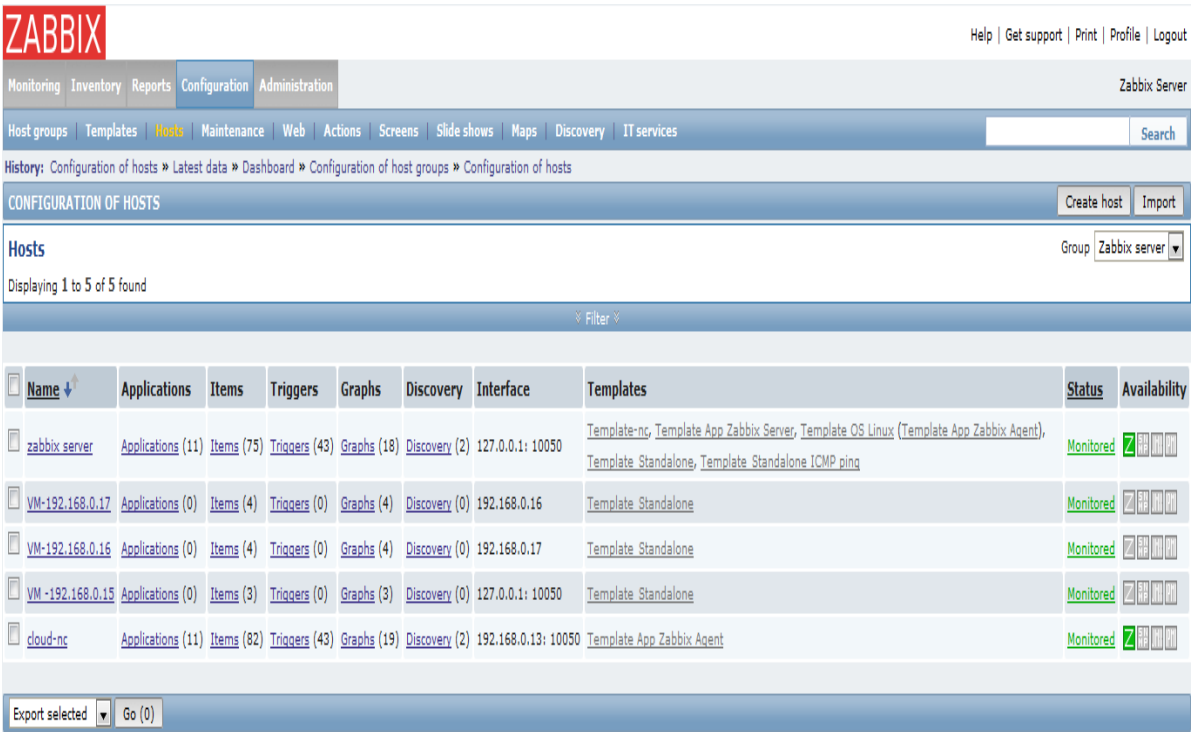

Figura 8 - Interface do Zabbix monitorando MV's e Máquinas Físicas.
Fonte: Própria.

6.3 Cenário de Avaliação e Resultados

Para a realização do monitoramento da nuvem, foram definidos dois conjuntos de métricas: um para o monitoramento das máquinas físicas e o outro para as máquinas virtuais. Enfatizamos que não foi possível utilizar as mesmas métricas para todas as máquinas envolvidas (física e virtual), pois o Zabbix não foi projetado para ambientes de computação em nuvem.

As métricas definidas para o monitoramento das máquinas físicas objetivaram obter informações sobre o seu desempenho e disponibilidades. Vale ressaltar que realizar o monitoramento das máquinas físicas se faz necessário, uma vez que é sob elas que está configurada a nuvem privada e são elas que fornecem os recursos físicos necessários para a instanciação das máquinas virtuais. Nesse sentido, as verificações da utilização de memória disponível, análise da carga e do desempenho da CPU (Unidade Central de Processamento), bem como o número total de processos ativos, são de fundamental importância para a

verificação da saúde dessas máquinas e para o gerenciamento de seus recursos. Para isso as métricas escolhidas foram:

- CPU Utilizada – Monitora a utilização da CPU
- Carga da CPU – Verifica a carga da CPU, de modo a identificar se ela esta sendo utilizada além de sua capacidade ou subutilizada.
- Memória – Essa métrica faz a verificação do uso da memória
- Numero de Processos Funcionando – É o numero de processos que estão em execução em uma máquina.

Quanto às métricas para o monitoramento das máquinas virtuais, foram definidas o SSH e o Ping. Elas provêm um conjunto mínimo de informações das instâncias como o funcionamento e disponibilidade. A métrica de Ping visa testar a conectividade com as MV's de modo a verificar a sua disponibilidade na nuvem, isto é muito importante pois verifica a disponibilidade dessa MV na nuvem .Já a métrica de SSH também desempenha um papel importante na verificação da disponibilidade das instancias virtuais, uma vez que todo o acesso a elas é feito por SSH e se porventura essa conexão falhar as MV's ficam impossibilitadas de, por exemplo, receber a instalações de novos aplicativos.

- SSH – Verifica a disponibilidade de acesso as MV's (O SSH é um protocolo que provê mecanismos para conectar de modo seguro a um sistema).
- Ping – Verifica a alcançabilidade das MV's (O Ping é um programa utilizado para testar conectividade entre ativos. Essa métrica executa o Ping a cada intervalo de tempo predeterminado).

No Quadro 4, é apresentado de forma resumida as métricas aplicadas as respectivas máquinas da nuvem privada. Ressaltamos que todas as máquinas do experimento foram monitoradas.

Quadro 4 - Métricas aplicadas às maquina do experimento.

Tipo de Requisição	Host
Ping	MV1, MV2 e MV3
SSH	MV1, MV2 e MV3
Memória	Desktop e Notebook
Carga da CPU	Desktop e Notebook
CPU Utilizada	Desktop e Notebook
Numero de Processos Funcionando	Desktop e Notebook

Fonte: Própria.

Com o decorrer dos testes, observou-se também que à nuvem privada apresenta certo nível de segura, pois após a instalação do *framework* Zabbix, o *Eucalyptus* alterou as chaves de acesso as MV's bloqueando assim o seu acesso, uma vez que ele acreditava estar sofrendo uma possível invasão, de modo que foi emitido um alerta de segurança sobre a alteração da chave conforme Quadro 5.

Quadro 5 - Mensagem de alerta de alteração de chave da MV.

```
1-[cloud@cloud ~]$ su -
2-Senha:
3-[root@cloud ~]# ssh -i ./credentials/admin/admin.private ec2-ser@192.168.0.15
4-@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
5-@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
6-@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
7-IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
8-Someone could be eavesdropping on you right now (man-in-the-middle attack)!
9-It is also possible that the RSA host key has just been changed.
10-The fingerprint for the RSA key sent by the remote host is
11-f2:f2:92:c8:86:28:be:b4:a6:69:e2:1e:00:be:22:56.
12-Please contact your system administrator.
13-Add correct host key in /root/.ssh/known_hosts to get rid of this message.
14-Offending key in /root/.ssh/known_hosts:5
15-RSA host key for 192.168.0.15 has changed and you have requested strict checking.
16-Host key verification failed.
17-[root@cloud ~]#
```

Fonte: Própria.

Para facilitar o entendimento, o Quadro 5 foi enumerado, de modo que pode-se observar que na linha 3, é executado o comando para acessar via SSH a MV1. Em seguida (linha 5) o Eucalyptus emite um alerta informando que a chave de acesso a essa MV mudou, e na linha 8 ele avisa que acredita ter sofrido uma possível invasão, e que por isso (linha9) possivelmente a chave RSA (Serviço de Acesso Remoto) da máquina foi alterada. Na linha 12 ele pede para que o usuário contate o administrador da nuvem para solicitar uma nova chave e adicioná-la no arquivo correto conforme linha 13. Nas linhas 14, 15 e 16 ele apenas enfatiza que a conexão a verificação da chave falhou.

6.3.1 Resultado do Monitoramento das MV's

Os gráficos a seguir, apresentam os resultados do monitoramento de uma MV, onde o *framework* Zabbix utilizou a métrica de Ping para testar a disponibilidade das MV's, e o monitoramento foi realizado através do *Simple Check*, um recurso do Zabbix que realiza o monitoramento simples de ativos (câmeras de segurança, nobreak, impressoras, catracas entre outros) e máquinas comuns, porém é mais utilizado para monitoramento de ativos.

Os dados retornados ao servidor Zabbix sobre o monitoramento das MV's são "0" ou "1". Para mostrar que ela está "*running"* executando, o valor do Ping dessa instancia deve ser "1", mas se for "0", significa que o valor de resposta é baixo e que a máquina não está respondendo. Nas Figuras 9 e 10, é apresentado respectivamente o resultado de um Ping bem sucedido e de um Ping perdido. No final foi feito um teste de sobrecarga nas MV's, para saber o quanto isso afeta os demais componentes da nuvem (*Frontend* e Nó Controlador) e se elas conseguem ficar ativas até o fim do teste.

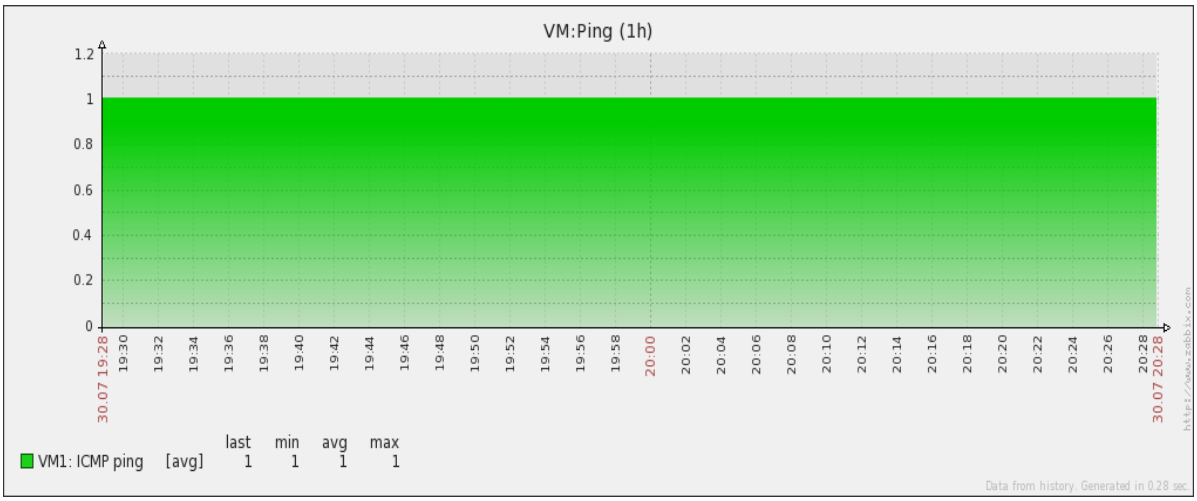

Figura 9 - Ping Bem Sucedido da MV1.
Fonte: Própria.

Foi feito um teste de falha da conectividade, uma simulação de queda na rede, com MV's, e assim o pico neste gráfico, denota o momento que houve uma falha na comunicação, ou seja, a perca do Ping.

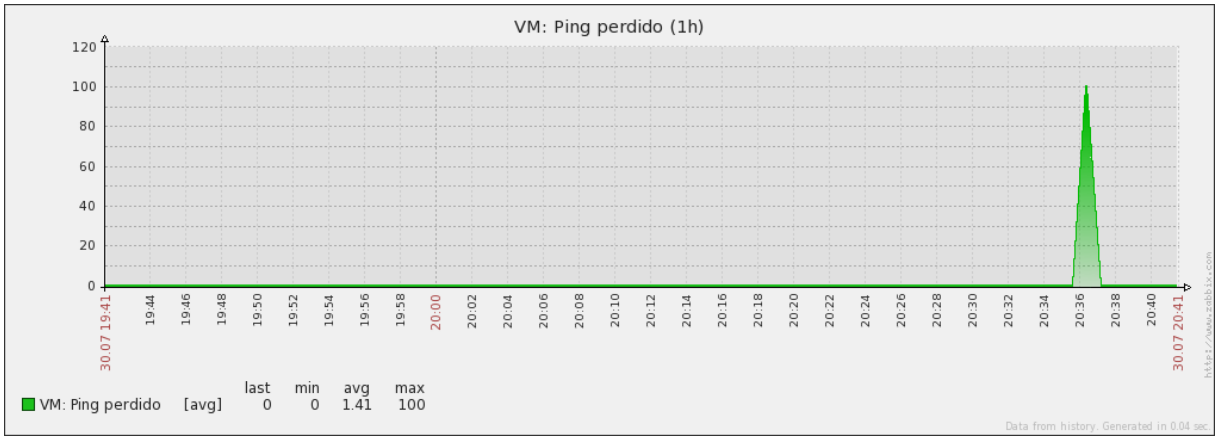

Figura 10 - Ping Perdido da MV2.
Fonte: Própria.

A segunda métrica utilizada para monitoramento das MV's foi o SSH. Monitorar o SSH das MV's é importante, visto que elas utilizam chaves SSH para serem acessadas, e através do monitoramento SSH, pode-se saber sobre o "tempo de vida" da MV, ou seja, se ela ainda está funcionando. A Figura 11 abaixo mostra uma MV em funcionamento constante.

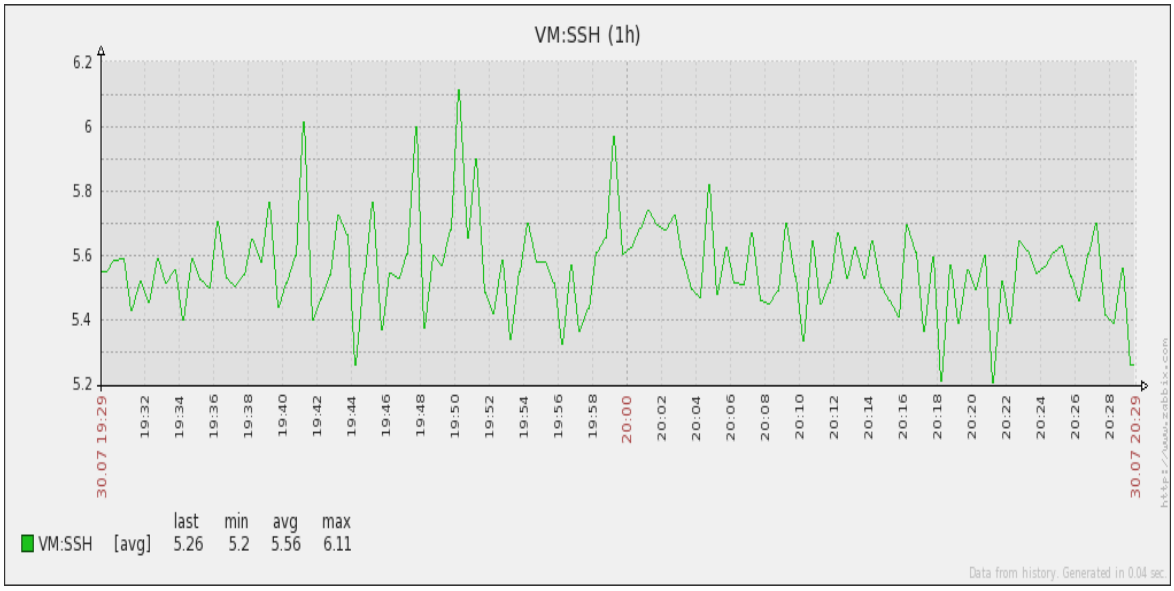

Figura 11 - Verificação do SSH da MV1.
Fonte: Própria.

O SSH da Figura 12, é de outra maquina virtual que entrou em execução após a Figura 11, e demonstra que está MV, está com chegando ao fim, ou seja, seu "ciclo de vida" está

acabando. As MV’s possuem um ciclo de 10 horas. Após isso devem ser executadas novamente.

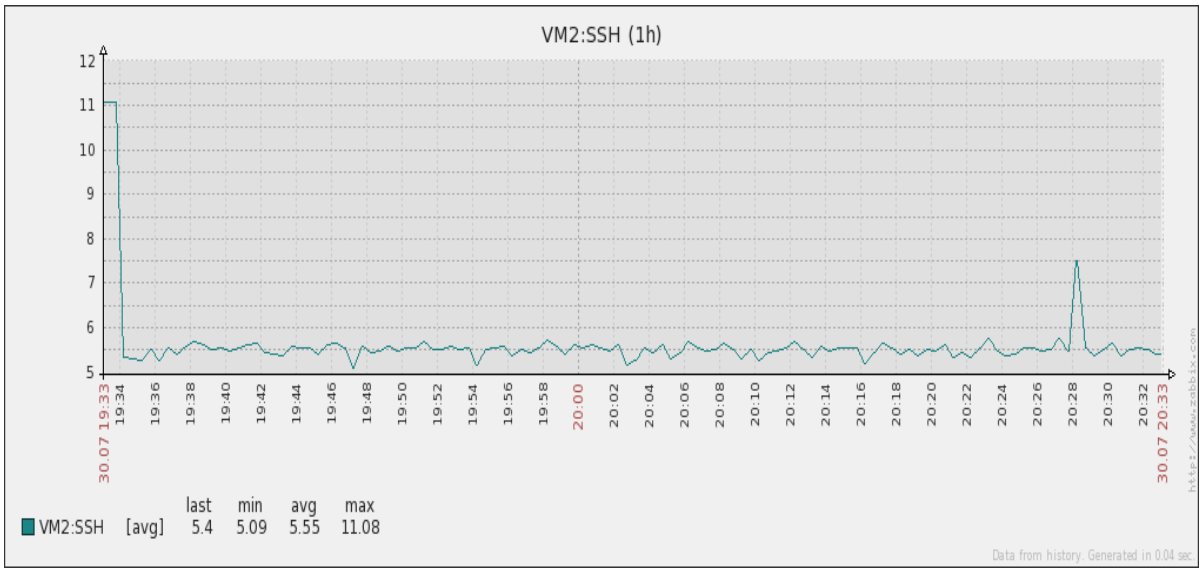

Figura 12 - Monitoramento do SSH da MV2.
Fonte: Própria.

6.3.2 Resultados do Monitoramento das Máquinas Físicas

Os gráficos a seguir correspondem aos resultados relacionados ao monitoramento do *Frontend* e do Nó controlador. Diferente das máquinas virtuais que foram monitoradas com o *Simple Check* foi instalado o Agente Zabbix nas duas máquinas, pois são máquinas físicas e encontram-se trabalhando na nuvem. É importante salientar que foram feitos testes de sobrecarga nas MV’s, esse teste é feito através de um código que emite um *loop* de requisições de um arquivo txt infinito (detalhes estão no APÊNDICE D), estes testes foram feitos para analisar as seguintes métricas do *Frontend* e do Nó Controlador: CPU (*Central Processing Unit* - Unidade Central de Processamento) Utilizada e Carga da CPU. O intuito é entender como essa nuvem iria se comportar com uma sobrecarga nas máquinas virtuais, ou seja, como os demais componentes seriam afetados.

No monitoramento da memória do *Frontend* na Figura 13, percebeu-se que há a utilização de toda a memória do mesmo. Isso acontece por que os números de recursos sendo utilizados nessa máquina são tantos que acabam ocupando toda a memória.

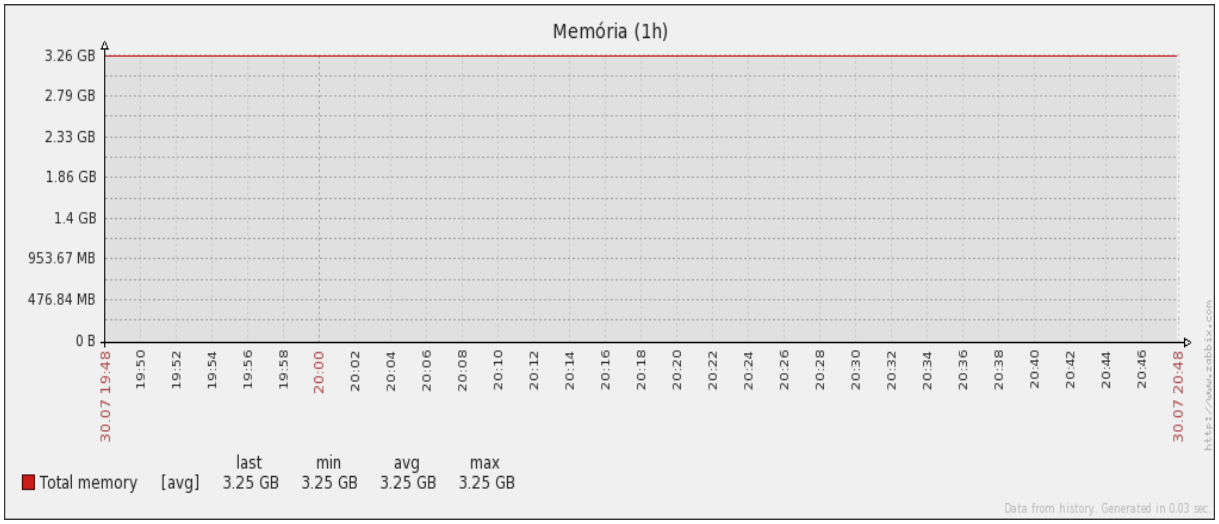

Figura 13 - Utilização da Memória do *Frontend*.
Fonte: Própria.

Na Figura 14 do Nó Controlador, é possível perceber que a memória oscila bastante, demonstrando que não está sendo toda utilizada, isso acontece porque nesse momento não existe muitas atividades em execução.

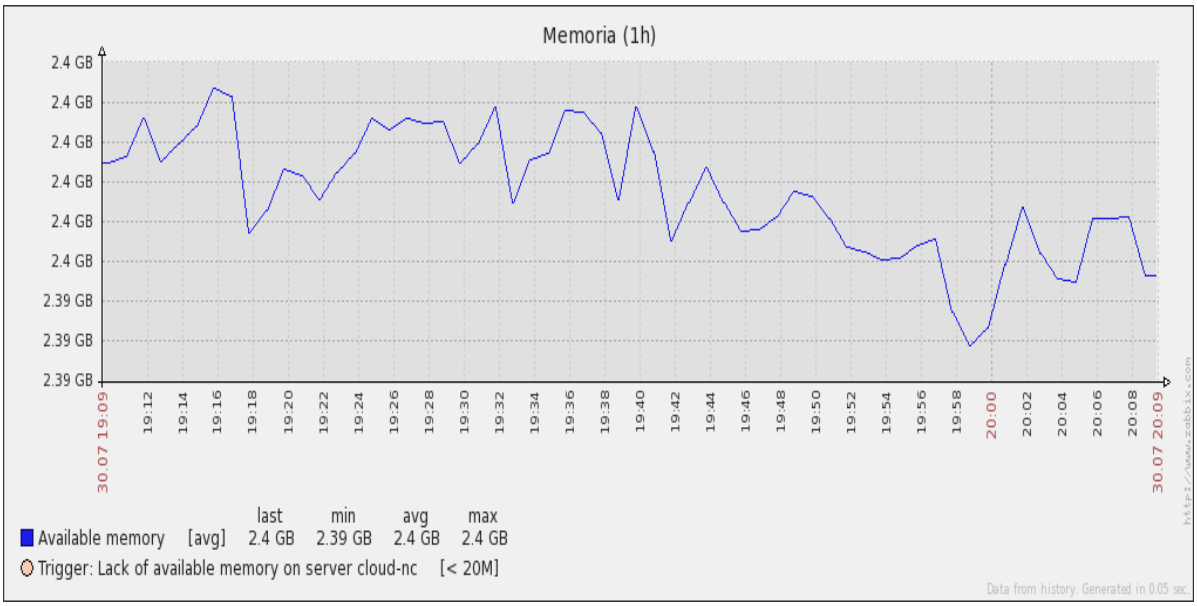

Figura 14 - Utilização da Memória do Nó controlador.
Fonte: Própria.

Na métrica de CPU monitorada no *Frontend*, na Figura 15, percebeu-se que antes do teste de sobrecarga, a utilização da CPU estava menos de 20%, estava normal.

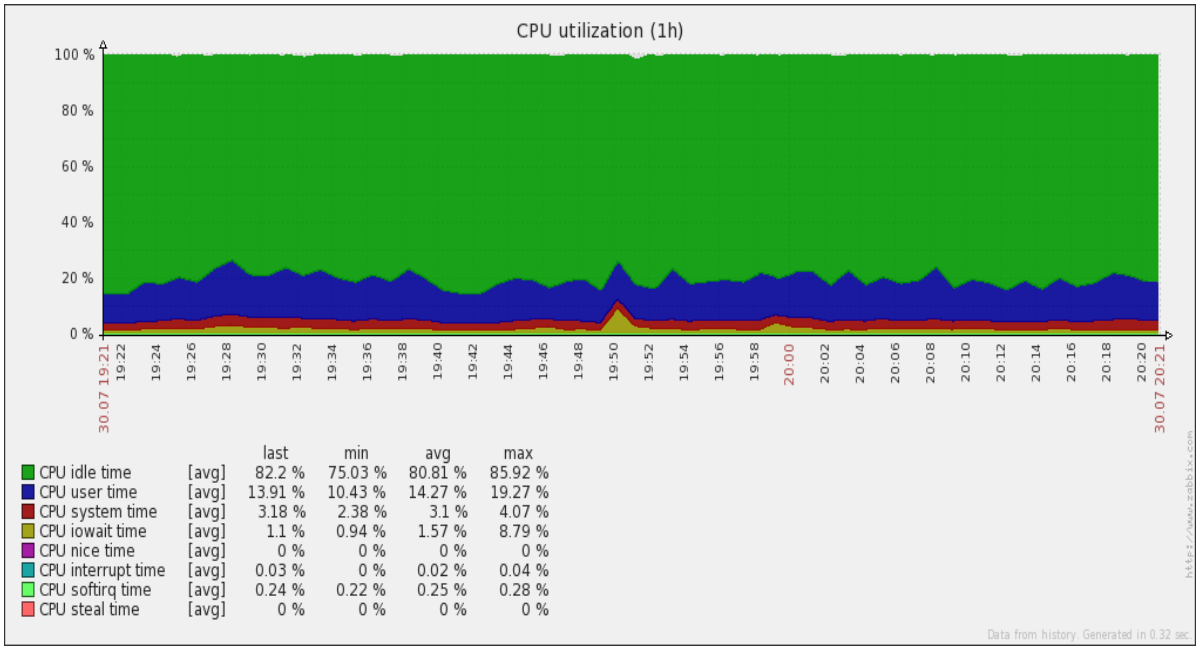

Figura 15 - Utilização da CPU na maquina *Frontend*
Fonte: Própria.

Porém, foi feito um teste de sobrecarga nas MV's, como é observado na Figura 16, para entender como o *Frontend* se comporta em nuvem. E o resultado foi que tudo que acontece nas máquinas virtuais afeta diretamente a performance de toda a nuvem.

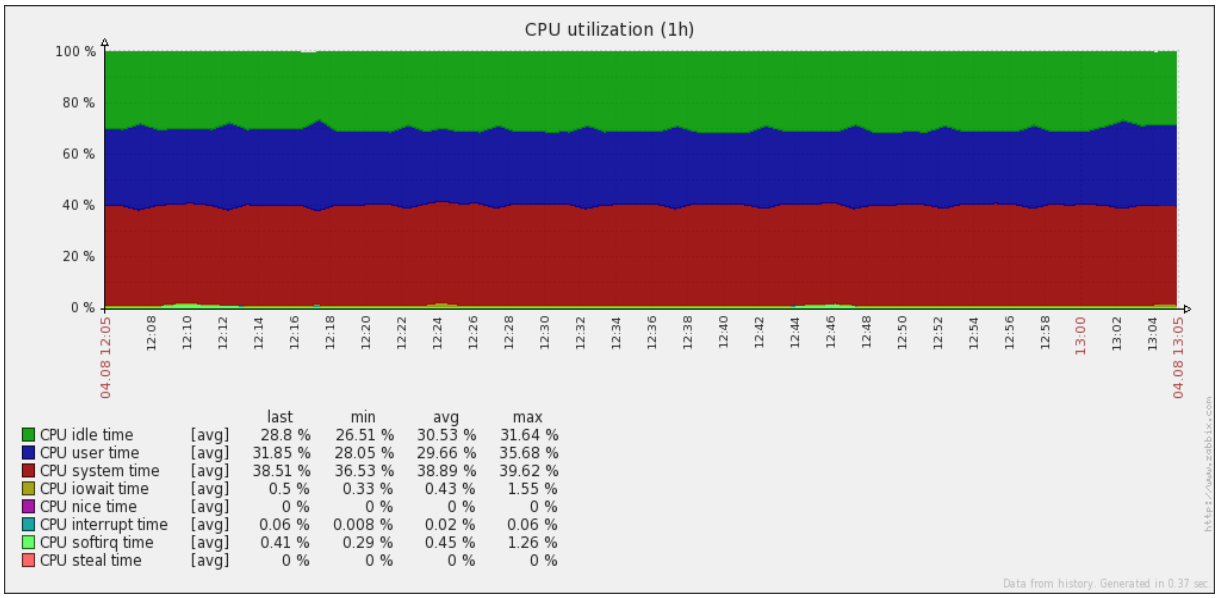

Figura 16 - Teste de sobrecarga no *Frontend.*
Fonte: Própria.

Antes do teste de sobrecarga, como pode-se observar na Figura 17, o Nó Controlador, mesmo com as MV's instanciadas, não demonstrava mudança no gráfico, ele permanecia estável.

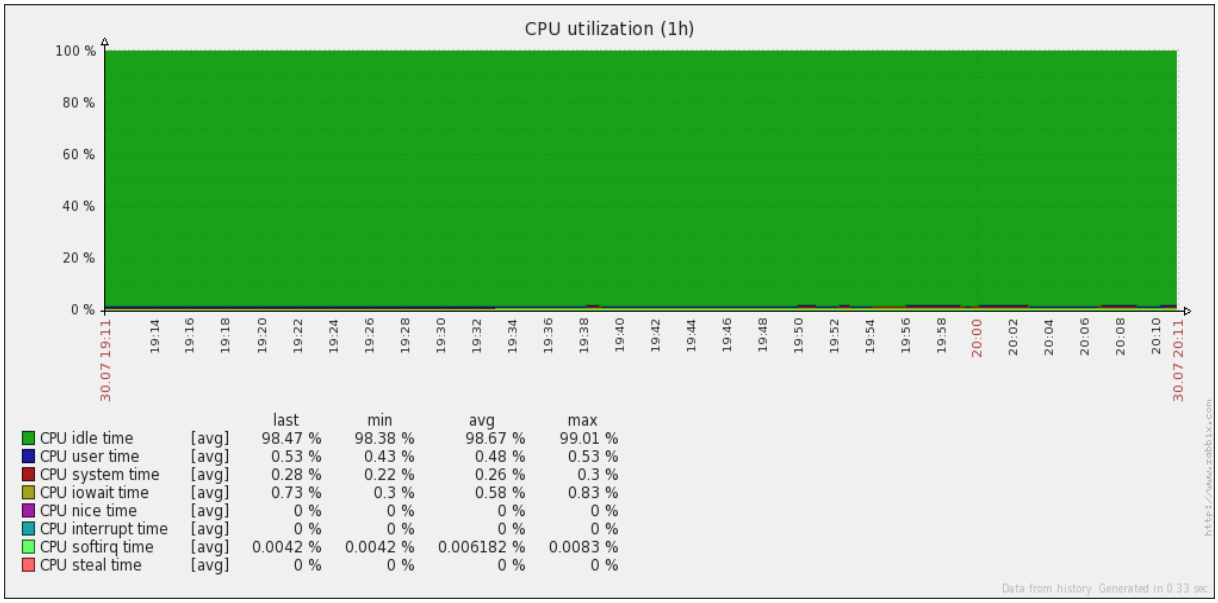

Figura 17 - Verificação da CPU utilizada antes dos testes de sobrecarga do Nó controlador.
Fonte: Própria.

Percebeu-se o aumento significante no uso da CPU do Nó Controlador, conforme a Figura 18, assim que iniciou os testes. Isso ocorreu porque é no Nó Controlador que estão às máquinas virtuais.

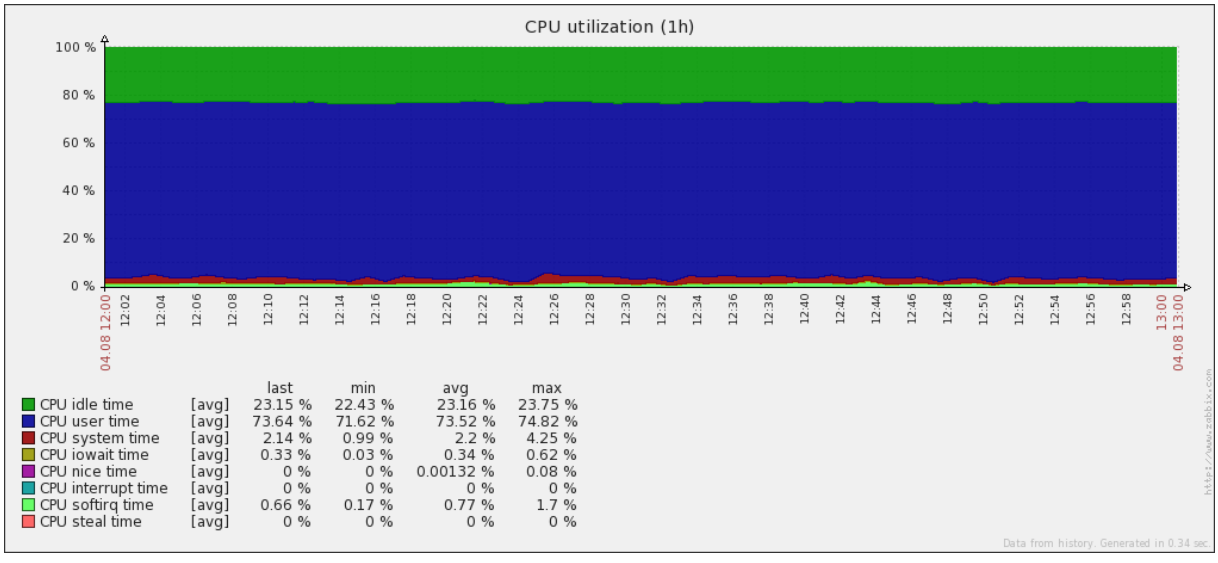

Figura 18 - Estado da CPU durante os testes de sobrecarga.
Fonte: Própria.

Na carga de CPU do *Frontend*, conforme a Figura 19, inicialmente pode-se notar que não havia muitas mudanças nos processos (troca de dados, utilização de serviços e etc) executados no mesmo. Apesar do momento de pico as 19:00.

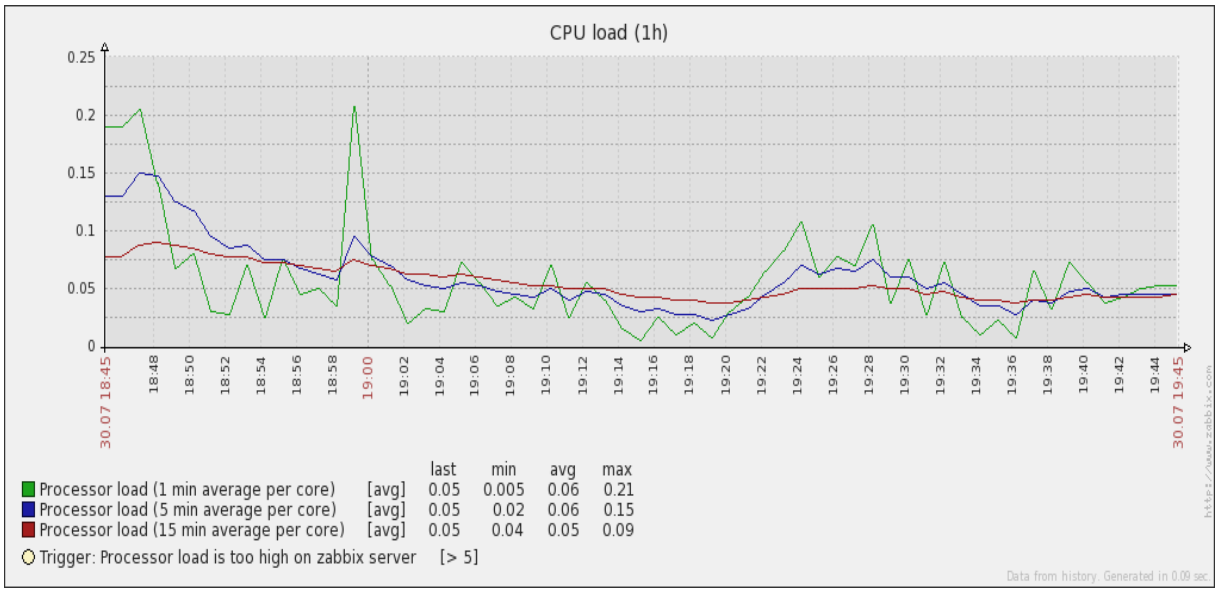

Figura 19 - Carga de CPU do *Frontend*.
Fonte: Própria.

Durante os teste de sobrecarga houve vários estados de "pico" no sistema conforme Figura 20, esses momentos ocorreram as 12:18, 12:42 e as 13:05 o que mostra o seu constante funcionamento, diferente do outro acima que demonstrou queda, no trabalho da CPU em alguns momentos.

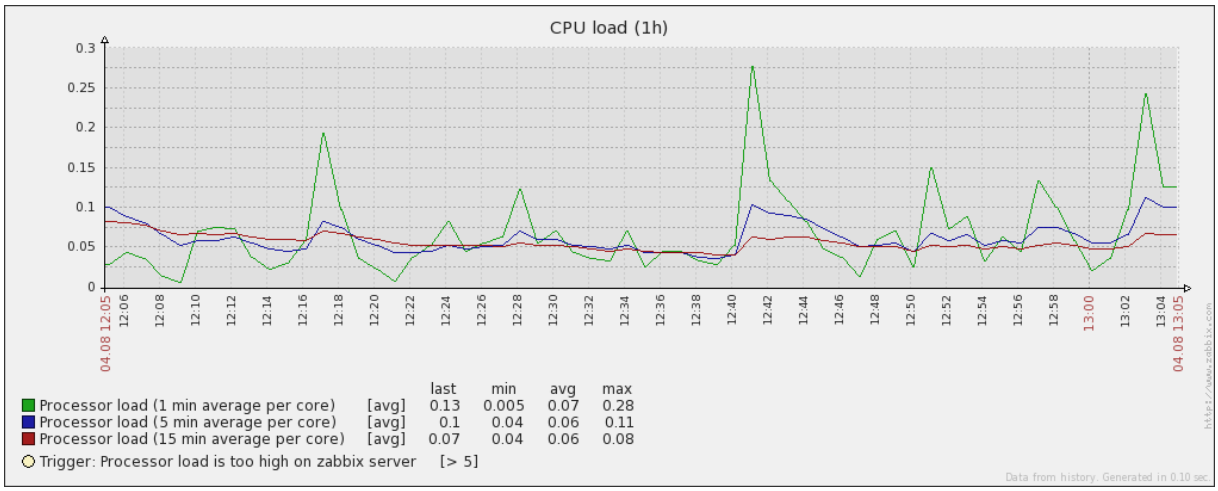

Figura 20 - Teste de sobrecarga da CPU do *Frontend*.
Fonte: Própria.

Na Figura 21 do Nó Controlador, a CPU não obteve muito trabalho. Em alguns momentos ela obteve alguns picos devido à manipulação das MV's.

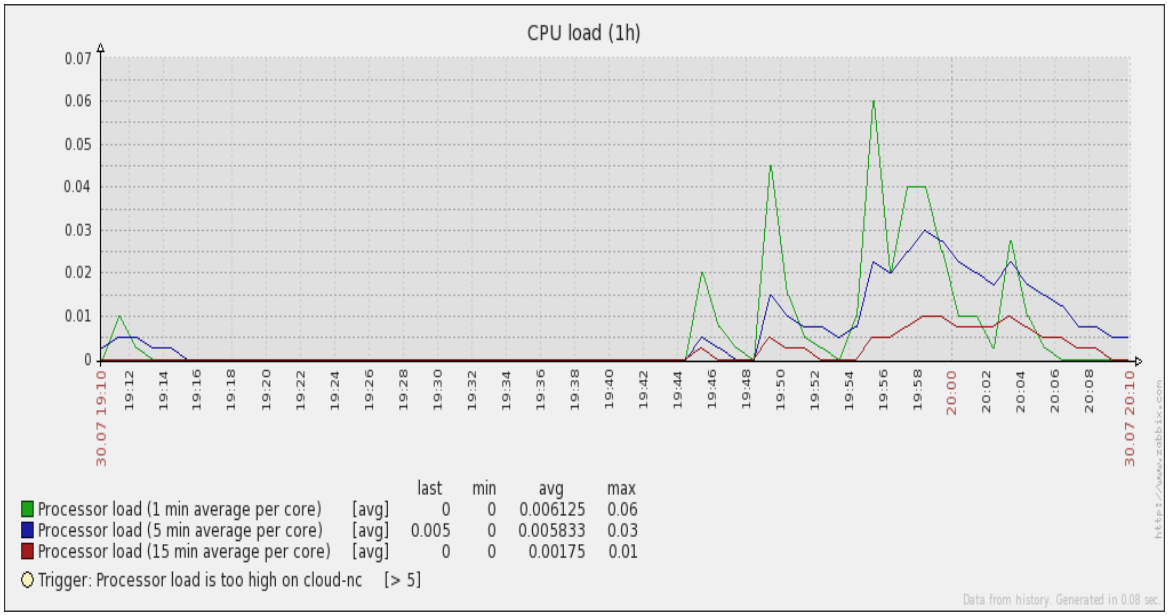

Figura 21 - CPU utilizada no Nó controlador.
Fonte: Própria.

Porém, no momento dos testes, o trabalho da CPU aumentou cerca de 80% em relação a carga da CPU do gráfico da Figura 21. Esse aumento é indicado pelo gráfico abaixo no qual 0,8 significa 80%.

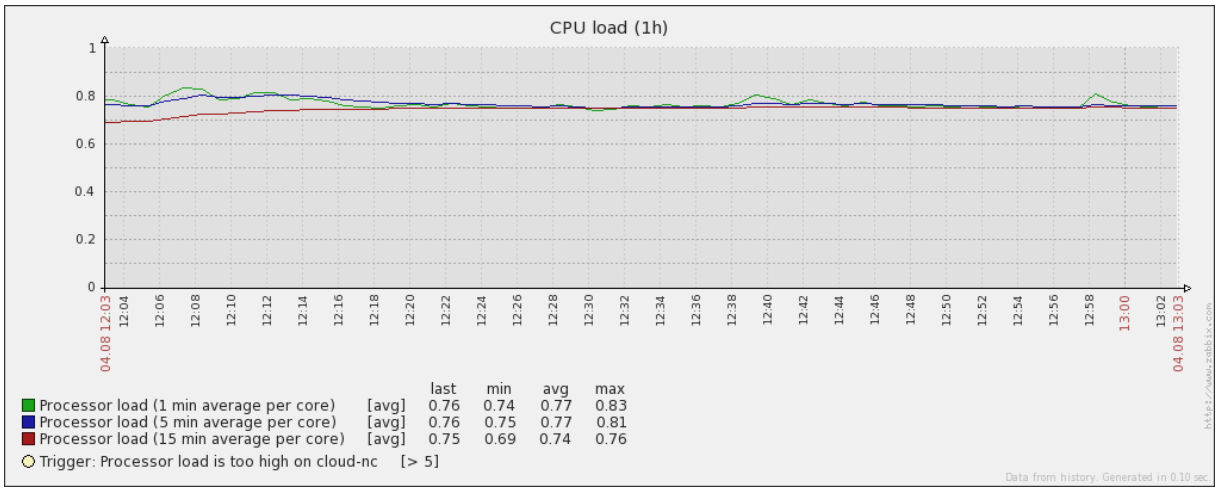

Figura 22 - Teste de sobrecarga da CPU no Nó controlador.
Fonte: Própria.

Na Figura 23, os dados foram coletados antes dos testes de sobrecarga, entretanto não foram feito teste de sobrecarga no Número de Processos funcionando do *Frontend*. Porém percebe-se que o numero de processos ocorre com poucos intervalos, e isso denota a troca de informações constantes.

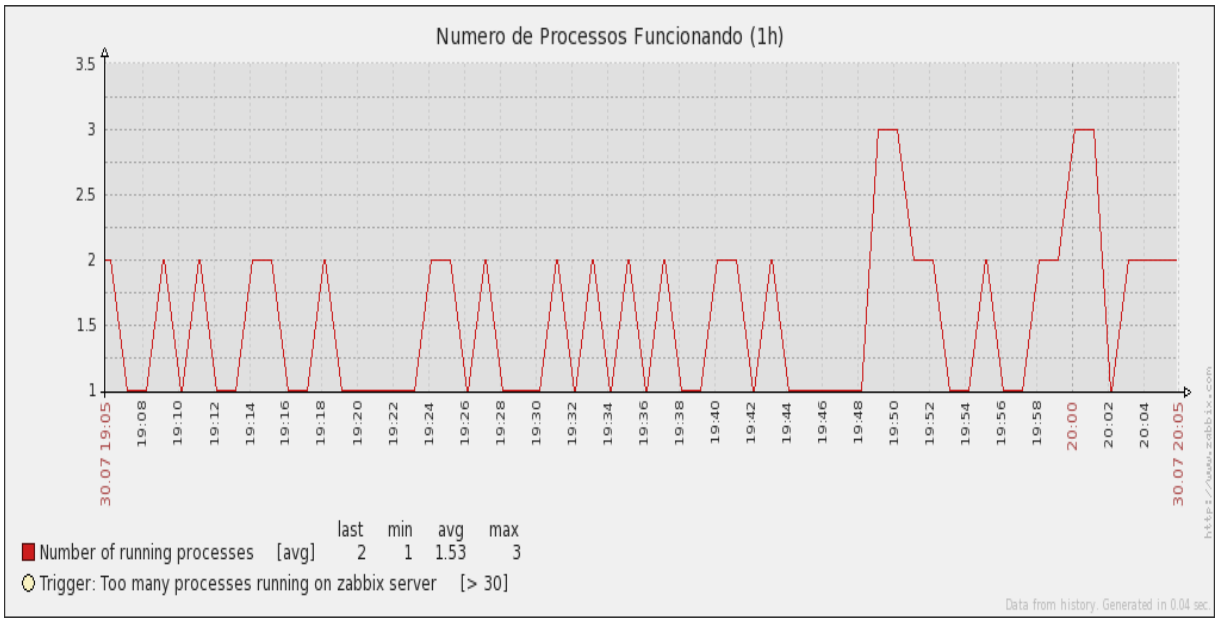

Figura 23 - Número de Processos ativos do *Frontend*.
Fonte: Própria.

Conforme a Figura 24, no Nó controlador, ouve somente um momento de pico as 10:52. Isso ocorreu porque nesse momento as máquinas virtuais estavam funcionando mas não foi manipulado nenhuma função delas que exigisse muita requisição de processos.

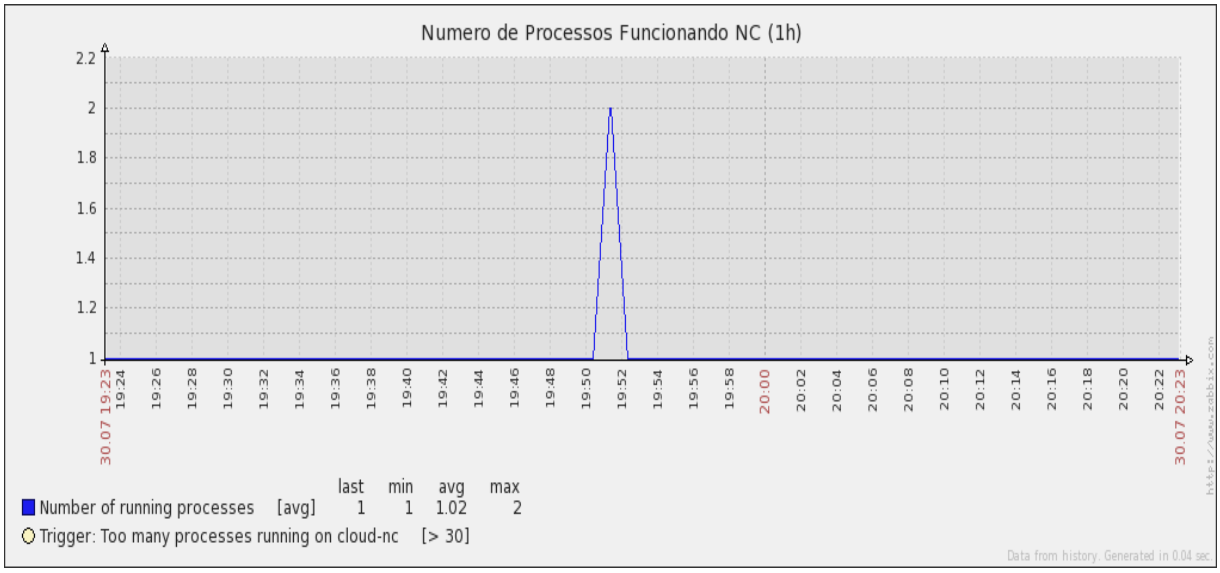

Figura 24 - Número de Processos ativos do Nó controlador.
Fonte: Própria.

6.4 Resumo do Capítulo

Nesse capítulo foram apresentados os recursos de hardware e software utilizados, o cenário do experimento e a arquitetura proposta para o monitoramento simples de uma nuvem privada utilizando o *famework* Zabbix. Por fim foram detalhados os resultados obtidos com este estudo de caso. No próximo capítulo será apresentada a análise conclusiva.

7 CONSIDERAÇÕES FINAIS

Este projeto implementou, configurou e adaptou uma nuvem privada com ferramentas gratuitas e de código fonte aberto.

Foram monitorados o desempenho e disponibilidade das máquinas virtuais e físicas, através de um cenário de testes, no qual foi escolhido o *framework* de monitoramento Zabbix para monitorar este cenário.

É importante salientar que foram várias as problemáticas enfrentadas para que a nuvem fosse implementada, a primeira foi a mudança da infraestrutura da sala de pesquisa, no qual estava instalada a primeira nuvem privada que tinha como base o *Ubuntu Server Cloud 11.04*. Visto que as infraestruturas de nuvem precisam de IP estático, ocorreu que a rede da sala de pesquisa mudou os IP's para dinâmico. Dessa forma, a infraestrutura foi refeita, mas como a Canonical retira as atualizações dos seus sistemas operacionais antigos de seus servidores, ficou impossível concluir a infraestrutura de nuvem de forma concisa e sem erros. Assim, fez-se a mudança para o Eucalyptus *FastStart*, que orquestrava as mesmas vantagens do *Ubuntu Server Cloud 11.04*, porém em uma interface gráfica, simples e de fácil entendimento.

Quanto às dificuldades relacionadas ao monitoramento, vale ressaltar que em um primeiro momento foi utilizado o agente Zabbix para monitorar as MV's, porém como não houve comunicação entre as camadas de integração e de visualização, foi utilizado o recurso do Zabbix chamado de *Simple Check* para monitorar a disponibilidades das instancias virtuais através dos testes de conectividade e acesso remoto com as métricas PING e SSH.

Percebeu-se no decorrer do trabalho a veracidade da segurança da nuvem privada. No momento em que as MV's foram monitoradas, por um determinado tempo foi impossível acessá-las, pois a nuvem emitia avisos de segurança, e trocou as chaves que davam acesso as máquinas virtuais para evitar possíveis invasões ao sistema.

Foi possível notar que a computação em nuvem se vale de outros conceitos e serviços, ou seja, podemos fazer adaptações como foi no caso do Zabbix para monitorar a nuvem, porém a nuvem privada requer um esforço maior para ser monitorada.

Este trabalho é um primeiro passo para que este projeto possa ser ainda mais explorado, é proposto que futuramente possamos utilizar o framework Zabbix para monitorar outras métricas das maquinas virtuais, como: CPU, memória, além de simular o

comportamento de servidor web, ftp e de e-mail, e também a possibilidade de tornar está nuvem escalável, ou seja, que possamos atribuir mais Nós Controladores a sua estrutura.

8 REFERÊNCIAS BIBLIOGRÁFICAS

BENÍTEZ, C. K. IBERO; DILL, M.R et al. EXEHDA-RM: Um Serviço para Monitoramento Autônomo de Recursos no Middleware EXEHDA**.** In: ESCOLA REGIONAL DE AUTO DESEMPENHO, 4 . 2011, Porto Alegre. Anais... Porto Alegre, UFRGS 2011. P 139-140.

BLACK, Tomas Lovis. Comparação de Ferramentas de Gerenciamento de Redes. Porto Alegre, 2008. 64 f. Trabalho de Conclusão apresentado como requisito parcial para a obtenção de grau de especialista (Especialização em tecnologia gerencia e segurança de redes de computadores), Instituto de Informática, Universidade Federal do Rio Grande do Sul, 2008.

CANEDO, Edna Dias. Modelo de Confiança para a Troca de Arquivos em uma Nuvem Privada**.** Brasília, 2012. 107 f. Tese (Doutorado em Engenharia Elétrica) Departamento de Engenharia Elétrica, Universidade de Brasília, 2012.

CARISSIMI, Alexandre. Virtualização: da teoria a soluções**.** In: 26º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC) 2008, Porto Alegre, Livro texto dos Minicursos. Porto Alegre, 2008, v. 1, p. 173-207.

CARVALHO, Márcio Barbosa de. Adaptação da Ferramenta Nagios para Monitoramento de Servidores Virtuais. Porto Alegre, 2010. 39 f. Trabalho de Conclusão de Curso (Graduação em Bacharelado em Ciência da Computação), Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2010.

CENTOS (Org). Disponível em: <http:// www.centos.org>. Acesso em: 27 Jul. 2013.

CHAVES, Shirlei Aparecida. Arquitetura e Sistema de Monitoramento para Computação em Nuvem Privada. Florianópolis, 2010. 114 f. Dissertação (Mestrado em Ciência da Computação) Programa de Pós-graduação em Ciências da Computação, Universidade Federal de Santa Catarina, Florianópolis, 2010.

CHAVES, Shirlei Aparecida; URIARTE, Rafael Brundo; WESTPHALL, Carlos Becker. Implantando e Monitorando uma Nuvem Privada. In: VII Workshop em Clouds, Grids e Aplicações. 2010, Gramado, Anais... Gramado: UFRGS, 2010. p 31- 42.

CONVERGÊNCIA DIGITAL. Rio de Janeiro: Convergência digital, 2013.

D.A. MENASCÉ, V.A.F. Almeida. Capacity Planning for Web Services: Metrics, Models ,and Methods, Prentice Hall, 2002.

DEBIAN. Disponível em: <http://www.debian.org/index.pt.html>. Acesso em: 29 Jul. 2013.

ELMROTH, Erik; LARSSON, Lars. Interfaces for Placement, Migration, and Monitoring of Virtual Machines in Federated Clouds. In: INTERNATIONAL CONFERENCE ON GRID AND COOPERATIVE COMPUTING, 8, 2009, Lanzhou, Gansu, China. Proceedings... Washington, DC, EUA: IEEE Computer Society, 2009. p. 253 - 260. EUCALYPTUS (Org.). The Open Source Cloud Platform. v. 1.6.1.

EUCALYPTUS. Disponível em: <http://www.eucalyptus.com/eucalyptus-cloud/get-started/try/faststart>. Acesso em: 03 Ago. 2013.

FERREIRA, André Osório de Castro. IPBrick-Controle e Monitorização do Parque Informático. Porto, 2008. 92 f. Dissertação (Mestrado Integrado em Engenharia Eletrotécnica e de Computadores) Major Telecomunicações, Faculdade de Engenharia, Universidade do Porto, Portugal, 2011.

GONÇALVES, Bruno Ortale. Gerência e Monitoramento de uma Nuvem Privada. Florianópolis, 2011. 57 f. Trabalho de Conclusão de Curso (Graduação em Bacharelado em Ciência da Computação), Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis, 2011.

J. SHAO, Q. Wang, A performance guarantee e approach for cloud Applications based on monitoring, in: Computer Software and Applications Conference Workshops (COMPSACW),2011 IEEE35th Annual,2011, p.25–30.

KVM. Disponível em: <http://www.linux-kvm/page/Main_page>. Acesso em: 27 Jul. 2013.

LIMA, Paulo. As mídias estão com os dias contados?. São Paulo, Set. 2011. Disponível em: <http://www.mundodastribos.com/as-midias-estao-com-os-dias-contados.html> Acesso em: 03 Mar. 2013.

MELL, Peter; GRANCE, Timothy. The NIST Definition of cloud computing. Estados Unidos, 2011. Disponível em:<http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf>. Acesso em: 15 Out. 2012.

MICROSOFT (Org). Disponível em: <http://msdn.microsoft.com/pt-br/library/office/fp161226.aspx> .Acesso em: 29 Jul. 2013.

NIMBUS (Org). Disponível em: < http://www.trynimbus.com>. Acesso em: 29 Jul. 2013.

NIST - National Institute of Standards and Technology. DRAFT Cloud Computing Synopsis and Recommendations**.** Estados Unidos**,** 2011. Disponível em: <http://csrc.nist.gov/publications/drafts/800-146/Draft-NIST-SP800-146.pdf>. Acesso em: 04 Jan. 2013.

OPENNEBULA (Org). About OpenNebula Disponível em:<http://opennebula.org/about:about>. Acesso em 25 Jul. 2013.

OPENSTACK (Org). Software. Disponível em: <http://www.openstack.org/>. Acesso em 15 Jul. 2013.

RATIONAL MODELER IBM (Org). Disponível em: <http://www.ibm.com/developerworks>>. Acesso em: 29 Jul. 2013.

RHODEN, Guilherme Eliseu. Detecção de intrusões em Backbones de redes de computadores através da analise de comportamento de SNMP. Florianópolis, 2002. 101 f. Dissertação (Mestrado Ciência da Computação). Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de Santa Catarina, 2002.

SOBRAGI, Cyro Gudolle. Adoção de computação em nuvem: Estudos de casos múltiplos. Porto Alegre, 2012. 155 f. Dissertação (Mestrado em Administração) Programa de Pós-graduação em Administração, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2012.

SOUSA, Flávio R. C.; MOREIRA, Leandro O; MACHADO, Javam C. Computação em Nuvem: Conceitos, Tecnologias, Aplicações e Desafios. In: ERCEMAPI, 30. 2009, Parnaíba. Anais... Parnaíba: UESPI. Disponível em: <http://files.0fx66.com/paper/Computacao_Nuvem.pdf >. Acessado em: 25 fev 2013.

SOUSA, Flávio, R.C; MOREIRA, Leonardo, O; MACHADO, Javam, C; 2011. Computação em Nuvem Autônoma: Oportunidades e Desafios. XXIX Simpósio Brasileiro de Rede e Sistemas Distribuídos, 2011, Campo Grande, Anais, UFMS, 2011. Disponível em:<sbrc2011.facom.ufms.br/files/workshops/wosida/ST01_2.pdf>. Acessado em: 20 Jan 2012.

STARUML (Org). Disponível em:<http://www.staruml.sourceforget.net/en/>. Acesso em: 29 Jul. 2013.

TAURION, Cezar. Cloud Computing-Computação em Nuvem-Transformando o Mundo da Tecnologia de Informação. Rio de Janeiro: Editora Brasport, 2009.

UBUNTU. Disponível em: <http://www.ubuntu.com>. Acesso em: 29 Jul. 2013.

URIARTE, Rafael Brundo. Um Arcabouço de Monitoramento e Autoproteção para Nuvens Privadas**.** Florianópolis, 2012. 107 f. Dissertação (Mestrado em Ciência da Computação) Programa de Pós-graduação em Ciências da Computação, Universidade Federal de Santa Catarina, Florianópolis, 2012.

VAQUERO, Luis M; RODERO-MERINO, Luis; CACERES, Juan; LINDNER Maik. A Break in the Clouds: Towards a Cloud Definition. ACM Sigcomm Computer Communication Review, Volume 39, numero 1, Jan. 2009.

VELTE, Anthony, T; VELTE, Toby, J; ELSENPETER, Robert; Cloud Computing, A Practical Approach. Índia, Ed. McGraw-Hill Education, 2010.

VERAS, Manoel. Cloud Computing Nova Arquitetura de TI. Rio de Janeiro, Editora Brasport, 2012.

VERAS, Manoel. Virtualização: Componente Central do Datacenter. Rio de Janeiro, Editora Brasport, 2011.

VITTI, Pedro Artur Figueiredo. Integração do PCMONS com o OpenNebula para Gerência e Monitoramento de Nuvens Privadas. Florianópolis, 2012. 95 f. Trabalho de Conclusão de Curso (Graduação em Bacharelado em Ciência da Computação), Departamento de Informática e Estatística, Universidade Federal de Santa Catarina, Florianópolis, 2012.

XEN (Org.). What is Xen Hypervisor? Disponível em: <http://www.xen.org/files/Marketing/WhatisXen.pdf>. Acesso em: 15 Mai. 2013.

ZABBIX (Org). What is New? Disponível em: <http://www.zabbix.com/>. Acesso em: 25 Jul. 2013.

APÊNDICE A – Definição da faixa de IP que o Eucalyptus utiliza para atribuir as MV

```
################################################################################
# GLOBAL CONFIGURATION
################################################################################

# Where Eucalyptus is installed
EUCALYPTUS="/"

# This is the username that you would like eucalyptus to run as
EUCA_USER="eucalyptus"

# Extra options to pass to the eucalyptus-cloud process, such as log
# levels, heap size, or other JVM flags.
CLOUD_OPTS=""

################################################################################
# STORAGE CONTROLLER (SC) CONFIGURATION
################################################################################

# The number of loop devices to make available at SC startup time.
# The default is 256.  If you supply "max_loop" to the loop driver
# then this setting must be equal to that number.
#CREATE_SC_LOOP_DEVICES=256

################################################################################
# CLUSTER CONTROLLER (CC) / NODE CONTROLLER (NC) SHARED
CONFIGURATION
################################################################################

# The level of logging output.  Valid settings are, in descending order of
# verbosity: EXTREME, TRACE, DEBUG, INFO, WARN, ERROR, and FATAL.  The
# default is INFO.
#LOGLEVEL="INFO"

# The number of old log files to keep when rotating logs, in range [0-999].
# The default is 10. When set to 0, no rotation is performed and log size
# limit is (LOGMAXSIZE, below) is not enforced.
#LOGROLLNUMBER="10"

# The maximum size of the log file, in bytes. 100MB by default. For this
```

```
# size to be enforced, LOGROLLNUMBER, above, must be 1 or higher. If log
# rotation is performed by an outside tool, either set LOGROLLNUMBER to 0
# or set this limit to a large value.
#LOGMAXSIZE=104857600

# On a NC, this defines the TCP port on which the NC will listen.
# On a CC, this defines the TCP port on which the CC will contact NCs.
NC_PORT="8775"

##############################################################################
# CLUSTER CONTROLLER (CC) CONFIGURATION
##############################################################################

# The TCP port on which the CC will listen.
CC_PORT="8774"

# The scheduling policy that the CC uses to choose the NC on which to
# run each new instance.  Valid settings include GREEDY and ROUNDROBIN.
# The default scheduling policy is ROUNDROBIN.
SCHEDPOLICY="ROUNDROBIN"

# A space-separated list of IP addresses for all the NCs that this CC
# should communicate with.  The ``euca_conf --register-nodes" command
# manipulates this setting.
NODES="192.168.0.13"

# When multiple CCs reside in the same layer 2 broadcast domain, change
# this setting to "Y" to disable tunneling.  This setting has no effect
# in Static or System modes.
#DISABLE_TUNNELING="N"

# The location of the NC service.  The default is
# axis2/services/EucalyptusNC
NC_SERVICE="axis2/services/EucalyptusNC"

# Set this to make the CC cache images, kernels and ramdisks.  NCs must
# be able to reach the CC with the specified value.
#CC_IMAGE_PROXY="ip_of_cc"

# Set this to the location where the CC image proxy should store cached
# images.  The default is /var/lib/eucalyptus/dynserv/
#CC_IMAGE_PROXY_PATH="/var/lib/eucalyptus/dynserv/"

# Set this to the maximum size (in megabytes) of the CC image proxy cache.
```

```
# The default is 32768, or 32 gigabytes.
#CC_IMAGE_PROXY_CACHE_SIZE="32768"
################################################################################
# NODE CONTROLLER (NC) CONFIGURATION
################################################################################

# The hypervisor that the NC will interact with in order to manage
# virtual machines.  Supported values include "kvm" and "xen".
HYPERVISOR="kvm"

# The following three options determine whether KVM uses Virtio for
# specific types of I/O with instances.  These options only affect the
# KVM hypervisor.

# If "1", use Virtio for the root file system
USE_VIRTIO_ROOT="1"

# If "1", use Virtio for dynamic block volumes
USE_VIRTIO_DISK="1"

# If "1", use Virtio for the network card
USE_VIRTIO_NET="1"

# The amount of memory, in megabytes, that Eucalyptus is allowed to
# allocate to instances running on this system.  The default value of
# 0 allows Eucalyptus to use all available memory for instances.
#MAX_MEM="0"

# The number of virtual CPU cores that Eucalyptus is allowed to allocate
# to instances.  The default value of 0 allows Eucalyptus to use all
# CPU cores on the system.
#MAX_CORES="0"

# The amount of disk space, in megabytes, that the NC is allowed to use
# in its work directory ($INSTANCE_PATH/eucalyptus/work).  By default
# the NC chooses automatically.  Values below 10 are ignored.
#NC_WORK_SIZE=50000

# The amount of disk space, in megabytes, that the NC is allowed to use in
# its image cache directory ($INSTANCE_PATH/eucalyptus/cache).  By default
# the NC chooses automatically.  A value below 10 will disable caching.
#NC_CACHE_SIZE=50000
# The number of disk-intensive operations that the NC is allowed to
# perform at once.  A value of 1 serializes all disk-intensive operations.
```

```
# The default value is 4.
#CONCURRENT_DISK_OPS=4

# By default, a NC attempts to write the SSH public key associated to
# the instance's filesystem before the instance starts.  A value of 1
# disables this behavior.
#DISABLE_KEY_INJECTION="0"

# The number of loop devices to make available at NC startup time.
# The default is 256.  If you supply "max_loop" to the loop driver then
# this setting must be equal to that number.
#CREATE_NC_LOOP_DEVICES=256

# The directory where the NC will store instances' root filesystems,
# ephemeral storage, and cached copies of images.
INSTANCE_PATH="/var/lib/eucalyptus/instances"

# If euca-bundle-upload, euca-check-bucket, or euca-delete-bundle do
# not appear in the NC's search PATH then specify their locations here.
#NC_BUNDLE_UPLOAD_PATH="/usr/bin/euca-bundle-upload"
#NC_CHECK_BUCKET_PATH="/usr/bin/euca-check-bucket"
#NC_DELETE_BUNDLE_PATH="/usr/bin/euca-delete-bundle"

# The maximum amount of time, in seconds, that an instance will remain
# in a migration-ready state on a source NC while awaiting the
# preparation of a destination NC for a migration. After this time
# period, the migration request will be terminated and the any
# preparation on the source NC will be rolled back. Default is 15
# minutes.
#NC_MIGRATION_READY_THRESHOLD=900

##############################################################################
# NETWORKING CONFIGURATION
#
# The set of networking settings that apply to a cloud varies based on
# its networking mode.  Each setting in this section lists the modes in
# which it applies.  Unless otherwise noted, all of these settings apply
# only to CCs.  All settings that lack default values must be specified
# in the networking modes that use them.
##############################################################################

# The networking mode in which to run.  The same mode must be specified
# on all CCs and NCs in the entire cloud.  Valid values include SYSTEM,
# STATIC, MANAGED, and MANAGED-NOVLAN.
```

```
VNET_MODE="MANAGED-NOVLAN"

# The name of the network interface that is on the same network as
# the NCs. The default is "eth0".
# Networking modes: Static, Managed, Managed (No VLAN)
VNET_PRIVINTERFACE="eth0"

# On a CC, this is the name of the network interface that is connected
# to the "public" network.  When tunnelling is enabled, this must be
# a bridge.  The default is "eth0".
# Networking modes: Managed, Managed (No VLAN)
#
# On an NC, this is the name of the network interface that is connected
# to the same network as the CC.  The default is "eth0".
# Networking modes: Managed
VNET_PUBINTERFACE="eth0"

# On an NC, this is the name of the bridge interface to which instances'
# network interfaces should attach.  A physical interface that can reach
# the CC must be attached to this bridge.
# Networking modes: System, Static, Managed (No VLAN)
VNET_BRIDGE="br0"

# A map of MAC addresses to IP addresses that Eucalyptus should allocate
# to instances when running in Static mode.  Separate MAC addresses and
# IP addresses with '=' characters.  Separate pairs with spaces.
# Networking modes: Static
#VNET_MACMAP="AA:DD:11:CE:FF:ED=192.168.1.2
AA:DD:11:CE:FF:EE=192.168.1.3"

# A space-separated list of individual and/or hyphenated ranges of public
# IP addresses to assign to instances.
# Networking modes: Managed, Managed (No VLAN)
#VNET_PUBLICIPS="your-free-public-ip-1 your-free-public-ip-2 ..."
# The address and network mask of the network the cloud should use for
# instances' private IP addresses.
# Networking modes: Static, Managed, Managed (No VLAN)
#VNET_SUBNET="192.168.0.0"
#VNET_NETMASK="255.255.0.0"

# The number of IP addresses to allocate to each security group.
# Specify a power of 2 between 16 and 2048.
# Networking modes: Managed, Managed (No VLAN)
#VNET_ADDRSPERNET="32"
```

```
# The address of the DNS server to supply to instances in DHCP responses.
# Networking modes: Static, Managed, Managed (No VLAN)
#VNET_DNS="your-dns-server-ip"

# The search domain to supply to instance in DHCP responses.
# NOTE: This should always be cloud.vmstate.instance_subdomain + ".internal",
# and may be overridden by the CLC property in a future release. See EUCA-4226
# Networking modes: Static, Managed, Managed (No VLAN)
#VNET_DOMAINNAME="eucalyptus.internal"

# The network broadcast address and default gateway to supply to instances
# in DHCP responses.
# Networking modes: Static
#VNET_BROADCAST="192.168.1.255"
#VNET_ROUTER="192.168.1.1"

# Set this to the IP address that other CCs can use to reach this CC
# if layer 2 tunneling between CCs does not work.  It is not normally
# necessary to change this setting.
# Networking modes: Managed, Managed (No VLAN)
#VNET_LOCALIP="your-public-interface's-ip"

# The ISC DHCP server executable to use.  The default is
# "/usr/sbin/dhcpd3".
# Networking modes: Static, Managed, Managed (No VLAN)
VNET_DHCPDAEMON="/usr/sbin/dhcpd41"

# The user as which the DHCP daemon runs on your distribution.
# The default is "dhcpd".
# Networking modes: Static, Managed, Managed (No VLAN)
#VNET_DHCPUSER="dhcpd"
VNET_PUBLICIPS="192.168.0.15-192.168.0.30"
VNET_SUBNET="172.31.254.0"
VNET_DNS="192.168.0.254"
VNET_NETMASK="255.255.254.0"
VNET_ADDRSPERNET="32"
BAL CONFIGURATION
##############################################################################

# Where Eucalyptus is installed
EUCALYPTUS="/"

# This is the username that you would like eucalyptus to run as
```

```
EUCA_USER="eucalyptus"

# Extra options to pass to the eucalyptus-cloud process, such as log
# levels, heap size, or other JVM flags.
CLOUD_OPTS=""
#############################################################################
# STORAGE CONTROLLER (SC) CONFIGURATION
#############################################################################

# The number of loop devices to make available at SC startup time.
# The default is 256.  If you supply "max_loop" to the loop driver
# then this setting must be equal to that number.
#CREATE_SC_LOOP_DEVICES=256

#############################################################################
# CLUSTER CONTROLLER (CC) / NODE CONTROLLER (NC) SHARED
CONFIGURATION
#############################################################################

# The level of logging output.  Valid settings are, in descending order of
# verbosity: EXTREME, TRACE, DEBUG, INFO, WARN, ERROR, and FATAL.  The
# default is INFO.
#LOGLEVEL="INFO"

# The number of old log files to keep when rotating logs, in range [0-999].
# The default is 10. When set to 0, no rotation is performed and log size
# limit is (LOGMAXSIZE, below) is not enforced.
#LOGROLLNUMBER="10"

# The maximum size of the log file, in bytes. 100MB by default. For this
# size to be enforced, LOGROLLNUMBER, above, must be 1 or higher. If log
# rotation is performed by an outside tool, either set LOGROLLNUMBER to 0
# or set this limit to a large value.
#LOGMAXSIZE=104857600

# On a NC, this defines the TCP port on which the NC will listen.
# On a CC, this defines the TCP port on which the CC will contact NCs.
```

APÊNDICE B – Alteração das Chaves de acesso as MV’s

92.168.0.14 ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAxiAaYtmlrUVym31JbzwbhRGAxm5xlTnVTuTg
LoFEdnV7oEFwOS+a5Z+b9vmObv2XRbBM4MYfr9qW6PfL6+a/n8JryFXTvF9mcs+ZFYg
CfC2i9CghhsMmTgIEtU1yAXjwtNthDAg6Ros6WiEZJoY+Kd+yZGgnkOag4x7O6K8ljG4e
BPV89yPVIrQF9rJqgxuuWPG8fHAA6XrZI34uyulc3CelavYTRyG5Le/OBNqa0RSEd1wadq
G9AFJcIilTm4BUbaJSy8nMUS+akCr895tAF4P1PWQltdP+7lD376bWVBsKGmTnGve7SH
vodUrQfz2LzKE5DARLkGS3nWAOWFlMYw==

192.168.0.13 ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEA4srT/l5fLrwSSZIl7DF5WSciNu/mAlDpWKs0v
Mq1d9c1F7+ERgaFZ/r3Hf+YAvAILEB0IoeJRIHIV0+AqBOOF0AWdNaiSx0PQh/Ctc2HT
HsC+Roc2C/WEizpHxcE9N0l3X1IQ68zLK3hTu6inYdiAS3W44el0esTgn8NnMDS54clPM
2KXOLFUuapWKzSmjHb1RGAmYZGgFdy1wsQKLmYiMGxR0jto9ZM8K5B/hOKrqJXM
K8pfPVhaZ9z9c8gIw6x/ygPi3Hff/MiTpU+sJKYv/b4wAnfTYp7wwsQKzV+IqzXOoMngzcr
8GMiVkdV/xHjF5ugz9Bljy+jS6TNAzUNZw==

192.168.0.15 ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAtngVk68X71mPNaBIyADXBqNaDKneAyATm
wyRTRwxhChi70LbGxz7fpMQ+2Bxfetuxv56TMb5J4g3aEUZnx7WjVJz9902ISqq1kDYgdf
4NNWf0FjXlcaIoWS0NzXVpZ15EIwU1w7CgUgXDE1W5+kD+O20oSHwAqvGoM8sHL
O1siTDWIY8dhEVwjiedTUMzzp1GiikPf6ZoZFnnbOLCClb7JzcG9bNXSZFhD+/7c74E7bh
KVqdUjP/fCdDsJ+f2GeFfkYcdDdaTQpGJornEaY5qbFwDV2HTcCxBjhPssFFjA1qkLslUZ
P0S9uLXSkVCwsAfgIz6EWvdPGVyW2jr5GtfQ==

192.168.0.16 ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAoBM/Ju9+wSuNqGUyDLn0VzNDzoBbBVXZZ
6rJd55g1in6XpP2rJ/EcU/pTUJ6Cffug9L3G4hj06QjA32kTBXSiKZFokG7FSBAitL39ua24w
E5VlelTU6JPQHdETRRpz3HIcPs1B99LD21w7rxLin7B/7q5JuHpKzj3atDqwLE0uNJ2K7Lj
Or5kQz6RZ7L0FYhFuJvlBwIYVlTOKdo6slQDvfwTSB8LCQmIqUylXJPxA5OydN4NONe
hRYA5VWT7iE5VLN5LPglC995FCHXQ7qCkeNPrszlUgS74TZBZ/rw61k60lL+NUZiyL5+
ru/u6WBBiz7gn6/CGrhwnMvrMfIpzw==

192.168.0.17 ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAusADakrk1XmYA7uRp/zE3bGHfwlKl1ETku7Z
7XsyQ5o8XrychRrkpptC69DqF/YFzexZBbQy40mIkrLadta7CINf5CCRpfb2uw0bDemrA5m
7YjnsluKViuneY+imO1Ltck33HD4803lhJ62PzrHQQeirK4lyde62u3GkE05Ml1nua56o56g1
YTtWbS2HbCVDgoV1EOWvLxc3OEdC7Bj9SXh0HWx7itkpHfzCdsFFay2TVrONso8XxX
WFEEo3ssOrbbb/OUqXTADjU2QV4gAH5N/aoNC24+uLN3Sb2VNBMhtJg4JL8fSjQTH/g
MYow1991APfyYwebmY7BCVAxoUJeQ==

APÊNDICE C – Arquivo de atribuição do IP do servidor no Nó Controlador

```
# This is a config file for the Zabbix agent daemon (Unix)
# To get more information about Zabbix, visit http://www.zabbix.com

############ GENERAL PARAMETERS #################

### Option: PidFile
#       Name of PID file.
#
# Mandatory: no
# Default:
# PidFile=/tmp/zabbix_agentd.pid

### Option: LogFile
#       Name of log file.
#       If not set, syslog is used.
#
# Mandatory: no
# Default:
# LogFile=

LogFile=/tmp/zabbix_agentd.log

### Option: LogFileSize
#       Maximum size of log file in MB.
#       0 - disable automatic log rotation.
#
# Mandatory: no
# Range: 0-1024
# Default:
# LogFileSize=1

### Option: DebugLevel
#       Specifies debug level
#       0 - no debug
#       1 - critical information
#       2 - error information
#       3 - warnings
#       4 - for debugging (produces lots of information)
#
```

```
# Mandatory: no
# Range: 0-4
# Default:
# DebugLevel=3

### Option: SourceIP
#
# Mandatory: no
# Default:
# SourceIP=

### Option: EnableRemoteCommands
#       Whether remote commands from Zabbix server are allowed.
#       0 - not allowed
#       1 - allowed
#
# Mandatory: no
# Default:
# EnableRemoteCommands=0

### Option: LogRemoteCommands
#       Enable logging of executed shell commands as warnings.
#       0 - disabled
#       1 - enabled
#
# Mandatory: no
# Default:
# LogRemoteCommands=0

##### Passive checks related

### Option: Server
#       List of comma delimited IP addresses (or hostnames) of Zabbix servers.
#       Incoming connections will be accepted only from the hosts listed here.
#       No spaces allowed.
#       If IPv6 support is enabled then '127.0.0.1', '::127.0.0.1', '::ffff:127.0.0.1' are treated
equally.
#
# Mandatory: no
# Default:
# Server=

Server=192.168.0.14
```

```
### Option: ListenPort
#       Agent will listen on this port for connections from the server.
#
# Mandatory: no
# Range: 1024-32767
# Default:
# ListenPort=10050
### Option: ListenIP
#       List of comma delimited IP addresses that the agent should listen on.
#       First IP address is sent to Zabbix server if connecting to it to retrieve list of active
checks.
#
# Mandatory: no
# Default:
# ListenIP=0.0.0.0

### Option: StartAgents
#       Number of pre-forked instances of zabbix_agentd that process passive checks.
#       If set to 0, disables passive checks and the agent will not listen on any TCP port.
#
# Mandatory: no
# Range: 0-100
# Default:
# StartAgents=3

##### Active checks related

### Option: ServerActive
#       List of comma delimited IP:port (or hostname:port) pairs of Zabbix servers for active
checks.
#       If port is not specified, default port is used.
#       IPv6 addresses must be enclosed in square brackets if port for that host is specified.
#       If port is not specified, square brackets for IPv6 addresses are optional.
#       If this parameter is not specified, active checks are disabled.
#       Example: ServerActive=127.0.0.1:20051,zabbix.domain,[::1]:30051,::1,[12fc::1]
#
# Mandatory: no
# Default:
# ServerActive=

ServerActive=127.0.0.1

### Option: Hostname
#       Unique, case sensitive hostname.
```

```
#       Required for active checks and must match hostname as configured on the server.
#       Value is acquired from HostnameItem if undefined.
#
# Mandatory: no
# Default:
# Hostname=

Hostname=Zabbix server

### Option: HostnameItem
#       Item used for generating Hostname if it is undefined.
#       Ignored if Hostname is defined.
#
# Mandatory: no
# Default:
# HostnameItem=system.hostname

### Option: RefreshActiveChecks
#       How often list of active checks is refreshed, in seconds.
#
# Mandatory: no
# Range: 60-3600
# Default:
# RefreshActiveChecks=120

### Option: BufferSend
#       Do not keep data longer than N seconds in buffer.
#
# Mandatory: no
# Range: 1-3600
# Default:
# BufferSend=5

### Option: BufferSize
#       Maximum number of values in a memory buffer. The agent will send
#       all collected data to Zabbix Server or Proxy if the buffer is full.
#
# Mandatory: no
# Range: 2-65535
# Default:
### Option: MaxLinesPerSecond
#       Maximum number of new lines the agent will send per second to Zabbix Server
#       or Proxy processing 'log' and 'logrt' active checks.
#       The provided value will be overridden by the parameter 'maxlines',
```

```
#       provided in 'log' or 'logrt' item keys.
#
# Mandatory: no
# Range: 1-1000
# Default:
# MaxLinesPerSecond=100

### Option: AllowRoot
#       Allow the agent to run as 'root'. If disabled and the agent is started by 'root', the agent
#       will try to switch to user 'zabbix' instead. Has no effect if started under a regular user.
#       0 - do not allow
#       1 - allow
#
# Mandatory: no
# Default:
# AllowRoot=0

############ ADVANCED PARAMETERS #################

### Option: Alias
#       Sets an alias for parameter. It can be useful to substitute long and complex parameter
name with a smaller and simpler one.
#
# Mandatory: no
# Range:
# Default:

### Option: Timeout
#       Spend no more than Timeout seconds on processing
#
# Mandatory: no
# Range: 1-30
# Default:
# Timeout=3

### Option: Include
#       You may include individual files or all files in a directory in the configuration file.
#       Installing Zabbix will create include directory in /usr/local/etc, unless modified during
the compile time.
#
# Mandatory: no
# Default:
# Include=
```

```
# Include=/usr/local/etc/zabbix_agentd.userparams.conf
# Include=/usr/local/etc/zabbix_agentd.conf.d/

####### USER-DEFINED MONITORED PARAMETERS #######

### Option: UnsafeUserParameters
#       Allow all characters to be passed in arguments to user-defined parameters.
#       0 - do not allow
#       1 - allow
#
# Mandatory: no
# Range: 0-1
# Default:
# UnsafeUserParameters=0

### Option: UserParameter
#       User-defined parameter to monitor. There can be several user-defined parameters.
#       Format: UserParameter=<key>,<shell command>
#       See 'zabbix_agentd' directory for examples.
#
# Mandatory: no
# Default:
# UserParameter=
```

APÊNDICE D – Teste utilizado para sobrecarregar as MV's

```
PARA_BAIXAR="$PWD/pega.para_baixar.txt"
BAIXADOS="$PWD/pega.baixados.txt"
E_LST='THE END'
APP_NM='PEGA'
if [ ! -e $BAIXADOS ]; then
   touch $BAIXADOS
fi

if [ ! -e $PARA_BAIXAR ]; then
   echo >> $E_LST $PARA_BAIXAR
fi

while : ; do
   IFS_O=$IFS
   IFS=$'\n'
   for i in $( grep -v '#' "${PARA_BAIXAR}" | uniq | grep -
v '^$' ); do
      JA_FOI_BAIXADO=$( grep $i "${BAIXADOS}" )
      echo -e "${APP_NM}:\n Baixando \n $i...\n"
      if [ ${#JA_FOI_BAIXADO} -eq 0 -a $i != $E_LST ]; then
         wget -c $i &&
         echo $i >> "${BAIXADOS}"
      else
         echo -e "${APP_NM} :\n $i \n já foi baixado
anteriormente. \n"
      fi
   done
   if [ $i == $E_LST ]; then
      IFS=$IFS_O
      break
   fi
done
exit
```