UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Pedro Henrique Pereira Martins

PROPOSTA DE UM MODELO AUTÔNOMO PARA MONITORAMENTO DE AMBIENTES DE COMPUTAÇÃO EM NUVEM

Florianópolis

2015

SUMÁRIO

1 INTRODUÇÃO

Computação em nuvem vem sendo cada vez mais utilizada devido ao seu modelo permitir um melhor aproveitamento dos recursos de hardware e software, acarretando em redução de custos, melhor eficiência energética, elasticidade, flexibilidade e disponibilidade de recursos por demanda (ARMBRUST et al., 2010) e (SINGH; SOOD; SHARMA, 2011).

Com o aumento da demanda por serviços prestados pela nuvem, há um constante crescimento da infraestrutura de modo a atender a demanda (SINGH; SOOD; SHARMA, 2011) e (WEINGÄRTNER; BRÄSCHER; WESTPHALL, 2015). O crescimento constante dos ambientes da nuvem acaba aumentando a complexidade de sua gerência tornando esta inviável de ser realizada por administradores sem auxilio de ferramentas apropriadas (WEINGÄRTNER; BRÄSCHER; WESTPHALL, 2015), (HALL, 2012) e (URGAONKAR; SHENOY; ROSCOE, 2002). Kephart e Chess (2003) apresentam um modelo de computação autônoma como uma solução para lidar com grandes ambientes computacionais heterogêneos e complexos como ambientes de computação em nuvem.

Kephart e Chess (2003) definem um elemento autônomo constituinte de quatro (4) etapas, monitoramento, análise, planejamento e execução, onde cada etapa depende da anterior. Cada etapa é detalhada a seguir.

- Monitoramento: durante este processo são coletadas informações sobre os recursos alocados e utilizados. Este processo é realizado periodicamente de modo a criar um histórico de alocação e uso de recursos;

- Análise: após ter os dados coletados, são iniciados os processos de análise em busca de tendências de consumo e comportamento das cargas e problemas referentes a carga que competem por recursos no mesmo servidor físico;

- Planejamento: depois de identificar as tendências de comportamento das cargas do ambiente, e possíveis problemas de degradação de serviço, pode-se traçar ações de modo a evitar as degradações de serviço e viabilizar a otimização do ambiente;

- Execução: após ter traçado uma sequência de ações serem tomadas é necessário executar todas as ações que foram previamente

planejadas de modo a buscar atender aos objetivos do sistema de gerenciamento.

Os processos descritos anteriormente são executados constantemente, de modo que seja possível para o sistema de gerenciamento identificar o contexto em que se encontra e adaptar-se ao ambiente a ser gerenciado de modo a atender aos objetivos gerenciais.

Uma das necessidades para se efetuar o monitoramento do uso de recursos de ambientes de nuvem se da para fins comerciais, como cobrança relativa a serviços prestados. O valor a ser cobrado pode ser acordado entre o usuário e o fornecedor de recursos da nuvem ou pode ser pré-definido pelo fornecedor (ACETO et al., 2012). Além disso, o monitoramento também pode ser usado para viabilizar técnicas de otimização e auxiliar numa melhor gerência dos ambientes de computação em nuvem. Para viabilizar tais otimizações é importante que o monitoramento forneça dados precisos e em tempo hábil sem afetar o funcionamento da nuvem (ACETO et al., 2012) e (VIRATANAPANU et al., 2010)

O monitoramento em tempo real da nuvem é desafiador. Deve-se levar em conta o grande número de serviços e usuários, que torna a nuvem um ambiente com um grande volume de informação para ser coletada e analisada (SHAO et al., 2010) e (CLAYMAN; GALIS; MAMATAS, 2010).

Outro fator que contribui para a complexidade de monitorar os ambientes de nuvem é a heterogeneidade dos seus componentes. Ambientes de nuvem são constituídos de diferentes clusters de servidores, cada um com seus respectivos processadores, memória, placa de rede, periféricos e virtualizadores. Deste modo, é necessário implementar meios para monitorar cada um desses elementos distintos (SHAO et al., 2010). Além disso, os elementos que compõem o ambiente de computação em nuvem, podem estar dispersos geograficamente, o que gera a necessidade de ter um sistema de monitoramento distribuído, com nodos colhendo informações em cada elemento e enviando para um repositório de dados.

O processo de monitoramento pode vir a sobrecarregar os recursos da nuvem se não for devidamente implantado, sendo este um dos principais desafios para o monitoramento dos recursos (SHAO et al., 2010) e (CLAYMAN; GALIS; MAMATAS, 2010). Além disso, após implantado o sistema de monitoramento, qualquer modificação necessária deve ser feita sem a necessidade de desligar ou reiniciar o ambiente (CLAYMAN; GALIS; MAMATAS, 2010).

Problemas resultantes de um monitoramento mal sucedido são:

- A queda de desempenho da nuvem devido a sobrecargas de processamento com monitoramento;
- Ações de gerência mal sucedidas devido a dados inconsistentes advindos do processo de monitoramento;
- Perda de dados e/ou obtenção de dados corrompidos durante o processo de atualização dos componentes de monitoramento.

Atualmente as ferramentas de orquestração de computação em nuvem (levantadas durante o desenvolvimento deste trabalho) em sua maioria não realizam ou realizam mas não armazenam de modo histórico os dados resultantes do monitoramento dos recursos utilizados. O que estas ferramentas costumam adotar é o monitoramento momentâneo do ambiente para indicar aos administradores o estado corrente deste, sem a criação de um histórico de uso dos recursos.

Hoje em dia, existem ferramentas que realizam o monitoramento e podem ser acopladas a ferramentas de orquestração, porém são de difícil instalação e dependentes de APIs (Interface de programação de aplicações) de dos orquestradores que não são completas pois não permitem o monitoramento de uma série de recursos físicos e algumas costumam apresentar dados inconsistentes com o ambiente, além de ser mais uma camada de software para gerenciar.

O Zenoss (ZENOSS, 2015) é um exemplo de ferramenta de monitoramento que apresentam alguns problemas. No caso do Zenoss, e sua extensão para monitoramento do CloudStack (APACHE, 2015b), o CloudStack ZenPack (CLOUDSTACKZENPACK, 2015), existe um problema no monitoramento de uso de memória do servidor físico. A ferramenta faz uma requisição do uso de memória para o Xen (INC., 2015) e projeta o retorno como se fosse o uso total de memória das máquinas virtuais alocadas no servidor. O problema é que esta requisição do Xen, retorna apenas o uso de memória relativo ao *dom0*. Deste modo o Zenoss acaba fornecendo dados de apenas uma máquina virtual específica e não do servidor monitorado como um todo, o que pode levar a decisões inconsistentes no momento de gerenciamento e otimização dos ambientes.

É necessário fazer uso de boas técnicas de manipulação de dados devido ao grande volume de informações que podem ser monitoradas em ambientes de nuvem. A ferramenta de monitoramento necessita ser o menos invasiva possível, ou seja, ser invisível para os usuários de ambientes de computação em nuvem. Além disso, deve-se tratar a precisão das informações capturadas, os dados fornecidos devem condizer com o ambiente monitorado (ACETO et al., 2012). Quanto mais rápida

as informações sobre o estado do ambiente forem disponibilizadas, mais eficientemente se conseguirá administrá-lo.

1.1 MOTIVAÇÃO

O monitoramento é de fundamental importância para ambientes de nuvem, pois proporciona um melhor entendimento do estado em que o ambiente se encontra, do estado das aplicações que estão hospedadas, do uso/disponibilidade de recursos físicos do ambiente e também para determinar se os acordos de prestação de serviço estão sendo devidamente cumpridos.

Como apresentado em (ACETO et al., 2013), existe uma carência de autonomia nas ferramentas de monitoramento atuais. Assim, toda vez que novos recursos são inseridos no ambiente, deve-se informar às ferramentas que tais recursos devem ser monitorados, quando o ideal era a ferramenta detectar e passar a monitorar os recursos autonomamente.

Outro ponto sobre as ferramentas atuais, é que a proposta delas em sua maioria, é monitorar desde a parte de aplicação até o nível de infraestrutura. Com isso, as ferramentas de monitoramento acabam se tornando complexas, com uma instalação complicada e consumindo mais recursos do ambiente do que se fossem específicas para apenas um modelo de serviço de nuvem.

Motivado pela carência de ferramentas de monitoramento pouco intrusivas, autônomas, com foco apenas na infraestrutura de ambientes de computação em nuvem, compatível com as ferramentas de orquestração atuais e de fácil instalação. Surgiu o interesse no projeto de uma abordagem que supra a necessidade de monitoramento da infraestrutura da nuvem. Capacitando assim, que ferramentas como CloudStack, possam fornecer todo o suporte para o monitoramento da infraestrutura do ambiente de nuvem sem interferência nos serviços por ela já fornecidos de modo autônomo e sem configurações adicionais.

2 OBJETIVOS

Este capítulo irá tratar dos objetivos que este trabalho pretende alcançar.

2.1 OBJETIVOS GERAIS

Este trabalho busca analisar o tema de monitoramento de ambientes de computação em nuvem e elementos autônomos, para poder então propor um modelo de monitoramento autônomo para ambientes de computação em nuvem que fornecem infraestrutura como serviço.

2.2 OBJETIVOS ESPECÍFICOS

- Levantar trabalhos relacionados e fazer um estudo sobre monitoramento de ambientes de computação em nuvem;
- Definir um modelo para realização de monitoramento autônomo de um ambiente de nuvem que forneça infraestrutura como serviço;
- Criar um *framework* a partir do modelo definido para viabilizar a implantação em ferramentas de orquestração de nuvem;
- Verificar o impacto que o *framework* pode causar no ambiente de nuvem monitorado.

3 FUNDAMENTAÇÃO

Neste capítulo são apresentados os conceitos e tecnologias utilizadas para o desenvolvimento deste trabalho.

3.1 COMPUTAÇÃO EM NUVEM

A computação em nuvem é um modelo ubíquo, conveniente e de acesso compartilhado a recursos computacionais como servidores, armazenamento, aplicações e redes (MELL; GRANCE, 2011). Estes recursos compartilhados podem ser rapidamente provisionados e liberados com um esforço mínimo de gestão ou interação com o provedor de serviços (MELL; GRANCE, 2011).

3.1.1 Características

Em (MELL; GRANCE, 2011) são definidas como caracteristicas básicas de um ambiente de computação em nuvem:

1. Acesso aos recursos sob demanda - o usuário pode escolher quais serviços/componentes ele irá usar dentro do que é disponibilizado pela nuvem sem a necessidade de interação com administradores;

2. Disponibilidade - ambientes de computação em nuvem devem estar sempre disponíveis para o usuário pela internet independente da plataforma de acesso (*tablet*, *smartphone*, *desktop* e outros);

3. Recursos compartilhados - os recursos físicos da nuvem devem poder atender diferentes usuários simultaneamente;

4. Elasticidade - conforme um usuário solicita mais recursos, a nuvem deve alocar mais recursos para este usuário evitando a degradação dos serviços prestados, assim como se um usuário deixar de usar recursos, a nuvem pode remover recursos excedentes;

5. Monitoramento de serviços - a nuvem deve ser capaz de monitorar o uso de recursos por cada serviço, proporcionando transparência aos fornecedores e usuários do serviço.

3.1.2 Modelos de serviço

Mell e Grance (2011) descrevem os modelos de serviço disponibilizados pela nuvem como sendo:

1. Software como serviço: o provedor de nuvem fornece uma aplicação final via rede para usuários. O usuário não tem controle algum sobre a infraestrutura que hospeda a aplicação ou da plataforma que a aplicação usa, ele apenas consegue usar a aplicação;

2. Plataforma como serviço: são disponibilizadas diferentes plataformas para o usuário desenvolver e hospedar suas próprias aplicações. Neste nível o usuário tem controle total das aplicações, e pode configurar por completo a plataforma. O usuário não possui acesso a recursos do sistema operacional que hospeda a plataforma;

3. Infraestrutura como serviço: a nuvem fornece toda a infraestrutura para o usuário, como processador, disco, memória e rede. Neste nível o usuário tem uma máquina virtual pronta para uso, ele define quais plataformas ele irá usar e quais aplicações ele irá executar em cada plataforma, porém o usuário não tem acesso e controle dos componentes físicos do ambiente.

   Cada modelo de nuvem descrito acima visa um público diferente com diferentes propósitos, o software como serviço visa mais usuários de aplicações. O modelo de plataforma como serviço procura atender os desenvolvedores que usarão as plataformas para hospedar suas aplicações. Em contra partida, o modelo de fornecimento de infraestrutura como serviço tem como foco empresas que desenvolvem serviços e os hospedam na infraestrutura de ambientes de computação em nuvem para minimizar seus custos com infraestrutura. Este trabalho tem como foco ambientes de nuvem computacional que atuam fornecendo infraestrutura como serviço.

3.2 VIRTUALIZAÇÃO

A virtualização é um processo que foi primeiramente desenvolvido pela IBM com intuito de particionar um *mainframe* com alta capacidade computacional (SAHOO; MOHAPATRA; LATH, 2010).

O processo de virtualização consiste na existência de um virtualizador que gerencia sistemas operacionais convidados e intermedeia toda a comunicação entre hardware e esses sistemas (MENASCÉ, 2005). Sistemas operacionais convidados são executados sobre o virtualizador como se fossem aplicações em um sistema operacional qualquer. Portanto os sistemas operacionais convidados, não possuem acesso direto aos dispositivos físicos.

O uso da virtualização possibilita que os mesmos componentes de hardware suportem diferentes sistemas operacionais como sendo diversas unidades lógicas. Estas unidades lógicas em que o hardware é dividido são nomeadas de *Virtual Machine* (VM), em português, máquina virtual. Os sistemas operacionais e aplicações dentro das máquinas virtuais enxergam esses ambientes como reais. Ao final do processo de virtualização, para o usuário final, ter uma máquina virtual ou uma máquina física não irá mudar sua experiência de uso do computador (SAHOO; MOHAPATRA; LATH, 2010).

3.2.1 Vantagens e desvantagens da virtualização

A virtualização permite um melhor aproveitamento dos recursos de hardware disponíveis. Permite a alteração de peças de hardware sem a necessidade de parar o funcionamento das VMs por longos períodos, bastando migrar a VM para outro servidor físico. O custo alugar uma máquina virtual em um servidor de computação em nuvem é menor que manter uma máquina física equivalente. (SAHOO; MOHAPATRA; LATH, 2010).

Como problemas temos uma sobrecarga de processamento para gerenciar a inúmera quantidade de máquinas virtuais. O desenvolvimento de virtualizadores não é algo simples e é de grande importância ter redundância de hardwares pois se ocorrer defeito em um componente, todas as VMs vinculadas a tal componente serão penalizadas (POPEK; GOLDBERG, 1974) .

3.2.2 Tipos de virtualização

Nesta seção, serão discutidos os tipos de virtualizações segundo (BEN et al., 2010), e suas principais características.

- Virtualização por contêiner: este tipo de virtualização precisa de um sistema operacional base. Neste caso, o virtualizador irá criar

instâncias virtuais do sistema operacional base, isoladas logicamente. Estas instâncias são nada mais que ponteiros para os endereços físicos das rotinas do sistema operacional base. Isto gera um problema que não existe nos outros tipos de virtualização, que é o fato de todas as aplicações das instâncias dividirem os mesmos endereços físicos para suas chamadas de sistema, com isso, se o sistema operacional base for sucetível a algum ataque, uma aplicação maliciosa pode danificar o sistema operacional base, e com isso, prejudicar todas as outras instâncias virtuais desse sistema. Esse tipo de virtualização consome menos memória que as demais, pois usa apenas uma instância física de sistema operacional. Neste tipo de virtualização, os clientes estão presos a criarem VMs exclusivamente do mesmo tipo do sistema operacional base;

- Virtualização do tipo 1: este tipo de virtualização executa diretamente sobre o hardware, neste caso o virtualizador irá gerar abstrações de hardware completas para poder hospedar um sistema operacional qualquer. Isto cria um consumo extra de memória, pois cada instância irá ter seu próprio sistema operacional em memória. Este tipo de virtualização, possui a melhor performance de uso do processador dentre os tipos aqui apresentados. Neste tipo de virtualização, os clientes podem escolher seus próprios sistemas operacionais para serem instaladas nas VMs;

- Virtualização do tipo 2: esta virtualização assim como a virtualização por contêiner, precisa de um sistema operacional como base. Porém nesse caso, o virtualizador irá emular um hardware que será usado pelas instâncias virtuais criadas. Isso possibilita que as instâncias tenham seus próprios sistemas operacionais independentemente de qual é o sistema operacional base. O virtualizador não tem acesso direto ao hardware, podendo assim ser um virtualizador mais simples que da virtualização do tipo 1, já que é o sistema operacional base que irá tratar da comunicação com o mesmo. Esse tipo de virtualização é um meio termo entre a virtualização do tipo 1 e por contêiner.

Cada tipo de virtualização possui vantagens e desvantagens. O tipo 1, é o mais complexo entre os tipos de virtualização e o com menor penalidade de processamento. O tipo 2 usa um sistema operacional nativo como base para o funcionamento do virtualizador, o que remove parte da complexidade do virtualizador, porém acaba prejudicando o desempenho devido a camada de software extra. A virtualização por

contêiner é o tipo com menor uso de memória, porém ela amarra os usuários ao sistema operacional base onde é executado o virtualizador.

3.2.3 Virtualizadores

O virtualizador é responsável pela gerência dos recursos de hardware e fornecimento desses recursos para as máquinas virtuais. O responsável de fato pela gerência, criação e destruição das máquinas virtuais, é o *Virtual Machine Monitor* (VMM), em português, monitor de máquina virtual (VMWARE, 2007). Portanto, toda vez que uma VM é criada, antes, é criado um VMM, e este passa a gerenciar a VM, e destruí-la quando necessário. Logo, existe um VMM para cada VM.

O Xen (INC., 2015) é um virtualizador do tipo 1 de código aberto disponibilizado pela Citrix (SYSTEMS, 2015a), que da suporte as arquiteturas x86 (MAZEGEN, 2007), x86 64 (CORPORATION, 2015b), IA32 (CORPORATION, 2015a), IA64 (DOSHI, 2015) e PowerPC (IBM, 2015). As VMs instanciadas com esse virtualizador, são denominadas de domínio sem privilégios (DomU) (SYSTEMS, 2015c), além disso, existe uma VM que cuida de todo o fluxo de entrada e saída de dados dos DomU, que é denominada de domínio de controle (Dom0) (SYSTEMS, 2015b). Além de intermediar o acesso a entrada e saída de dados dos DomU, o Dom0 também fornece uma série de comandos para destruir, criar ou modificar cada DomU. Esses comandos de gerência disponibilizados no Dom0, são nativos da biblioteca XL (SYSTEMS, 2015e), porém, são de difícil usabilidade. O Xapi (SYSTEMS, 2015d), é uma abstração para esses comandos que torna mais simples o uso dos recursos de criação de VMs, clusters e migração de VMs entre servidores de um mesmo cluster.

O KVM (LINUX-KVM, 2015) assim como o Xen, é um virtualizador do tipo 1 de código aberto. Suportado pela Red Hat (REDHAT, 2015), o KVM é uma extenção do kernel do linux que permite a criação e gerência de VMs. O KVM usa o QEMU (QEMU, 2015) para emular o hardware que as VMs instanciadas irão usar. Um dos diferenciais do KVM é o fato dele não se preocupar com a gestão de diferentes servidores presentes em cluster, não tendo assim, qualquer custo de processamento para gerência dos servidores em um cluster. Usar esta abordagem no controle dos clusters, torna o KVM dependente de uma ferramenta de orquestração que dê suporte a gerência de servidores dentro de um cluster.

O ESXi (VMWARE, 2015a) é um virtualizador de código fechado e

licença gratuíta que foi desenvolvido pela VMware (VMWARE, 2015c). Tem suporte apenas para arquiteturas x86 de 32 e 64 bits. É um virtualizador do tipo 1. Este virtualizador é usado no vSphere que é uma ferramenta de criação de ambientes de nuvem criada pela VMware. O ESXi só permite a migração de VMs para diferentes servidores em sua versão comercial.

O LXC (CONTAINERS., 2015) é um virtualizador por contêiner gratuito de código aberto suportado pela Canonical (CANONICAL, 2015). O LXC da suporte para criação de instâncias de distribuições linux, logo este virtualizador suporta todas as arquiteturas suportadas pelo linux. Instâncias criadas pelo LXC não possuem privilégios de root. O LXC está no repositório de software de distribuições linux, a instalação dele é bem simples.

VirtualBox (ORACLE., 2015) é um virtualizador de código aberto do tipo 2 mantido pela Oracle (INC., 2015). Este virtualizador pode ser instalado em distribuições Windows, Linux, Macintosh, Solaris e da suporte para instâncias de Windows (NT 4.0, 2000, XP, Server 2003, Vista, Windows 7, Windows 8), DOS/Windows 3.x, Linux (2.4, 2.6 e 3.x), Solaris e OpenSolaris, OS/2, e OpenBSD.

Tabela 1 – Comparação entre virtualizadores analizados

	Empresa Responsável	Arquiteturas Suportadas	Tipo de virtualização	Tipo de Licensa
Xen	Citrix Systems	x86, x86-64, IA 64, PowerPC	1	Código aberto
KVM	Red Hat	x86, x86-64, IA 64, PowerPC	1	Código aberto
ESXi	VMware	x86, x86-64	1	Gratuita/Comercial
LXC	Canonical	x86, x86-64, IA 64, PowerPC	Contêiner	Código aberto
VirtualBox	Oracle	x86, x86-64, IA 64	2	Código aberto

Dentre os virtualizadores analizados, será utilizado neste trabalho o Xen devido a ser um virtualizador de código aberto e suportado por uma comunidade amplamente ativa. O fato do Xen ter seu código aberto, possibilita que toda uma comunidade interaja e possa contribuir com melhorias tanto na documentação quanto em correções de *bugs* e evolução de suas funcionalidades.

3.3 ORQUESTRAÇÃO DE AMBIENTES DE COMPUTAÇÃO EM NUVEM

As ferramentas de orquestração na nuvem servem para abstrair do administrador toda a parte de interação com os virtualizadores, serviços de armazenamento, dispositivos de redes e outros. Permitindo que o administrador não precise de amplos conhecimentos nas tecnolo-

gias de virtualização, rede, armazenamento e outras utilizadas para se criar o ambiente de nuvem.

Essas abstrações proporcionadas pelas ferramentas de orquestração tornam possível que o administrador da nuvem possa delimitar recursos físicos para cada usuário facilmente. Assim como também possibilita que usuários sem conhecimento nenhum de computação possam criar suas próprias máquinas virtuais.

3.3.1 Ferramentas de orquestração

Seguindo a análise realizada no trabalho (BRäSCHER, 2015), foram análizadas as seguintes ferramentas de orquestração para uso no decorrer deste trabalho.

- CloudStack: o cloudStack (APACHE, 2015b) é uma ferramenta de orquestração de código aberto, suportada pela fundação Apache (APACHE, 2015a) e desenvolvida em java. O cloudStack da suporte para virtualizadores como KVM, VM-ware , Xen e Hyper-V (MICROSOFT, 2015a). Esta ferramenta da suporte aos sistemas de arquivos iSCSI (MICROSOFT, 2015b) e NFS (MICROSOFT, 2013);
- Eucalyptus: o Eucalyptus (HP, 2015a), assim como o CloudStack, é uma ferramenta de orquestração de ambientes de computação em nuvem de código aberto. Suportada pela HP (HP, 2015b) e desenvolvida em java e C. O Eucalyptus, usa o Walrus como sistema de arquivo e da suporte aos virtualizadores KVM, XenServer e VMware;
- OpenStack: O OpenStack (OPENSTACK, 2015), da suporte aos mesmos virtualizadores que o CloudStack, é uma ferramenta de código aberto desenvolvida em Python. Suporta os sistemas de arquivo Swift e Cinder.

Tabela 2 – Comparação entre ferramentas de orquestração analizadas

	Linguagem Suportada	Sistema de Arquivos	Tipo de Licensa
CloudStack	Java	iSCSI , NFS	Código aberto
OpenStack	Python	Swift , Cinder	Código aberto
Eucalyptus	Java , C	Walrus	Código aberto

Dentre essas ferramentas de orquestração em nuvem, foi optado o uso do CloudStack devida a uma série de motivos, tais como:

- Suportado por uma fundação e uma comunidade amplamente ativa;
- Possui uma documentação detalhada;
- É codificada em java que é uma linguagem de programação bem conhecida para futuras extensões;
- Da suporte para sistemas de arquivos conhecidos, o que diminui o tempo que seria gasto com familiarizações com novos conceitos;
- É uma ferramenta de código aberto, o que possibilita um melhor conhecimento da ferramenta e implantação de melhorias.

3.4 MONITORAMENTO DE AMBIENTES DE COMPUTAÇÃO EM NUVEM

Nesta seção, serão abordadas as dificuldades relacionadas ao monitoramento na nuvem, sua importância, as características que o monitoramento deve possuir e algumas ferramentas que realizam monitoramento de ambientes de computação em nuvem.

3.4.1 Características de ferramentas de monitoramento na nuvem

Segundo Aceto et al. (2012) existem 11 características que uma ferramenta de monitoramento na nuvem deve possuir para ter um funcionamento correto e sem degradar os serviços fornecidos:

1. Escalabilidade: a ferramenta pode ser considerada escalável se ela consegue suportar uma larga quantidade de objetos a serem monitorados (CLAYMAN; GALIS; MAMATAS, 2010). Na nuvem, tal característica tem importância devida a grande gama de componentes que podem existir;
2. Elasticidade: para uma ferramenta ser considerada elástica, ela deve conseguir suportar mudanças ocorridas no ambiente monitorado, tais como novos objetos para serem monitorados ou objetos deixando de ser monitorados (CLAYMAN; GALIS; MAMATAS, 2010). Isto é de fundamental importância para ambientes de nuvem, pois o ambiente vive em contínua mudança com a adição e remoção de recursos dinamicamente;

3. Adaptabilidade: uma ferramenta de monitoramento é adaptável, se ela consegue se ajustar a mudanças de cargas impostas ao ambiente sem que este seja penalizado e mantendo o seu serviço de monitoramento ativo e funcionando corretamente (CLAYMAN; GALIS; MAMATAS, 2010). Para realizar o monitoramento do ambiente, a ferramenta consome recursos de rede, armazenamento e processamento. A ferramenta deve se ajustar para consumir menos recursos em ocasiões que o ambiente se encontra sobrecarregado;

4. Pontualidade: a ferramenta pode ser considerada pontual, se ela é capaz de relatar eventos que acontecem no ambiente, como perda repentina de um recurso, ou sobrecarga de outro, em tempo hábil para estes serem tratados (WANG et al., 2011);

5. Autonomia: para a ferramenta ser considerada autônoma, ela deve ser capaz de se gerenciar, conseguindo responder a mudanças repentinas do ambiente sem a necessidade de uma outra entidade a supervisionando. Ao mesmo tempo em que torna a complexidade do seu funcionamento discreta para usuários e gerentes do ambiente;

6. Abrangente: uma ferramenta de monitoramento é abrangente, se ela consegue monitorar diferentes tipos de recursos e capturar diferentes métricas destes recursos (HASSELMEYER; D'HEUREUSE, 2010). Isto é importante, já que ambientes de nuvem costumam ser heterogêneos;

7. Extensividade: a ferramenta pode ser considerada extensível, se ela suporta a adição de novas métricas para monitorar, sem a necessidade de modificações no código fonte da ferramenta.

8. Não intrusiva: para a ferramenta não ser considerada intrusiva, ela deve poder ser implantada no ambiente sem precisar de modificações significativas deste (KATSAROS; KUBERT; GALLIZO, 2011);

9. Confiabilidade: uma ferramenta de monitoramento é dita confiável, se mesmo com falhas em diversos componentes, ela ainda continua fornecendo seu serviço normalmente com os componentes que ainda lhe restam (LAPRIE, 2008);

10. Disponibilidade: a ferramenta de monitoramento deve estar disponível sempre que os seus serviços forem solicitados. Isto é importante para ambientes da nuvem, pois estes estão em constante atividade (SHIREY, 2007);

11. Precisão: a ferramenta é considerada precisa se as informações disponibilizadas pela ferramenta contem valores que expressem o mais aproximadamente possível o estado em que o objeto monitorado se encontra no momento que a medida foi realizada.

Além das características levantadas por (ACETO et al., 2012), também são apontadas algumas dificuldades na área de monitoramento da nuvem.

- Grande volume de informação: a nuvem possui diversos usuários, assim como componentes e um grande tráfego de rede. Com isso, algumas horas de monitoramento contínuo, podem gerar Tera Bytes de informações para serem armazenadas;
- Mudança constante do ambiente: devida a elasticidade de um ambiente de computação em nuvem, as métricas a serem monitoradas podem mudar a qualquer momento, pode haver a necessidade de monitorar um novo elemento que não estava previsto;
- Sensibilidade de ambientes carregados: em ambientes de computação em nuvem muito carregados, a presença do monitoramento pode acarretar em sobre carga de processamento, e acabar comprometendo o funcionamento de todo o ambiente;
- Garantir autonomia da ferramenta: esta demanda a possível existência de um laço que irá receber informações do ambiente de tempos em tempos (a própria ferramenta que deve obter essas informações), processar essas informações e repassar as ações que cada nodo deve executar (no caso de uma ferramenta distribuída). Deve-se realizar este processo tentando gerar o menor impacto possível no ambiente;
- Heterogeneidade do ambiente: ambiente de computação em nuvem possui inúmeros dispositivos com diferentes arquiteturas e tecnologias. Deve-se conseguir prover suporte de monitoramento para cada um desses dispositivos e ainda manter um bom isolamento entre eles;
- Rastreamento de elementos da nuvem: o ambiente de nuvem permite que seus recursos sejam migrados de um dispositivo físico para outro. Deve-se garantir que a ferramenta consiga detectar a nova localização de um recurso migrado e continuar monitorando este recurso em sua nova localização, em vez de apenas acusar uma falta do recurso em um local e a detecção de um novo recurso em outro.

O monitoramento na nuvem é de grande importância tanto para os clientes quanto para os fornecedores de serviços na nuvem. O monitoramento é um meio para se garantir um melhor controle dos dispositivos de hardware e das aplicações que são executadas nos ambientes de nuvem (ACETO et al., 2012).

O constante monitoramento do ambiente, auxilia na garantia da qualidade do serviço prestado para os usuários do ambiente, como por exemplo: disponibilidade e tempo de resposta das aplicações hospedadas. Para os fornecedores de serviços, o monitoramento é um modo de mensurar os recursos utilizados pelo usuário e então poder ser cobrada uma taxa referente ao uso dos mesmos (ACETO et al., 2012). Para ambos os casos, o monitoramento auxilia na prevenção e até mesmo correção de desvios no termo de prestação de serviços acordado por usuário e provedor (ACETO et al., 2012).

3.4.2 Ferramentas de monitoramento

Nesta seção serão descritas ferramentas de monitoramento de código aberto que são comumente usadas para monitorar ambientes de computação em nuvem.

- Nagios: o Nagios (NAGIOS, 2015) suporta o monitoramento de máquinas virtuais e serviços de armazenamento (CARON et al., 2012). O Nagios é uma ferramenta de código aberto que possui inúmeras extensões. Existem extensões que possibilitam o Nagios monitorar sistemas como Eucalyptus, CloudStack(NAGIOS., 2015a) e OpenStack (NAGIOS., 2015b);

- Zenoss: O Zenoss (ZENOSS, 2015) é uma ferramenta opensource escrita em Java com suporte a diversos virtualizadores, como por exemplo, KVM, XenServer e VMware. Usuários do CloudStack, tem a possibilidade de usar o Zenoss ao instalar a extensão (CLOUDSTACKZENPACK, 2015), usuários de OpenStack podem usar a extensão (OPENSTACKZENPACK, 2015). Estas extensões ficam limitadas pelas funcionalidades disponibilizadas através das APIs das ferramentas de orquestração;

- Nimbus: a Nimbus (PROJECT, 2015) é uma ferramenta de criação e gerência de ambientes de computação em nuvem, Suporta virtualizadores como Xen ou KVM e da suporte para monitoramento do ambiente através do protocolo SNMP(NET-SNMP, 2015). Esta ferramenta é voltada especialmente para projetos relacionados a

comunidade científica. A Nimbus visa autonomia no monitoramento tanto de aplicações do ambiente quanto na infraestrutura como um todo, no caso, ela não suporta monitoramento de clusters, apesar de monitorar aplicações sendo executadas nas VMs.

- Zabbix: o Zabbix (ZABBIX, 2015) é uma ferramenta de monitoramento de infraestrutura e aplicações. Esta ferramenta da suporte para abordagens multi agente ou sem nenhum agente (centralizado). Esta ferramenta é compatível com vCenter (VMWARE, 2015b) permitindo monitoramento destes ambientes. Além de ser compatível com o vCenter, o Zabbix permite o monitoramento de qualquer dispositivo de hardware que suporte IPMI (INTEL, 2015) e dispositivos que suportem SNMP, conseguindo capturar as mesmas métricas suportadas por ambos os protocolos IPMI e SNMP. Usuários de CloudStack podem usar o Zabbix através a extensão (ZABBIX-CLOUDSTACK, 2015) e usuários de OpenStack através da extensão (ZABBIXAGENTADOPTION, 2015)

- Icinga: o Icinga (ICINGA, 2015) é uma ferramenta que pode ser instalada em qualquer distribuição Linux. Nesta ferramenta, o usuário deve especificar o que deve ser monitorado antes de inicializar a ferramenta, ou seja, qualquer novo item a ser monitorado, irá requerer uma reinicialização da ferramenta. Assim como o Nagios, esta ferramenta promete uma fácil extensibilidade, possibilitando que os usuários criem plugins para expandir o funcionamento da ferramenta, além de suportar todos os plugins desenvolvidos para o Nagios. Esta ferramenta monitora serviços de rede e consumo de recursos de hardware, como CPU e memória. Ela possui uma extensão para usuário do OpenStack (ICINGAOPENSTACKGIT, 2015), usuários de CloudStack não tem suporte ainda para uso do Icinga e acabam adotando o Nagios como substituto já que ambas as ferramentas são similares.

Dentre as ferramentas analisadas, todas suportam monitoramento de recursos de hardware usando protocolos padrões de monitoramento como SNMP e IPMI, além de usar estes protocolos, usam também APIs (Interface de programação de aplicações) de ferramentas de orquestração. Uma característica em comum dessas ferramentas, é a abordagem de coletar os dados e armazená-los de modo histórico. Todas usam o mesmo conceito de ter uma entidade centralizada que de tempos em tempos, envia mensagens para os nodos que estão nos servidores físicos coletarem os dados e enviarem para o nó central.

Tabela 3 – Características suportadas pelas ferramentas analisadas

	Nagios	Zenoss	Nimbus	Zabbix	Icinga
Escalabilidade					
Elasticidade					
Adaptabilidade					
Pontualidade		X		X	
Autonomia			X		
Abrangencia					
Extensividade	X			X	X
Não intrusiva					
Confiabilidade				X	
Disponibilidade					
Precisão					

Como mostrado na tabela 3, as ferramentas analisadas não dão suporte a inúmeras das características abortadas em (ACETO et al., 2012), isso se deve pelo fato de que as ferramentas em si não interagem diretamente com os Virtualizadores dos ambientes de computação em nuvem e sim com APIs das ferramentas de orquestração. Essas APIs por muitas vezes são incompletas e inconsistentes. Outro ponto que as ferramentas adotam é a inserção de agentes dentro das máquinas virtuais para coletar os dados de cada máquina virtual e então juntar esses dados para obter as métricas de todo o ambiente, o que torna a ferramenta intrusiva.

A proposta deste trabalho será de criar uma ferramenta que suporte essas características. A ferramenta irá usar chamadas de sistema dos Virtualizadores para conseguir obter os dados dos ambientes em vez de usar APIs das ferramentas de orquestração. Outra abordagem diferente da ferramenta proposta neste trabalho, será uma inversão do paradigma adotado pelas ferramentas analisadas, que é o fato de um nodo central fazer requisições para os nodos folhas coletarem os dados e retornarem estes para o nodo central. A proposta deste trabalho consiste em os nodos folhas enviarem periodicamente os dados coletados para o nodo central sem o nodo central precisar requisitar, isso gera um menor fluxo de rede e torna a ferramenta menos impactante para o ambiente.

4 TRABALHOS RELACIONADOS

Este capitulo aborda sobre modelos de monitoramento de ambientes de computação em nuvem, pontos fortes e fracos de cada modelo, e por fim, uma comparação entre os modelos apresentados e o modelo proposto neste trabalho.

Em (KATSAROS et al., 2012) é proposto um modelo para monitorar todas as camadas (aplicação, plataforma e infraestrutura) de um ambiente de computação em nuvem. O modelo consiste de 6 componentes constituindo a ferramenta de monitoramento como um todo, cada componente é descrito a seguir:

- Framework de serviço de monitoramento: é o principal componente, responsável por orquestrar os outros componentes e disponibilizar os resultados do monitoramento para o usuário;
- Repositório central: responsável por organizar e guardar todos os dados do monitoramento;
- Serviço de monitoramento de infraestrutura: responsável por obter dados do uso de recursos da infraestrutura;
- Serviço de monitoramento de instancia: responsável por obter dados do uso de recursos de cada VM de um servidor através do coletor de dados e guardar no repositório central;
- Repositório local: responsável por armazenar os dados de monitoramento de uma única VM;
- Coletor de dados: coleta os dados de uso de recursos de uma única VM e guarda no repositório local.

Principais características abordadas no modelo de (KATSAROS et al., 2012) é a elasticidade e acurácia, o modelo permite reconfigurar todas as métricas de monitoramento sem a necessidade de reiniciar o sistema ou proporcionar algum impacto de desempenho para o ambiente. Principal problema encontrado, cada instancia de máquina virtual irá gerar uma instancia de coletor de dados e repositório local, o que gera um uso extra de memória, sem falar que é uma solução invasiva, pois insere software dentro da VM do usuário.

Andreozzi et al. (2005) propõem

REFERÊNCIAS

ACETO, G. et al. Cloud monitoring: A survey. **Computer Networks**, Elsevier, v. 57, n. 9, p. 2093–2115, 2013.

ACETO, G. et al. Cloud monitoring: Definitions, issues and future directions. **CLOUDNET**, v. 12, p. 63–67, 2012.

ANDREOZZI, S. et al. Gridice: a monitoring service for grid systems. **Future Generation Computer Systems**, Elsevier, v. 21, n. 4, p. 559–571, 2005.

APACHE. **Apache**. 2015. Acesso em 14 de setembro de 2015. Disponível em: <http://apache.org/>.

APACHE, f. **CloudStack**. 2015. Acesso em 14 de setembro de 2015. Disponível em: <http://cloudstack.apache.org/>.

ARMBRUST, M. et al. A view of cloud computing. **Communications of the ACM**, ACM, v. 53, n. 4, p. 50–58, 2010.

BEN, A. et al. **System x Virtualization Strategies**. [s.n.], 2010. Disponível em: <https://lenovopress.com/redp4480>.

BRäSCHER, G. B. **PROPOSTA DE UM FRAMEWORK PARA CONSOLIDAÇÃO DE RECURSOS EM AMBIENTES DE COMPUTAÇÃO EM NUVEM**. 2015.

CANONICAL. **Canonical**. 2015. Acesso em 24 de novembro de 2015. Disponível em: <http://www.canonical.com/about>.

CARON, E. et al. Auto-scaling, load balancing and monitoring in commercial and open-source clouds. 2012.

CLAYMAN, S.; GALIS, A.; MAMATAS, L. Monitoring virtual networks with lattice. In: IEEE. **Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP**. [S.l.], 2010. p. 239–246.

CLOUDSTACKZENPACK. **Zenoss**. 2015. Disponível em: <https://github.com/zenoss/ZenPacks.zenoss.CloudStack>.

CONTAINERS., L. **LXC**. 2015. Acesso em 24 de novembro de 2015. Disponível em: <https://linuxcontainers.org/lxc/introduction/>.

CORPORATION, I. **IA 32**. 2015. Acesso em 15 de outubro de 2015. Disponível em: <http://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-manual-325462.pdf>.

CORPORATION, I. **x8664**. 2015. Acesso em 15 de outubro de 2015. Disponível em: <http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html>.

DOSHI, G. **IA 64**. 2015. Acesso em 15 de outubro de 2015. Disponível em: <http://www.csee.umbc.edu/portal/help/architecture/idfisa.pdf>.

HALL, P. Opportunities for csps in enterprise-grade public cloud computing. **OVUM, May**, 2012.

HASSELMEYER, P.; D'HEUREUSE, N. Towards holistic multi-tenant monitoring for virtual data centers. In: IEEE. **Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP**. [S.l.], 2010. p. 350–356.

HP. **Eucalyptus**. 2015. Acesso em 14 de setembro de 2015. Disponível em: <http://www8.hp.com/us/en/cloud/helion-eucalyptus-overview.html>.

HP. **HP**. 2015. Acesso em 29 de setembro de 2015. Disponível em: <http://www.hp.com/>.

IBM. **Power PC**. 2015. Acesso em 15 de outubro de 2015. Disponível em: <http://www.ibm.com/developerworks/systems/library/es-archguide-v2.html>.

ICINGA. **Icinga**. 2015. Acesso em 30 de novembro de 2015. Disponível em: <https://www.icinga.org/>.

ICINGAOPENSTACKGIT. **OpenStack-Icinga**. 2015. Acesso em 30 de outubro de 2015. Disponível em: <https://github.com/hpcloud/icinga>.

INC., C. S. **Xen Project Software Overview**. 2015. Acesso em 10 de novembro de 2015. Disponível em: <http://wiki.xenproject.org/wiki/Xen_Overview>.

INC., O. **ORACLE**. 2015. Acesso em 10 de novembro de 2015. Disponível em: <http://www.oracle.com/>.

INTEL. **IPMI**. 2015. Acesso em 29 de novembro de 2015. Disponível em: <http://www.intel.com/content/www/us/en/servers/ipmi/ipmi-home.html>.

KATSAROS, G. et al. A self-adaptive hierarchical monitoring mechanism for clouds. **Journal of Systems and Software**, Elsevier, v. 85, n. 5, p. 1029–1041, 2012.

KATSAROS, G.; KUBERT, R.; GALLIZO, G. Building a service-oriented monitoring framework with rest and nagios. In: IEEE. **Services Computing (SCC), 2011 IEEE International Conference on**. [S.l.], 2011. p. 426–431.

KEPHART, J. O.; CHESS, D. M. The vision of autonomic computing. **Computer**, IEEE, v. 36, n. 1, p. 41–50, 2003.

LAPRIE, J.-C. From dependability to resilience. In: CITESEER. **38th IEEE/IFIP Int. Conf. On Dependable Systems and Networks**. [S.l.], 2008. p. G8–G9.

LINUX-KVM. **KVM**. 2015. Acesso em 14 de setembro de 2015. Disponível em: <http://www.linux-kvm.org/page/Main_Page>.

MAZEGEN. **x86**. 2007. Acesso em 15 de outubro de 2015. Disponível em: <http://x86asm.net/articles/x86-64-tour-of-intel-manuals/>.

MELL, P.; GRANCE, T. The nist definition of cloud computing. Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology Gaithersburg, 2011.

MENASCÉ, D. A. Virtualization: Concepts, applications, and performance modeling. In: **Int. CMG Conference**. [S.l.: s.n.], 2005. p. 407–414.

MICROSOFT. **NFS**. 2013. Acesso em 18 de novembro de 2015. Disponível em: <https://technet.microsoft.com/en-us/library/jj592688.aspx>.

MICROSOFT. **Hyper-V**. 2015. Acesso em 29 de setembro de 2015. Disponível em: <http://hyperv.veeam.com/what-is-hyper-v-technology/>.

MICROSOFT. **iSCSI**. 2015. Acesso em 18 de novembro de 2015. Disponível em: <http://windows.microsoft.com/pt-br/windows-vista/what-is-internet-small-computer-system-interface-iscsi>.

NAGIOS. **Nagios**. 2015. Acesso em 14 de setembro de 2015. Disponível em: <https://assets.nagios.com/downloads/nagioscore/docs/nagioscore-3-en.pdf>.

NAGIOS. **Nagios-CloudStack**. 2015. Acesso em 13 de novembro de 2015. Disponível em: <https://exchange.nagios.org/directory/Plugins/Cloud/nagios-2Dcloudstack/details>.

NAGIOS. **Nagios-OpenStack**. 2015. Acesso em 13 de novembro de 2015. Disponível em: <https://exchange.nagios.org/directory/Plugins/Cloud/openstack-infrastructure-monitoring/details>.

NET-SNMP. **SNMP**. 2015. Acesso em 30 de outubro de 2015. Disponível em: <http://www.net-snmp.org/>.

OPENSTACK. **OpenStack**. 2015. Acesso em 14 de setembro de 2015. Disponível em: <http://docs.openstack.org/index.html>.

OPENSTACKZENPACK. **Zenoss**. 2015. Acesso em 30 de outubro de 2015. Disponível em: <http://wiki.zenoss.org/ZenPack:OpenStack_(Provider_View)>.

ORACLE. **VirtualBox**. 2015. Acesso em 10 de novembro de 2015. Disponível em: <https://www.virtualbox.org/wiki/Documentation>.

POPEK, G. J.; GOLDBERG, R. P. Formal requirements for virtualizable third generation architectures. **Communications of the ACM**, ACM, v. 17, n. 7, p. 412–421, 1974.

PROJECT, N. **Nimbus**. 2015. Acesso em 30 de outubro de 2015. Disponível em: <http://www.nimbusproject.org/>.

QEMU. **QEMU**. 2015. Acesso em 7 de outubro de 2015. Disponível em: <http://wiki.qemu.org/Main_Page>.

REDHAT. **RedHatVirtualization**. 2015. Acesso em 14 de setembro de 2015. Disponível em: <http://www.redhat.com/en/technologies/virtualization/enterprise-virtualization>.

SAHOO, J.; MOHAPATRA, S.; LATH, R. Virtualization: A survey on concepts, taxonomy and associated security issues. In: IEEE.

Computer and Network Technology (ICCNT), 2010 Second International Conference on. [S.l.], 2010. p. 222–226.

SHAO, J. et al. A runtime model based monitoring approach for cloud. In: IEEE. **Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on**. [S.l.], 2010. p. 313–320.

SHIREY, R. W. Internet security glossary, version 2. 2007.

SINGH, G.; SOOD, S.; SHARMA, A. Cm-measurement facets for cloud performance. **International Journal of Computer Applications**, International Journal of Computer Applications, 244 5 th Avenue,# 1526, New York, NY 10001, USA India, v. 23, n. 3, p. 37–42, 2011.

SYSTEMS, C. **Citrix**. 2015. Acesso em 14 de setembro de 2015. Disponível em: <http://www.citrix.com/>.

SYSTEMS, C. **Xen Dom0**. 2015. Acesso em 15 de outubro de 2015. Disponível em: <http://wiki.xenproject.org/wiki/Dom0>.

SYSTEMS, C. **Xen DomU**. 2015. Acesso em 15 de outubro de 2015. Disponível em: <http://wiki.xenproject.org/wiki/DomU>.

SYSTEMS, C. **Xen XAPI**. 2015. Acesso em 15 de outubro de 2015. Disponível em: <http://wiki.xenproject.org/wiki/XAPI>.

SYSTEMS, C. **Xen XL**. 2015. Acesso em 15 de outubro de 2015. Disponível em: <http://wiki.xenproject.org/wiki/XL>.

URGAONKAR, B.; SHENOY, P.; ROSCOE, T. Resource overbooking and application profiling in shared hosting platforms. **ACM SIGOPS Operating Systems Review**, ACM, v. 36, n. SI, p. 239–254, 2002.

VIRATANAPANU, A. et al. On demand fine grain resource monitoring system for server consolidation. In: IEEE. **Kaleidoscope: Beyond the Internet?-Innovations for Future Networks and Services, 2010 ITU-T**. [S.l.], 2010. p. 1–8.

VMWARE. **Understanding Full Virtualization, ParaVirtualization, and Hardware Assist.** 2007. Acesso em 17 de outubro de 2015. Disponível em: <http://www.vmware.com/resources/techresources/1008>.

VMWARE. **ESXI**. 2015. Acesso em 14 de setembro de 2015. Disponível em: <http://www.vmware.com/products/esxi-and-esx/>.

VMWARE. **vCenter**. 2015. Acesso em 10 de novembro de 2015. Disponível em: <http://www.vmware.com/br/products/vcenter-server/>.

VMWARE. **VMware**. 2015. Acesso em 14 de setembro de 2015. Disponível em: <http://www.vmware.com>.

WANG, C. et al. A flexible architecture integrating monitoring and analytics for managing large-scale data centers. In: ACM. **Proceedings of the 8th ACM international conference on Autonomic computing**. [S.l.], 2011. p. 141–150.

WEINGÄRTNER, R.; BRÄSCHER, G. B.; WESTPHALL, C. B. Cloud resource management: A survey on forecasting and profiling models. **Journal of Network and Computer Applications**, Elsevier, v. 47, p. 99–106, 2015.

ZABBIX. **ZABBIX**. 2015. Acesso em 30 de novembro de 2015. Disponível em: <http://www.zabbix.com/product.php>.

ZABBIX-CLOUDSTACK. **Zabbix-Cloudstack**. 2015. Acesso em 30 de outubro de 2015. Disponível em: <https://github.com/ke4qqq/zabbix-cloudstack>.

ZABBIXAGENTADOPTION. **Zabbix agent adoption**. 2015. Acesso em 30 de outubro de 2015. Disponível em: <https://wiki.openstack.org/wiki/Zabbix-agent-adoption>.

ZENOSS. **Zenoss Wiki**. 2015. Acesso em 10 de novembro de 2015. Disponível em: <http://wiki.zenoss.org/Main_Page>.