UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Pedro Henrique Pereira Martins

PROPOSTA DE UM FRAMEWORK 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),(SINGH; SOOD; SHARMA, 2011),(FORUM, 2015).

Com o aumento da demanda por serviços prestados pela nuvem, há um constante crescimento da infraestrutura da nuvem de modo a atender a demanda (SINGH; SOOD; SHARMA, 2011; 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),(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 4 partes, monitoramento, análise, planejamento e execução. Nesta definição, o elemento deve ser capaz de se monitorar constantemente afim de verificar se está fazendo aquilo o que foi designado como foi designado. Com os dados colhidos na etapa de monitoramento, o elemento irá analizar os dados e com bases nesses dados, ele irá realizar ajustes para melhorar seu desempenho ou corrigir possíveis fugas do seu fluxo correto de execução.

O monitoramento constante da nuvem é comumente usado 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 destes ambiêntes. 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. Primeiro deve-se levar em conta o grande número de serviços e usuários, que torna a nuvem um ambiente com uma gama grande de informação

para ser processada (SHAO et al., 2010), (CLAYMAN; GALIS; MAMATAS, 2010).

Outro fator que contribui para a complexidade de monitorar os ambientes de nuvem é a heterogeneidade dos componentes da nuvem, os componentes da nuvem não são necessáriamente dos mesmos fabricantes ou da mesma arquitetura. A nuvem pode ser constituída de diferentes clusters, cada um com seus respectivos processadores, memória, placa de rede e periféricos. Logo deve-se ter em mente que será necessário implementar meios para monitorar cada um desses clusters (SHAO et al., 2010). Além disso, esses clusters podem estar em localidades diferentes, até mesmo em países diferentes, o que gera a necessidade de ter um sistema de monitoramento distribuído, com nodos colhendo informações em cada cluster e enviando para algum repositório de dados.

Outro ponto crucial é de que 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 recuros em nuvem (SHAO et al., 2010), (CLAYMAN; GALIS; MAMATAS, 2010). Além disso, após implantado o sistema de monitoramento, qualquer modifição necessária deve ser feita sem a necessidade de desligar ou reiniciar o ambiente (CLAYMAN; GALIS; MAMATAS, 2010).

Problemas resultantes de um monitoramento mau sucedido são:

1. A queda de desempenho da nuvem devido a sobrecargas de processamento com monitoramento;

2. Ações de gerência mal sucedidas devido a dados incosistentes advindos do processo de monitoramento;

3. Perda de dados e/ou obtenção de dados inconsistentes 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 por usuários ou pela nuvem. O que estas ferramentas costumam adotar é o monitoramento momentâneo do ambiente, sem a criação de um histórico de uso de recursos, para indicar aos administradores o estado atual do ambiente.

O que existe atualmente para monitoramento são extenções adicionadas as ferramentas de orquestração, porém estas extenções são de

difícil instalação e 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 (ZENPACK, 2015) é um exemplo desse tipo de ferramenta de monitoramento que apresentam alguns problemas. No caso do Zenoss, e sua extensão para monitoramento do CloudStack (APACHE, 2015b), existe um problema no monitoramento de uso de memória do servidor físico, a ferramenta faz uma query do uso de memória para o Xen (INC., 2015) e já projeta o retorno desta query como se fosse o uso total de memória pelo servidor. O problema é que esta query do Xen, retorna apenas o uso de memória relativo ao dom0, que é apenas uma máquina virtual, e não o servidor físico inteiro. Assim o Zenoss acaba fornecendo dados de apenas uma máquina virtual específica e não do ambiente inteiro.

Ao atuar nessa área, deve-se usar boas técnicas de manipulação de dados devido a grande gama de informações que podem ser monitorados na nuvem. Ser o menos invasivo possível, ou seja, ser invisível para os usuários da 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 atual do ambiente forem disponibilizadas, mais eficientemente se conseguirá administrar a nuvem.

1.1 MOTIVAÇÃO

O monitoramento é de fundamental importância para a nuvem, pois proporciona um melhor entendimento do estado em que o ambiente se encontra; do estado das aplicações que estão hospedadas no ambiente; do uso/disponibilidade de recursos físicos do ambiente, e também para determinar se os acordos de prestação de serviço estão sendo devidamente cumpridos.

Aceto et al. (2012) define uma série de características que uma ferramenta de monitoramento ideal deve possuir. Em (ACETO et al., 2013), são levantadas características apresentadas pelas ferramentas de monitoramento atualmente usadas.

Como apresentado em (ACETO et al., 2013), existe uma grande carência de autonomia da parte das ferramentas de monitoramento atuais. Assim, toda vez que novos recursos são inseridos no ambiente, deve-se informar às ferramentas que tais recursos devem ser monitora-

dos, quando o ideal era a ferramenta detectar e passar a monitorar os recursos novos autonomamente.

Outro ponto sobre as ferramentas atuais, é que a proposta delas em sua maioria, é monitorar desde a parte de aplicação até a parte de infraestrutura. Com isso, elas acabam se tornando mais complexas, com uma instalação mais 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 a ferramenta de orquestração cloudStack (APACHE, 2015b) 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 a ferramenta CloudStack, possa fornecer todo o suporte para o monitoramento da infraestrutura do ambiente de nuvem sem interferência nos serviços por ela já fornecidos.

2 OBJETIVOS GERAIS

3 OBJETIVOS ESPECÍFICOS

4 FUNDAMENTACAO

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

4.1 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).

4.1.1 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 - a nuvem deve estar sempre disponível para o usuário pela internet independente da plataforma de acesso (tablet, smartphone, desktop etc);

3. Recursos compartilhados - os recursos fisicos 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 desalocar 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.

4.1.2 Modelos de serviço

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

1. Software como um serviço: a nuvem fornece uma aplicação final via rede para usuários usarem. O usuário não tem controle algum sobre a infraestrutura que hospeda a aplicação ou da plataforma que a aplicação usa.

2. Plataforma como um serviço: a nuvem disponibiliza 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.

3. Infraestrutura como um 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 rodar em cada plataforma, porém a parte de hardware ele não tem controle, endereços físicos de memória e algumas instruções do processador não serão acessíveis ao usuário.

4.1.3 Tipos de nuvens

Em (MELL; GRANCE, 2011) são definidos quatro tipos de nuvem quanto a disponibilizade dos recursos para usuários:

1. Nuvem privada : a infraestrutura da nuvem é disponibilizada apenas para uma única organização utilizar. A nuvem pode ser gerenciada pela própria organização, por um terceiro ou por ambos.

2. Nuvem comunitárias : a infraestrutura da nuvem é disponibilizada para uma comunidade de organizações que tem interesses em comum. A nuvem pode ser gerenciada por alguma(s) dessas organizaçõe(s), terceiros ou ambos.

3. Nuvem pública : a nuvem é aberta para todo o público. A é gerenciada por alguma empresa, organização governamental/acadêmica ou alguma combinação destas.

4. Nuvem hibrida : a infraestrutuda da nuvem é um composto de duas ou mais infraestruturas (privada, comunitária, pública).

4.2 VIRTUALIZAÇÃO

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

O processo de virtualização consiste na existência de um virtualizador que gerencia sistemas operacionais convidados e intermedia toda a comunicação entre hardware e esses sistemas (MENASCÉ, 2005). Sistemas operacionais convidados rodam encima do virtualizador como se fossem aplicações em um sistema operacional qualquer. Portanto os sistemas operacionais convidados, não possuem acesso direto ao hardware.

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 $VirtualMachine$ (VM - Em português, máquina virtual). Os sistemas operacionais e aplicações dentro das máquinas virtuais, enchergam 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).

4.2.1 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, basta migrar a VM para outro servidor físico. O custo de ter um máquina virtual é menor do que o custo para ter uma máquina física com a mesma configuração (SAHOO; MOHAPATRA; LATH, 2010).

Como problemas temos uma grande sobrecarga de processamento para gerenciar a inúmera quantidade de máquinas virtuais. O desenvolvimento de virtualizadores não é algo simples e é de inúmera importância ter redundância de hardwares pois se um componente de hardware der defeito, todas as VMs vinculadas a tal componente serão

penalizadas (POPEK; GOLDBERG, 1974) .

4.2.2 Tipos de virtualização

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

1. Virtualização por Conteiner: este tipo de virtualização precisa de um sistema operacional como base. Neste caso, o virtualizador irá criar instâncias virtuais do sistema operacional base isoladas lógicamente. 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 divirem 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.

2. Virtualização do tipo 1: este tipo de virtualização roda diretamente encima do hardware, neste caso o virtualizador irá gerar abstrações de hardware completas para poder hospedar um sistema operacional qualquer. Isto gera um consumo extra de memória, pois cada instância irá ter seu próprio sistema operacional mapeado em memória. Este tipo de virtualização, possui a melhor performance de processamento dentre os tipos aqui apresentados. Neste tipo de virtualização, os clientes podem escolher seus próprios sistemas operacionais

3. Virtualização do tipo 2: esta virtualização assim como a virtualização por conteiner, 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 idependentemente de qual é o sistema operacional base. O virtualizador não tem acesso direto ao hardware neste tipo de virtualização, 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 hardware. Esse tipo de virtualização é um meio termo entre a virtualização do tipo 1 e por conteiner.

Cada tipo de virtualização possui vantagens e desvantagens. O tipo 1, é o tipo 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 tira parte da complexidade do virtualizador, porém acaba prejudicando o desempenho devida a camada de software extra para capacitar a virtualização dos componentes de hardware. A virtualização por conteiner é o tipo com menor uso de memória, porém ela amarra os usuários a poderem usar apenas VMs com o mesmo sistema operacional que o sistema base onde roda o virtualizador.

4.2.3 Virtualizadores

O virtualizador é o 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 $VirtualMachineMonitor$ (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.

Dentre os virtualizadores existentes, foram estudados nesse trabalho o Xen (INC., 2015), KVM (LINUX-KVM, 2015) e ESXi (VMWARE, 2015a)

Xen : 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 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.

KVM : o KVM 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 um 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 totalmente dependente de uma ferramenta de orquestração que dê suporte a gerência de servidores dentro de um cluster.

ESXi : O ESXi 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. Dentre os virtualizadores analizados, será usado neste tra-

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

balho 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 em seu código e possa contribuir com melhorias tanto na documentação quanto em correções de bugs.

4.3 MONITORAMENTO EM CLOUD

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 na nuvem.

4.3.1 Caracteristicas de ferramentas de monitoramento na nuvem

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

1. Escalabilidade: a ferramenta pode ser considerada escalável se ela consegue suportar uma larga escala 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 na nuvem, pois o ambiente vive em contínua mudança com a adição e remoção de recursos dinâmicamente;

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

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 pelo administrador do ambiente (WANG et al., 2011). Esta característica depende diretamente do desenvolvimento apropriado das outras características;

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 a nuvem está em constante atividade (SHIREY, 2007);

11. Precisão: a ferramenta é considerada precisa se as informações disponibilizadas pela ferramenta devem conter valores que expressem o mais aproximadamente possível o estado atual 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. Que são:

- Grande gama de informação: a nuvem possui diversos usuários, assim como componentes e um grande tráfego de rede. Com isso, armazenar informações colhidas na nuvem é uma tarefa nada fácil. 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 tarefa não é nada trivial, ela 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). Realizar este processo tentando gerar o menor impacto possível no ambiente é o grande desafio.

- Heterogeneidade do ambiente: a nuvem é um ambiente que possui inúmeros dispositivos com diferentes arquiteturas e tecnologias. Conseguir prover suporte de monitoramento para cada um desses dispositivos e ainda manter um bom isolamento entre eles, não é algo fácil.

- Rastreamento de elementos da nuvem: a nuvem permite que seus recursos sejam migrados de um dispositivo físico para outro. 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 migrado e a detecção de num novo recurso, é algo complicado.

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 estão rodando na 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).

Nesta seção, foram mostradas as características que uma ferramenta de monitoramento deve possuir segundo Aceto et al. (2012). Além disso, foram mostradas as dificuldades que são enfrentadas para implementar tais características nas ferramentas de monitoramento levando em conta as características de ambientes de computação em nuvem. Também foram apresentados algumas importâncias do monitoramento para a computação em nuvem.

4.3.2 Ferramentas de monitoramento

Nesta seção, serão abordadas algumas ferramentas de monioramento 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. O que permite que o Nagios consiga monitorar sistemas do tipo Eucalyptus (Elastic Utility Computing Architecture for Linking Your Programs To Useful System) (NAGIOS, 2015) . Nagios é uma das ferramentas usadas para prover o monitoramento do OpenStack (OPENSTACK, 2015), que é uma ferramenta de orquestração de infra-estruturas de computação em nuvem. A principal proposta do Nagios é a extensibilidade.

- CloudStack ZenPack: o CloudStack (APACHE, 2015b), é uma ferramenta de orquestração de infra-estrutura de ambientes de computação em nuvem. É uma ferramenta opensource escrita em java com suporte a diversos virtualizadores, como por exemplo, KVM, XenServer e VMware. Usuários do CloudStack, tem como opção para o monitoramento do ambiente, uma extenção (ZENPACK, 2015), disponível no ZenPack. Esta extensão fica limitada pelas funcionalidades disponibilizadas pelo CloudStack. Como principal caracteristica dessa ferramenta, se encontra a pontualidade

- Nimbus: a Nimbus (PROJECT, 2015) é uma plataforma de código aberto que da suporte a diferentes infra-estruturas de nuvens. Esta ferramenta é voltada especialmente para projetos relacionados a comunidade científica. A Nimbus, visa autonomia no

monitoramento tanto de aplicações da nuvem quanto na infra-estrutura como um todo, no caso, ela não suporta monitoramento de clusters, apesar de monitorar aplicações rodando em VMs.

- ZABBIX: o ZABBIX (ZABBIX, 2015) é uma ferramenta de monitoramento de infra-estrutura 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 de ambientes que usem o vCenter. Além de ser compatível com o vCenter, o ZABBIX permite o monitoramento de qualquer dispositivo de hardware que suporte IPMI (INTEL, 2015) e dispositivos de rede que suportem SNMP (NET-SNMP, 2015), conseguindo capturar as mesmas métricas suportadas por ambos os protocolos IPMI e SNMP.

- Icinga: o Icinga (ICINGA, 2015) é uma ferramenta que pode ser instalada em qualquer distribuição Linux e em algumas distribuições Unix. 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.

Dentre as ferramentas analizadas, todas suportam monitoramento de recursos de hardware usando protocolos padrões de monitoramento como SNMP e IPMI. E algumas dessas ferramentas, como o Zennos e o ZABBIX, além de usar estes protocolos, usam também apis de ferramentas de orquestração. Alem disso, uma característica em comum dessas ferramentas, é a abordagem de coletar os dados. 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 enviar para esta entidade.

4.4 ORQUESTRAÇÃO DA NUVEM

As ferramentas de orquestração na nuvem servem para abstrair do administrador toda a parte de interação com os virtualizadores,

serviços de armazenamento, rede e outros. Permitindo que o administrador não precise de amplos conhecimentos nas tecnologias 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 facilmente.

4.4.1 Ferramentas de orquestração

Dentre as ferramentas de orquestração existentes, foram analizadas algumas

- 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 (LINUX-KVM, 2015), VMware (VMWARE, 2015c), XenServer (INC., 2015) 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 bem descrita.
- É codificada em java, que é uma linguagem de programação bem conhecida para futuras extenções.
- Da suporte para sistemas de arquivos conhecidos, o que diminui o tempo que seria gasto com familiarizações com sistemas de arquivos.
- É uma ferramenta de código aberto, o que possibilita um melhor conhecimento da ferramenta e implantação de melhorias.

As ferramentas de orquestração proporcionam mais facilidades para administradores gerenciarem o ambiente de computação em nuvem e para usuários usarem os recursos do ambiente. Essas facilidades tornam a interação entre usuário e ambiente, mais amigável. Para proporcionar essas facilidades de interação com o ambiente, as ferramentas de orquestração acabam se tornando complexas pois precisam cuidar de toda a comunicação com os virtualizadores, e cada virtualizador tem seu modo de operar.

Dentre as ferramentas mostradas nessa seção, o CloudStack foi escolhido por ter o código aberto, ser escrito em Java, suportar sistemas de arquivos conhecidos e possuir uma comunidade amplamente ativa.

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.

APACHE. **Apache**. 2015. Disponível em: <http://apache.org/>.

APACHE, f. **CloudStack**. 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>.

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.

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

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

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

FORUM, C. I. **UK Cloud Adoption Trends for 2015 InsightBrief**. 2015. Disponível em: <http://cloudindustryforum.org/downloads/whitepapers/UK-Cloud-Adoption-Trends-for-2015-InsightBrief.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. Disponível em: <http://www8.hp.com/us/en/cloud/helion-eucalyptus-overview.html>.

HP. **HP**. 2015. Disponível em: <http://www.hp.com/>.

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

ICINGA. **Icinga**. 2015. Disponível em: <https://www.icinga.org/>.

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

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

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. Disponível em: <http://www.linux-kvm.org/page/Main_Page>.

MAZEGEN. **x86**. 2007. 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. Disponível em: <https://technet.microsoft.com/en-us/library/jj592688.aspx>.

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

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

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

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

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

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. Disponível em: <http://www.nimbusproject.org/>.

QEMU. **QEMU**. 2015. Disponível em: <http://wiki.qemu.org/Main_Page>.

REDHAT. **RedHatVirtualization**. 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. Disponível em: <http://www.citrix.com/>.

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

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

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

SYSTEMS, C. **Xen XL**. 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. Disponível em: <http://www.vmware.com/resources/techresources/1008>.

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

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

VMWARE. **VMware**. 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. Disponível em: <http://www.zabbix.com/product.php>.

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