UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Fernando Schubert

UMA ARQUITETURA DE COMPUTAÇÃO AUTÔNOMA E COGNITIVA PARA MONITORAMENTO DE NUVENS

Florianópolis

2014

Fernando Schubert

UMA ARQUITETURA DE COMPUTAÇÃO AUTÔNOMA E COGNITIVA PARA MONITORAMENTO DE NUVENS

Dissertação submetida ao Programa de Pós-graduação em Ciência da Computação para a obtenção do Grau de Mestre em Ciência da Computação.
Orientador: Prof. Dr. Carlos Becker Westphall

Florianópolis

2014

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Schubert, Fernando
Uma arquitetura de computação autônoma e cognitiva para monitoramento de nuvens / Fernando Schubert ; orientador, Carlos Becker Westphall - Florianópolis, SC, 2014.
112 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Ciência da Computação.

Inclui referências

1. Ciência da Computação. 2. Redes de computadores. 3. Computação em nuvem. 4. Sistemas especialistas. I. Westphall, Carlos Becker. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Ciência da Computação. III. Título.

Fernando Schubert

UMA ARQUITETURA DE COMPUTAÇÃO AUTÔNOMA E COGNITIVA PARA MONITORAMENTO DE NUVENS

Esta Dissertação foi julgada aprovada para a obtenção do Título de "Mestre em Ciência da Computação", e aprovada em sua forma final pela Programa de Pós-graduação em Ciência da Computação.

Florianópolis, 15 de Agosto 2014.

Prof. Ronaldo dos Santos Mello, Dr.
Coordenador do Curso

Banca Examinadora:

Prof. Carlos Becker Westphall, Dr.
Presidente

Prof. Dr. Carlos Becker Westphall
Orientador

Prof[a]. Carla Merkle Westphall

Para as mulheres da minha vida, minha mãe e esposa, que nunca deixaram de acreditar e me incentivar a seguir em frente. Obrigado!

AGRADECIMENTOS

O problema da seção de Agradecimentos é cometer a injustiça de esquecer de mencionar pessoas importantes na construção da obra. Porém algumas são de tal importância que necessitam ser mencionadas. Perdão aos demais, mas meu sincero agradecimento vai primeiramente para o meu orientador, Prof. Westphall, pela paciência, incentivo e apoio prestados nesta caminhada, sempre com palavras de motivação para não esmorecer em um processo de gestação complicado como o da presente dissertação. Não posso deixar de mencionar o Doutorando Rafael Mendes pelas incontáveis horas de discussões técnicas, teóricas e filosóficas, pelo auxílio valioso nas definições e embasamento científico do trabalho e pelos colegas do Laboratorio de Redes e Gerência pela construção da nuvem privada de pesquisa e auxilio no suprimento de necessidades técnicas. Por último, mas com a maior importância agradeço ao Criador Deus, Senhor de todo o conhecimento e sabedoria pelo dom da vida e pela oportunidade de concluir tamanha empreitada, toda honra e glória ao Deus Trino.

IN HOC SIGNO VINCES

Constantino

RESUMO

Em um curto espaço de tempo, a computação em nuvem evoluiu de mais um *buzzword* do mercado para um paradigma e modelo de serviço e entrega de recursos amplamente consolidado e aceito. Mesmo que ainda existam lacunas e divergências sobre conceitos e padrões, fato inegável é a expansão e utilização da Nuvem como modelo computacional. Neste contexto novos desafios surgem, entre eles a necessidade de monitorar tais infraestruturas complexas e heterogêneas, de forma a possibilitar a análise de grandes volumes de dados gerados, para tarefas de importância primordial para este modelo, como faturamento de recursos utilizados, identificação de falhas e predição de comportamentos. Tendo em vista este universo dinâmico e de rápidas mudanças, a presente arquitetura é apresentada, propondo um modelo de monitoramento não intrusivo, cujo foco está no armazenamento e recuperação de dados para a construção de sistemas autônomos utilizando aprendizado de máquina. Este trabalho visa evoluir o estado da arte ao propor uma arquitetura autônoma para o monitoramento de Nuvens, tanto privadas quanto híbridas e públicas.

Palavras-chave: Computação em Nuvem. Computação Autônoma. Aprendizado de máquina. Computação Cognitiva.

ABSTRACT

In a very short timespan, Cloud Computing has evolved from another market buzzword to a widely accepted and consolidated computing model. Even though there are still gaps and disagreements about concepts and patterns, an undeniable fact is the expansion and wide use of the Cloud as a computing model. In this context there are new challenges, including the need to monitor such complex and heterogeneous infrastructures, in order to enable the analysis of large volumes of data generated for tasks of paramount importance like billing, consolidated reports, detect failures and predict future issues. Given this dynamic and rapidly changing universe, the proposed architecture is presented, proposing a non-intrusive monitoring model, whose focus is the information storage and retrieval allowing the construction of autonomous systems using machine learning. This work aims to advance the state of the art by proposing an autonomous architecture for Clouds monitoring.

Keywords: Cloud Computing. Autonomic Computing. Big Data. Machine Learning. Cognitive Computing.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

LISTA DE SÍMBOLOS

SUMÁRIO

1 INTRODUÇÃO

> A computação em nuvem tornou-se a concretização de um objetivo buscado há muito tempo pela indústria de TI, da computação como utilidade. Este novo modelo tem a capacidade de mudar toda a forma de produção e implantação de software, além de guiar os rumos de como o hardware é desenvolvido e adquirido (ARMBRUST et al., 2009)

A atual relevância e crescimento da adoção da computação em nuvem é o resultado dos avanços da área nas últimas décadas consolidando as técnicas e modelos que possibilitaram o surgimento do que chamamos de computação em nuvem.

Desde visionários como John McCarthy que predisse em 1961 que no futuro, *"a computação seria organizada como uma utilidade pública"* (WEI; BLAKE, 2010), um bem comum, mensurado e faturado de acordo com a sua utilização, originando o conceito de computação utilitária, até avanços mais recentes como a massificação da virtualização, desenvolvimento das grades computacionais e aumento do poder de processamento de clientes e servidores, entre outros avanços (FOSTER et al., 2008). Todas estas contribuições foram relevantes na constituição do paradigma de coputação em nuvem.

Portanto, a computação em nuvem não pode ser considerada uma tecnologia em si, mas sim um conceito ou paradigma computacional da mesma forma que as arquiteturas mainframe e posteriormente a arquitetura PC desempenharam na história da computação.

Como todo paradigma emergente, o crescimento na adoção e consolidação da computação em nuvem tem levantado diversos desafios de pesquisa para satisfazer plenamente os seus conceitos. De acordo com (MELL; GRANCE, 2011) a computação em nuvem pode ser definida como um modelo que possibilita acesso em rede ubíquo e sob demanda a um conjunto de recursos computacionais configurável (redes, servidores, armazenamento, aplicações, serviços) que podem ser rapidamente provisionados e liberados com um esforço mínimo de gerenciamento ou interação do provedor de serviços.

Buyya refere-se à computação em nuvem como a quinta utilidade, juntando-se à utilidades tradicionais como água, eletricidade, telefonia e aquecimento, onde a revolução e a mudança de paradigma proporcionada a indústria de TI irá afetar toda a forma como se pensa

e fornece tecnologia (DUNCAN et al., 2009).

1.1 MOTIVAÇÃO

Apesar de todo este crescimento e adoção por parte de entidades públicas e privadas, a computação em nuvem ainda está na sua fase inicial, e apresenta diversos tópicos relevantes para pesquisa. Em (ZHANG; CHENG; BOUTABA, 2010),(ULLAH; ZHENG, 2013) e (MORENO-VOZMEDIANO; MONTERO; LLORENTE, 2013) vários desafios de pesquisa na área de CN são analisados, entre eles a necessidade de melhores ferramentas para monitoramento, análise e gerenciamento das nuvens.

Figura 1 – Propriedades de sistemas para o monitoramento de Nuvem.

A necessidade de monitoramento e as características principais

de um sistema de monitoramento para a nuvem é definido na taxonomia de (ACETO et al., 2013c), analisando os atuais arcabouços de monitoramento e avaliando suas funcionalidades de acordo com as propriedades básicas a um sistema de monitoramento para a nuvem.

As principais características definidas por (ACETO et al., 2013c) estão apresentadas brevemente na Figura 1, e serão explicadas em detalhe no Capítulo 3.

O monitoramento tem historicamente um papel fundamental no gerenciamento e controle de sistemas computacionais. Com a rápida e ampla adoção da computação em nuvem, atividades de monitoramento granular e acurado são fundamentais para a operação eficiente do ambiente complexo, heterogêneo e multi-camadas que a nuvem tem se tornado.

Neste contexto, o monitoramento assumiu um papel de ainda maior relevância. Se no âmbito das redes e administração de sistemas o monitoramento era importante na detecção de problemas como gargalos de rede, consumo de recursos ou disponibilidade dos serviços, na Nuvem esta importância é ainda maior. Devido a características inerentes à nuvem como o faturamento apenas pelo consumo efetivo e ampliação ou redução da capacidade instalada de forma instantânea, o monitoramento passou a ser vital, tanto para provedores quanto para seus consumidores, pois o monitoramento dos recursos utilizados passa a ser fundamental para o faturamento dos recursos e o controle da utilização e na tomada de decisão para o gerenciamento de capacidade.

1.2 JUSTIFICATIVA

Conforme (ACETO et al., 2013c) as plataformas e ferramentas tradicionais para monitoramento, comumente utilizadas para o monitoramento de redes não satisfazem as propriedades de um sistema de monitoramento para nuvens, da mesma forma, as ferramentas e plataformas de monitoramento que se destinam especificamente a nuvens apesar de apresentar melhor aderência, ainda carecem atender às propriedades apresentadas na Figura 1. Ademais, o expressivo crescimento no volume e complexidade de dados de monitoramento necessita de tratamento aderente às necessidades dos provedores e consumidores da nuvem, como análise em tempo real de fluxos de dados, mineração e extração de informações relevantes ao contexto e construção de conhecimento.

Portanto, o presente trabalho visa abordar este problema, da

falta de aderência das atuais ferramentas e sistemas de monitoramento para nuvens ao propor uma arquitetura modular de monitoramento, partindo de uma abordagem autônoma, avaliando métricas de monitoramento, armazenamento, recuperação e técnicas de análise de grandes conjuntos de dados para atender a um conjunto maior de propriedades de monitoramento e adequar-se a esta nova realidade onde existe um número elevado de provas e um conjunto de dados massivo com necessidades estritas de tempo para sua análise e apresentação.

1.3 OBJETIVOS

Esta dissertação procura avançar o estado da arte sobre o problema do monitoramento no contexto da computação em nuvem: a falta de aderência das ferramentas e sistemas de monitoramento desenvolvidos em código aberto atuais em relação as propriedades e características de um sistema de monitoramento para nuvens . Este problema é apresentado por (ACETO et al., 2013c) em sua taxonomia sobre monitoramento na nuvem, onde a análise de diversas ferramentas abertas para monitoramento apresentaram resultados pouco satisfatórios em relação a satisfação das propriedades necessárias para um sistema ou arquitetura de monitoramento para a nuvem.

Propõe-se, a definição de uma arquitetura modular baseada em computação autônoma para o monitoramento de nuvens, com o objetivo de satisfazer as propriedades de monitoramento propostas por (ACETO et al., 2013c).

Além de utilizar conceitos de computação autônoma, a arquitetura proporcionará o uso de técnicas de armazenamento e recuperação de grandes conjuntos de dados e algoritmos para a análise destes dados (comumente conhecido como *Big Data*) para o processamento e armazenamento destes conjuntos de dados.

Devido a limitações de escopo, o presente trabalho abrangerá apenas o conceito de auto-gerenciamento e as etapas de monitoramento e análise do ciclo autônomo.

1.3.1 Objetivos Específicos

As principais contribuições deste trabalho são:

- Apresentar os principais conceitos relacionados com computação em nuvem, computação autônoma e cognitiva, e trabalhos rela-

cionados com o monitoramento em CN, abordando o estado da arte de cada elemento envolvido na arquitetura proposta;

- Definir um modelo de armazenamento que possibilite a recuperação e análise de grandes quantidades de dados *Big Data*;
- Apresentar uma arquitetura-modelo, baseada no ciclo autônomo para o monitoramento em CN;
- Analisar os dados coletados de acordo com SLAs estabelecidos para a validação de contratos de serviço;
- Propor um modelo de gerenciamento autônomo que valide o estado do sistema de monitoramento e proponha correções.

1.4 ORGANIZAÇÃO DA DISSERTAÇÃO

A presente dissertação está organizada da seguinte forma:

- Capítulo 1. Introduz o tema principal da dissertação abordando o problema, origem, objetivos gerais e específicos;
- Capítulo 2. Apresenta os conceitos, teoremas e discorre sobre os principais temas abrangidos pela dissertação, tais como: características essenciais, modelos de serviço e arcabouços de gerenciamento e modelos de implementação da computação em nuvem; conceitos e ciclo autônomo; computação cognitiva; banco de dados NoSQL e (*Big Data*);
- Capítulo 3. Neste Capítulo é apresentada uma revisão do estado da arte e trabalhos relacionados a monitoramento na Computação em Nuvem;
- Capítulo 4. A arquitetura proposta é explicada detalhadamente neste capítulo;
- Capítulo 5. Implementa a arquitetura proposta e abrange uma explanação do ambiente aplicado;
- Capítulo 6. Conclusão.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 COMPUTAÇÃO EM NUVEM

Através da história da computação houveram várias mudanças de paradigma. Dos mainframes para minicomputadores, dos minicomputadores ao microprocessamento e do microprocessamento aos computadores em rede (GRIMES; JAEGER; LIN, 2009). O advento da computação em nuvem deve ser considerado um fato muito mais complexo e importante do que simplesmente mais uma tendência tecnológica, pois se constitui na mudança de paradigma na computação, transformando não apenas a forma como produtos são entregues e mantidos, mas sim toda a cadeia produtiva da industria de tecnologia da informação.

Entre os diversos conceitos de computação em nuvem, o conceito apresentado por (BUYYA et al., 2009), onde a computação em nuvem é definida como sendo a quinta utilidade será adotado. Este conceito apresenta CN como sendo um tipo de sistema paralelo e distribuído consistindo em uma coleção de recursos virtualizados interconectados que podem ser dinamicamente provisionados e apresentados como um ou vários recursos computacionais unificados, baseados em contratos de nível de serviço estabelecidos entre o consumidor e o provedor de serviços (BUYYA et al., 2009).

Adicionalmente a (BUYYA et al., 2009), as definições apresentadas em (MELL; GRANCE, 2011), uma das conceituações mais aceitas no contexto acadêmico (ACETO et al., 2013c),(HOEFER; KARAGIANNIS, 2010),(KHAJEH-HOSSEINI; SOMMERVILLE; SRIRAM, 2010) será abordada no que tange a explicação das características e propriedades de um sistema de nuvem.

2.1.1 Características Essenciais

As características principais para que um sistema seja considerado Nuvem de acordo com (MELL; GRANCE, 2011) são:

- Acesso sob demanda: um consumidor pode provisionar recursos computacionais sem a intervenção de terceiros, de um provedor ou de agentes humanos.
- Acesso amplo a rede: os recursos estão disponíveis a partir de redes e acessíveis de mecanismos padronizados possibilitando acesso

a uma ampla gama de dispositivos e plataformas clientes.

- Associação de recursos (*pooling*): os recursos computacionais do provedor são associados para servir a múltiplos clientes, em um modelo *multi-tenant*, com diferentes recursos físicos e virtuais associados e desassociados dinamicamente de acordo com a demanda do cliente. Existe também um desprendimento da localização geográfica do recurso, ficando sob a responsabilidade do provedor gerenciar e distribuir seus recursos em diferentes regiões geográficas sem a ciência do consumidor.

- Elasticidade: é a característica da Nuvem em se expandir rapidamente, ou seja, ampliar a capacidade dos recursos, de forma dinâmica ou com mínimo esforço de gerenciamento para atender uma demanda especifica, e da mesma forma desalocar recursos quando não são mais necessários.

- Mensurável: o monitoramento e a capacidade de mensurar os recursos disponíveis e alocados é uma característica essencial das Nuvens, pois através dele atividades como alocação e ampliação da capacidade são efetuadas bem como a própria bilhetagem e cobrança do serviço.

2.1.2 Modelos de Serviço

A computação em nuvem é oferecida em diferentes modelos de serviço. Cada modelo apresenta uma abstração em relação ao anterior, visando oferecer recursos de maneira transparente, através de interfaces definidas, ocultando aos usuários a complexidade que está incluída no modelo. As principais camadas oferecidas são as de IaaS, PaaS e SaaS.

Os modelos de serviço nada mais são que abstrações dos diferentes meios em que a computação em nuvem é oferecida, aglutinando e separando-o em camadas ou modelos que além de definir produtos e serviços inerentes à sua camada se destinam geralmente a contextos e nichos de mercado diferente. Por exemplo, o foco da camada de IaaS é prover recursos básicos como processamento e armazenamento, geralmente na forma de instâncias virtuais executando algum sistema operacional básico. Já a camada de PaaS destina-se a proporcionar uma camada de desenvolvimento ou uma plataforma que suporte certo tipo de aplicações ou linguagens. O SaaS possibilita ao usuário da aplicação ter acesso a partir da rede aos seus dados e aplicativos em qualquer local e dispositivo.

Tabela 1 – Classificação dos modelos de serviço em computação em nuvem de acordo com (BUYYA; PANDEY; VECCHIOLA, 2009)

Categoria	**Características**	**Tipo de produto**	**Produtos e Fornecedores**
SaaS	Os consumidores são providos de aplicações que são acessíveis em qualquer horário e qualquer lugar.	Aplicações Web e serviços online (Web 2.0, Streaming, etc)	Salesforce.Com (CRM), Clarizen.com (Gerenciamento de projetos), GMail, Gdocs (escritório e e-mail).
PaaS	Os consumidores recebem uma plataforma para o desenvolvimento de aplicações hospedadas na Nuvem.	APIs para programação e arcabouços de desenvolvimento.	Google AppEngine,Microsoft Azure,Manjrasoft Aneka.
IaaS	Consumidores tem acesso a hardware ou armazenamento virtualizado. No topo do qual podem construir sua infraestrutura.	Máquinas Virtuais, gerenciamento de infraestrutura e de armazenamento.	Amazon EC2 e S3, GoGrid, Nirvanix.

A Cloud Security Alliance por sua vez, classifica a nuvem em um modelo de sete camadas, sendo elas, facilidades, rede, hardware, sistema operacional, *middleware*, aplicação e usuário. Esta classificação visa dar maior granularidade às camadas e auxiliar no monitoramento e definição de metricas de SLA e SLOs. Neste contexto, (SPRING, 2011) propõe um mapeamento entre as camadas e os modelos tradicionais de serviço (IaaS, PaaS, e SaaS) representado na Tabela 1 com o foco na delimitação da responsabilidade e esfera de controle de cada camada, entre o provedor e seus consumidores

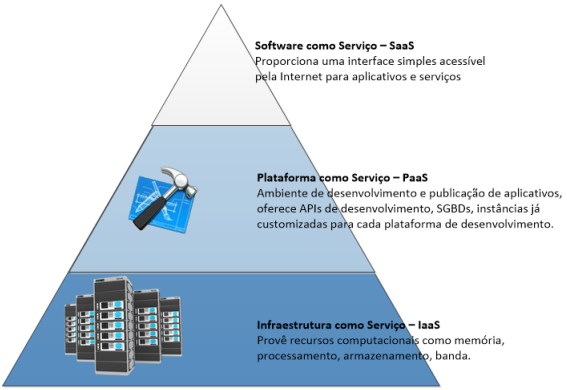

Figura 2 – Níveis de oferta da computação em nuvem (SCHUBERT, 2011)

.

Tabela 2 – Mapeamento e esfera de controle das camadas de nuvem (SPRING, 2011)

.

Camada	**IaaS**	**PaaS**	**SaaS**
Facilidades	Provedor	Provedor	Provedor
Rede	Provedor	Provedor	Provedor
Hardware	Provedor	Provedor	Provedor
S.O.	Provedor	Provedor	Provedor/Consumidor
Middleware	Provedor	Provedor/Consumidor	Consumidor
Aplicação	Provedor	Consumidor	Consumidor
Usuário	Consumidor	Consumidor	Consumidor

2.1.3 Modelos de Implantação

De acordo com os modelos de implantação, (MELL; GRANCE, 2011) divide as nuvens em:

- Nuvem privada: a infraestrutura da Nuvem é gerida apenas por uma organização.
- Nuvem comunitária: a infraestrutura da Nuvem é compartilhada por várias organizações e suporta uma comunidade específica que possui interesses em comum.

- Nuvem pública: a infraestrutura da nuvem está aberta e disponível para o público em geral ou um grande grupo industrial e é propriedade de uma organização responsável por sua comercialização.

- Nuvem híbrida: a infraestrutura da Nuvem é composta por duas ou mais Nuvens (privadas, públicas, comunitárias) que permanecem entidades separadas porém são interligadas por tecnologia padronizada ou proprietária possibilitando portabilidade de aplicações e dados.

2.1.4 Plataformas Não Comerciais de Código Aberto

Com a consolidação de conceitos e características da computação em Nuvem, plataformas aderentes a estes novos conceitos foram sendo desenvolvidas, entre plataformas comerciais de código fechado (por exemplo, Vmware VCloud) e plataformas de código aberto.

O foco da arquitetura proposta está na utilização de plataformas e arcabouços de código aberto, devido à facilidade de adaptação e compartilhamento de conhecimento. De acordo com (LOCKHEED, 2010), plataformas de código aberto tem impactado a adoção da computação em nuvem em agências governamentais (do governo dos Estados Unidos da América). Especificamente, a ausência de padrões representa atualmente um dos obstáculos para a adoção da computação em nuvem. Neste sentido soluções de código aberto são excelentes formas de facilitar a adoção de padrões, pois os desenvolvedores podem resolver problemas de compatibilidade ou falhas de forma mais rápida para suportar os padrões, além de possuir uma vasta gama de desenvolvedores trabalhando de forma colaborativa. Apenas as plataformas mais relevantes e ativas serão apresentadas.

2.1.4.1 OpenStack

O Openstack pode ser considerado uma coleção de projetos de software de código aberto que possibilitam a construção de uma Nuvem completa ou de estruturas separadas de armazenamento, computação, rede entre outros recursos para empresas ou provedores. Inicialmente patrocinado pela NASA e Rackspace, o projeto se expandiu rapidamente ganhando apoio de grandes empresas e agências como IBM, Red Hat, HP entre outras.

Considerado um sistema operacional em nuvem que controla grandes conjuntos de recursos computacionais, de armazenamento e rede através de centros de processamento, gerenciados através de uma interface central que habilita administradores e usuários a provisionar recursos através de uma interface web (OPENSTACK, 2013). A Figura 3 apresenta a arquitetura do OpenStack com os principais projetos.

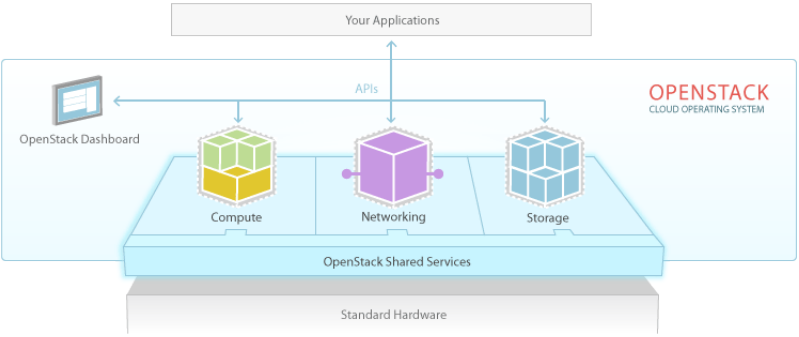

Figura 3 – Arquitetura principal do OpenStack (OPENSTACK, 2013).

Sua característica principal é a modularidade, fazendo com que apenas parte da solução seja implementada, de acordo com as necessidades da organização. Esta modularidade também gera uma certa complexidade na implementação de toda a suíte OpenStack.

2.1.4.2 OpenNebula

O projeto OpenNebula foi um dos pioneiros no desenvolvimento de plataformas para gerenciamento e orquestração de computação em nuvem. Seu início remonta a um projeto de pesquisa iniciado em 2005 por Ignacio M. Llorente e Rubén S. Montero, vindo a se concretizar como software em 2008, evoluindo como um projeto de código aberto cujas contribuições principais emergem do ambiente acadêmico.

A plataforma proporciona uma solução simples e flexível, porém rica em funcionalidades para o gerenciamento de centros de processamento virtualizados na construção de nuvens de infraestrutura como serviço. Apresenta uma interface web para instanciação e gerenciamento de recursos e uma arquitetura monolítica de configuração.

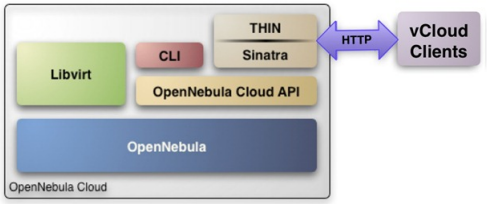

Figura 4 – Arquitetura principal do OpenNebula (OPENNEBULA, 2013).

2.1.4.3 Eucalyptus

A plataforma Eucalyptus é considerada um modelo de código aberto que utiliza infraestruturas computacionais e de armazenamento comumente disponíveis em organizações para prover uma plataforma modular e aberta à experimentação e estudo (NURMI et al., 2008).

No design da plataforma três componentes merecem destaque:

- **Gerenciador de instâncias**: controla a execução, inspeção e terminação de máquinas virtuais no hospedeiro onde são executadas;
- **Gerenciador de grupo**: coleta informações sobre as instâncias e controla a execução de uma instância em um hospedeiro específico, além de gerenciar a rede virtual;
- **Gerenciador de nuvem**: é o ponto de acesso à Nuvem para usuários e administradores. Coleta informações dos gerenciadores de cada nodo na rede, sobre utilização dos recursos e toma decisões de alto nível sobre o agendamento dos recursos, encaminhando as decisões aos gerenciadores de grupo.

2.1.4.4 CloudStack

O CloudStack é um arbacouço para gerenciamento de infraestrutura patrocinado pela Apache Software Foundation e desenvolvido em Java. Ao contrário do OpenStack, o Cloudstack e caracterizado por uma arquitetura monolítica, ou seja, não é disponibilizado em módulos integráveis (FOUNDATION, 2014).

2.2 COMPUTAÇÃO AUTÔNOMA

A computação autônoma foi inicialmente proposta como uma visão pela IBM no inicio dos anos 2000. Esta visão surgiu do paradoxo entre a crescente complexidade dos sistemas computacionais e a necessidade de administração e tomada de decisão sobre estes sistemas. Esta crescente complexidade tornou-se um obstáculo à limitada capacidade humana de processar informações, tomar decisões e administrar ambientes heterogêneos e integrados com requisitos estritos de tempo, performance e qualidade da informação apresentada.

Neste contexto de ambientes altamente heterogêneos e complexos, a computação autônoma foi apresentada, uma metodologia de construção de sistemas computacionais dotados de mecanismos de autogerenciamento, dirigido através de informações de alto nível e objetivos proporcionados por um operador humano.

O conceito de um sistema autônomo deriva da analogia ao sistema nervoso autônomo presente nos seres humanos que governa sem intervenção consciente o funcionamento dos órgãos, como por exemplo o batimento cardíaco e a temperatura corporal, liberando a área consciente do cérebro destas atividades fundamentais porém de baixo nível (KEPHART; CHESS, 2003).

Os sistemas autônomos diferem dos sistemas especialistas essencialmente na forma da tomada de ação, o que não é usual em sistemas especialistas (GUTIÉRREZ; BRANCH, 2011). Os sistemas autônomos possuem vários pontos em comum com sistemas especialista, porém em uma forma menos genérica, sendo aplicados no controle e gerenciamento de um amplo conjunto de sistemas computacionais, enquanto os sistemas especialistas são aplicados de forma mais generalista.

Para atender à propriedade essencial de auto-gerenciamento, quatro propriedades são fundamentais a um sistema autônomo: auto-cura, auto-proteção, auto-configuração e auto-otimização. Além das quatro *auto* propriedades, (DOBSON et al., 2010a) apresenta mais quatro atributos de um sistema autônomo: auto-conhecimento, auto-situação, auto-monitoramento e auto-ajuste, definidas em (HUEBSCHER; MCCANN, 2008) como:

- **Auto-configuração**: um sistema autônomo é capaz de auto-configurar-se de acordo com meta-objetivos, ou seja, ao especificar-se o que é desejado, não como o objetivo é alcançado. Isto pode representar ser capaz de auto instalar-se e adequar-se baseando-se nas necessidades da plataforma e do usuário;

- **Auto-otimização**: um sistema autônomo otimiza o seu uso a recursos. Ele deve ser capaz de decidir e iniciar mudanças proativas no sistema (ao contrário do comportamento reativo) na tentativa de melhorar a sua performance ou qualidade de serviço;

- **Auto-cura**: um sistema computacional autônomo detecta e diagnostica problemas. Os problemas encontrados podem ser interpretados amplamente, de baixo ou alto nível como correção de erros de memória ou dados incorretos em uma entrada de diretório. O sistema deve ser capaz de corrigir os erros encontrados, atuando de maneira reativa a falhas e não gerando novos erros.

- **Auto-proteção**: um sistema autônomo protege a si mesmo de ataques maliciosos e também de usuários finais que de forma inadvertida efetuam mudanças de software, por exemplo, ao deletar um arquivo importante. O sistema protege a si mesmo no que tange à segurança, privacidade e proteção de dados. Segurança é um aspecto importante da auto-proteção, não apenas em software, mas também em hardware. Um sistema também deve ser capaz de antecipar falhas de segurança e prevenir a sua ocorrência. Neste caso, o sistema autônomo age proativamente.

Nas palavras de (STERRITT; BUSTARD, 2003):

> O objetivo principal da computação autônoma é a criação de sistemas auto-gerenciáveis; estes são proativos, robustos, adaptáveis e fáceis de usar. Tais objetivos são conquistados através da auto-proteção, auto-configuração, auto-cura e auto-otimização. Para atingir estes objetivos um sistema deve ser auto-consciente e compreender o ambiente, o que significa ter conhecimento sobre o estado atual, tanto o seu próprio estado quanto o do seu ambiente operacional. Ele deve então auto-monitorar-se para reconhecer quaisquer mudanças no seu estado que necessitem de alguma alteração (auto-ajuste) para cumprir com o seu objetivo principal de auto-gerenciamento. Mais detalhadamente, isto significa que um sistema tendo conhecimento dos seus recursos disponíveis, componentes, suas métricas e limiares de performance característicos, seu estado atual, e o estado das interconexões com outros sistemas.

O modelo de referência proposto pela IBM em (KEPHART; CHESS, 2003) conhecido como o ciclo autônomo, ou MAPE-K apresenta uma

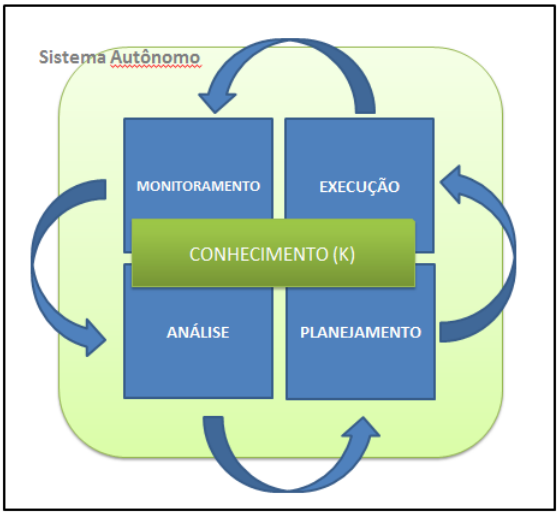

Figura 5 – Ciclo autônomo - MAPE-K (KEPHART; CHESS, 2003).

proposta para a arquitetura dos elementos autônomos. Esta arquitetura é detalhada em (HUEBSCHER; MCCANN, 2008) constituindo-se dos elementos Monitoramento, Analise, Planejamento, Execução e Conhecimento (*Knowledge*). A Figura 5 apresenta a estrutura de um sistema autônomo e as relações entre os elementos do ciclo. A primeira etapa do ciclo é o monitoramento, onde o elemento autônomo percebe o ambiente ao seu redor, após, os dados coletados são enviados para a etapa de análise, seguida pelo planejamento e execução.

Além dos elementos apresentados temos o gerenciador autônomo, que monitora o elemento gerenciado e executa ações através dos agentes executores. O gerenciador autônomo é geralmente configurado a partir de objetivos determinados por um administrador humano e atua na manutenção destes objetivos e configuração do elemento autônomo, melhorando ou corrigindo os elementos autônomos para atingir os objetivos de alto nível definidos, através de ações de baixo-nível em seus elementos gerenciados.

2.2.1 Computação em Nuvem Autônoma

O objetivo da computação autônoma e auto-gerenciar sistemas complexos e heterogêneos, onde a tomada de decisão e modificações são constantes e onerosas para operadores humanos. A computação em nuvem enquadra-se perfeitamente nesta descrição, devido ao seu dinamismo, complexidade e larga escala, sendo constituída por diversas camadas com diferentes níveis de complexidade abrangendo data centres com milhares de servidores e equipamentos em constante mudança e adaptação.

Em (BUYYA; CALHEIROS; LI, 2012) a computação em nuvem autônoma é apresentada como um importante desafio na obtenção de infraestruturas em Nuvem confiáveis, seguras e com um custo atrativo. Além de mencionar a importância da computação autônoma para as Nuvens, (BUYYA; CALHEIROS; LI, 2012) também elenca alguns pontos chave para a pesquisa:

- **Qualidade de serviço**: provedores de Nuvem necessitam garantir que a quantidade necessária de recursos da Nuvem será provisionada para que seus consumidores tenham os seus requisitos de qualidade de serviço mantidos, mantendo as limitações de orçamento e custo;

- **Eficiência energética**: inclui o uso eficiente de energia na infraestrutura, evitando utilização de mais recursos do que o necessário para as aplicações, minimizando o impacto das emissões de carbono pela Nuvem;

- **Segurança**: alcançar funções de segurança como confidencialidade, disponibilidade e confiabilidade contra ataques de negação de serviço . Os ataques de DoS são críticos pois em um ambiente de provisionamento dinâmico, o aumento no numero de usuários causa um aumento automático nos recursos utilizados pela aplicação. Em um ataque coordenado a um provedor de *SaaS*, um aumento inesperado nas requisições pode ser considerado um caso normal de ampliação na demanda, ampliando a oferta de recursos para atendê-la. Neste caso isto iria causar um aumento nos custos da aplicação que seria repassado ao cliente.

Porém na base de todos estes desafios esta o monitoramento eficiente da Nuvem, proporcionando uma base sólida para análise e tomada de decisões a partir de elementos autônomos.

2.3 COMPUTAÇÃO COGNITIVA

Apesar de ser considerada uma *buzzword* a computação cognitiva reúne conceitos interessantes como o encontro da inteligência artificial com a inteligência nos negócios (ARTIFICIAL..., 2014).

O termo computação cognitiva foi cunhado pela IBM como um conceito guarda-chuva que engloba diversas disciplinas da computação, partindo de iniciativas de arquiteturas cognitivas, em complemento à tradicional arquitetura de Von Neumann, na construção de componentes e chips, sistemas operacionais e na extração de informações relevantes em tempo real em grandes bases e conjuntos de dados, seja eles estruturados ou não e sumarizando o conhecimento através de sistemas especialistas. O objetivo final é a humanização da computação, tornando a informação armazenada e coletada em algo inteligente, onde sistemas possam aprender e não apenas apresentar ou agrupar dados, mas sugerir ações e ser o elemento atuante caso necessário.

Em (ARTIFICIAL..., 2014) o conceito de computação cognitiva e introduzido como:

> Sistemas computacionais cognitivos aprendem e interagem naturalmente com pessoas para estender o que humanos ou máquinas podem fazer por conta própria. Eles auxiliam especialistas na tomada de decisões ao penetrar a complexidade do Big Data.
>
> O crescimento do Big Data está acelerando pois cada vez mais a atividade do mundo e expressada digitalmente, aumentando em volume, velocidade e incerteza. A maioria dos dados gerados atualmente constituem-se em dados não estruturados como vídeo, imagens, símbolos e linguagem natural. Um novo modelo computacional e necessário para processar e extrair sentido disto.
>
> Os sistemas computacionais cognitivos não são baseados em programas que predeterminam cada resposta ou ação necessária para executar uma dada função ou conjunto de tarefas. Ao invés disso, eles são treinados utilizando inteligência artificial e aprendizado de máquina para sentir, prever, inferir e, de certa forma, pensar. ... Ao invés dos sistemas especialistas do passado que necessitavam regras fixas codificadas em um sistema definido por um especialista humano, computadores cognitivos podem processar linguagem natural e dados não estruturados e aprender por

> experiência, semelhante à forma que aprendemos. Apesar de possuírem grande domínio sobre o domínio, ao invés de substituir especialistas humanos, eles irão atuar como sistemas de suporte à decisão, ajudando na tomada de decisão em domínios como finanças, saúde e serviços a clientes.

Pode ser afirmado que a computação cognitiva e uma abordagem contemporânea à preocupação de tratarmos a incerteza e utilidade agregando novas técnicas como aprendizagem de máquina e processamento de linguagem natural, endereçando conceitos da teoria da decisão.

2.3.1 *Big Data* e Ferramental

Você não pode gerenciar o que você não monitora

A expressão acima é atribuída a ambos, W. Edward Deming e Peter Drucker e explica porque a recente explosão de dados digitais é tão importante. O avanço do Big Data irá transformar o mundo dos negócios e a tomada de decisões em todos os domínios onde for utilizado (MCAFEE; BRYNJOLFSSON et al., 2012).

O crescimento do volume de dados e infraestrutura para suportá-lo possibilitou a solução de novos problemas, envolvendo o armazenamento e processamento de quantidades de dados sem precedentes. Há um grande interesse em trazer novas ferramentas para solucionar problemas antes intratáveis, derivar produtos e algoritmos totalmente novos, tratar dados brutos e transformá-los em informações com sentido, produzindo novas ferramentas analíticas. Estas ferramentas não são novas, mas baseiam-se no *corpus* de conhecimento adquirido em décadas de pesquisa em aprendizado de máquina e evolução de armazenamento e processamento. Isto é em resumo a ciência de dados (MCCREARY; KELLY, 2013).

O termo *Big Data* vem sendo empregado para representar o volume cada vez maior de dados digitais gerados pela humanidade, que vêm crescendo de forma exponencial nos últimos anos. Segundo estimativas da IBM, este volume alcançará 20 *zettabytes* em 2020 (KELLY; HAMM, 2013).

Encontramos uma definição em (MANYIKA et al., 2011), onde *Big data* é definido como conjuntos de dados cujo tamanho excede a habilidade de armazenamento, coleta, análise e gerenciamento dos sistemas gerenciadores de banco de dados tradicionais e seu ferramental associado.

Um volume expressivo de dados heterogêneos, estruturados e

semi-estruturados tornou as ferramentas tradicionais de análise estatística e mineração de dados obsoletas ou limitadas para extrair informações em tempo hábil e da forma correta. Soluções analíticas que mineram dados estruturados e não estruturados são importantes ao auxiliar organizações a adquirir uma visão mais acurada das suas atividades, não apenas utilizando os seus dados privados, mas também a partir de dados públicos (ASSUNCAO et al., 2013).

As principais características do Big data são consideradas as propriedades V, usualmente variedade, velocidade e volume (ASSUNCAO et al., 2013), (IBM; ZIKOPOULOS; EATON, 2011), também incluindo uma quarta propriedade V, veracidade (KELLY; HAMM, 2013).

A variedade é a propriedade que representa a heterogeneidade dos dados, onde, fontes distintas necessitam ser consideradas ao analisar um domínio. No contexto do monitoramento para a computação em nuvem, esta heterogeneidade é apresentada pelas diferentes provas ou agentes que coletam ou geram dados, como sistemas de refrigeração, dados de tensão e corrente elétrica, indicadores de falha em hardware, agentes de software, provas genéricas e dados SNMP coletados, bem como dados de vídeo e áudio de câmeras de monitoramento, portas eletrônicas, controle biométrico e todos os controles e sensores que atuam em um ambiente de centro de processamento de dados. (KELLY; HAMM, 2013) defende que é necessária uma visão holística ao gerenciamento de dados e análise. Pois devem ser combinados diferentes tipos de dados e atuar sobre esses dados com conjuntos diferentes de ferramentas.

O manuseio e análise destes dados impõe diversos desafios pois, os dados podem ser de diferentes tipos, entre os tipos considerados no Big data temos os dados estruturados (bancos de dados tradicionais no modelo relacional), não estruturados, semi-estruturados e mistos. É argumentado que grande parte dos dados gerados atualmente pertencem ao grupo dos dados semi ou nao estruturados (ASSUNCAO et al., 2013).

O volume de dados é uma das características mais determinísticas do Big data. A definição de quão grande um conjunto de dados deve ser depende do contexto e da necessidade, podendo ser representado não apenas por uma unidade de quantidade em *bytes*, mas por exemplo pelo número ou volume de transações, arquivos ou até mesmo tempo, como por exemplo na necessidade das empresas estadunidenses em manter os seus registros durante um período de sete anos (RUSSOM, 2011). Entretanto, o volume de dados apresenta um dos principais desafios do Big data, pois, enquanto o custo por armazenamento decai consistentemente, a quantidade de dados coletada aumenta a uma taxa muito

maior, fazendo necessário que novas técnicas de armazenamento e movimentação de dados seja necessária.

A movimentação de dados refere-se especialmente à arquitetura de Von Neumann, onde os dados encontram-se em unidades separadas fisicamente do local onde são processados e existe um custo de tempo e energia para efetuar esta movimentação, dependendo ainda da localidade do dado. Existem estudos em andamento para criar sistemas inteligentes que consigam organizar os dados de forma dinâmica, visando minimizar as movimentações de dados e o custo de tempo nestas operações (KELLY; HAMM, 2013).

Além do volume e da variedade de dados, a terceira propriedade trata da velocidade em que estes dados necessitam ser processados e analisados para extrair valor e informações para a tomada de decisão. A velocidade depende do contexto dos dados, forma de chegada e necessidade de tempo de processamento. Por exemplo, algumas aplicações trabalham com *batch jobs*, outras necessitam de análise em tempo real e tomada de decisão com base nos dados atuais e histórico de desempenho.

A taxonomia apresentada por (KELLY; HAMM, 2013) adiciona uma quarta propriedade, a veracidade, que aborda a questão da confiabilidade das fontes dos dados.

2.3.2 Aprendizado de Máquina e Sistemas Especialistas

A visão da computação autônoma e, mais recentemente cognitiva e de modelos como Big Data devem muito ao uso de aprendizado de máquina e sistemas especialistas.

A inteligência, ou seja, a busca por sistemas que "pensem", ou de alguma forma simulem o comportamento humano faz parte da história da computação. Desde o artigo de Alan Turing na revista Mind, em 1950 que a pergunta "As máquinas podem pensar?"tem sido feita (TURING, 1950).

No contexto da computação em nuvem sistemas inteligentes tem sido propostos como soluções eficientes para alocação de recursos (BODIK et al., 2009), (SCHUBERT, 2011), automação de data centers (ARMBRUST et al., 2009), e na predição de violações de SLA e monitoramento (SCHUBERT; MENDES; WESTPHALL, 2013).

Devido à extensibilidade e profundidade do tema, para o contexto da dissertação apenas uma avaliação entre redes neurais e redes bayesianas será abordada, com o objetivo de verificar a aderência destes

modelos para o propósito da dissertação.

Conforme (ZHANG; BIVENS, 2007), o uso de aprendizado de máquina tem crescido para o treinamento de modelos a partir de dados de performance coletados por instrumentação/monitoramento. Estas abordagens de aprendizado estatístico não consideram envolvimento humano e necessitam pequena análise de domínio. Apesar de promissoras, estas abordagens ainda estão em seus estágios iniciais, e geralmente executadas aleatoriamente.

A avaliação entre redes neurais e redes bayesianas apresentadas por (ZHANG; BIVENS, 2007) faz uma avaliação de aderência e representatividade destas duas técnicas de aprendizado de máquina em relação a uma série de características, tendo como base acurácia e velocidade de execução. As tabelas 3 e 4 apresentam os resultados obtidos em relação aos testes efetuados. O escopo da avaliação está na aderência dos modelos a aplicações e ambientes em escala Web, ou seja, a possibilidade de representar conhecimento e a predição de variáveis em ambientes com conjuntos extensos de dados.

Tabela 3 – Características de Performance (ZHANG; BIVENS, 2007)

Características	**Redes Neurais**	**Redes Bayesianas**
Acurácia	SIM	NÃO
Velocidade de Validação do Modelo	NÃO	SIM

A Tabela 3 demonstra que as redes neurais possuem maior acurácia na representação dos modelos, porém perdem em velocidade de validação. Já a Tabela 4 apresenta demais parâmetros não relacionados diretamente com a performance, onde, aspectos fundamentais como a incorporação de conhecimento, interpretação do modelo, poder de representação e formas de validação. Destes aspectos o mais importante para o trabalho é a capacidade de incorporação do conhecimento, onde ambas abordagens mostraram-se satisfatórias.

2.4 BANCOS DE DADOS NÃO RELACIONAIS E NOSQL

Os bancos de dados não relacionais e mais recentemente o movimento NoSQL vieram para suprir necessidades específicas em situações onde a garantia das propriedades ACID (Atomicidade, Consistência,

Tabela 4 – Características não relacionadas a Performance (ZHANG; BIVENS, 2007)

Características	**Redes Neurais**	**Redes Bayesianas**
Incorporação do conhecimento do domínio	SIM	SIM
Interpretação do Modelo	SIM	NÃO
Poder de representação	NÃO	SIM
Diferentes formas de validação do modelo	SIM	NÃO

Isolamento e Durabilidade), característica principal dos bancos de dados relacionais, não e estritamente necessária.

Essencialmente, os bancos de dados NoSQL destinam-se ao rápido e eficiente processamento de grandes conjuntos de dados, com foco em performance, confiabilidade e agilidade (MCCREARY; KELLY, 2013). A Tabela 5 apresenta os diferentes tipos de bancos de dados NoSQL.

Tabela 5 – Tipos e armazenamento de dados NoSQL - as quatro categorias principais de sistemas NoSQL com produtos exemplo (MCCREARY; KELLY, 2013)

Tipo	**Uso Típico**	**Exemplos**
Incorporação do conhecimento do domínio	SIM	SIM
Interpretação do Modelo	SIM	NÃO
Poder de representação	NÃO	SIM
Diferentes formas de validação do modelo	SIM	NÃO

Uma das características principais que dos bancos de dados relacionais é a garantia das propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade), em geral, essas propriedades representam uma regra de tudo ou nada, ou seja, ou todas são cumpridas ou toda a transação é desfeita.

Além das propriedades ACID, os bancos de dados relacionais tradicionalmente sofrem do problema da escalabilidade: grandes conjuntos

de dados tornam-se complexos e custosos de serem processados, dadas as estritas e inflexíveis propriedades ACID e componentes arquiteturais que impedem uma escalabilidade horizontal, ou seja, a distribuição do processamento em diversos nós.

Historicamente, os bancos de dados relacionais tem desempenhado um papel essencial, especialmente em organizações que necessitam de total confiabilidade como instituições bancárias e setores empresariais públicos e privados. Porém, com o avanço da nuvem e o crescimento exponencial do volume de dados gerado, bem como a necessidade de armazenar e processar esses dados tem feito modelos alternativos surgirem.

2.5 CONCLUSÃO

Este capítulo teve como objetivo principal introduzir os principais conceitos teóricos, bem como fundamentar a proposta, apresentando as bases utilizadas como fonte para o trabalho.

Ao abordar os conceitos de computação em nuvem, buscou-se aproximar o leitor, de forma sintética com os conceitos mais importantes do tema. Ao abordar a computação autônoma e cognitiva, onde se enquadram conceitos importantes ao trabalho como o ciclo autônomo, as propriedades auto* e conceitos de aprendizado de máquina buscou-se abranger uma visão geral dos conceitos fundamentais da proposta.

Desta forma atendeu-se ao objetivo especifico descrito na seção 1.3 apresentando com rico referencial teórico os principais aspectos teóricos abordados pela dissertação, apresentando o estado da arte em pesquisas nas diferentes áreas abordadas pelo trabalho.

3 MONITORAMENTO PARA COMPUTAÇÃO EM NUVEM

O monitoramento é uma tarefa de importância fundamental para qualquer sistema computacional. A sua importância para a Nuvem é primaz para a própria conceituação de um ambiente como aderente aos conceitos de computação em nuvem abordados no Capítulo 2.

Neste contexto vários trabalhos endereçaram o tema do monitoramento em nuvem, em (SPRING, 2011) são definidas algumas características e elementos indispensáveis no monitoramento de Nuvens, já em (CHAVES; URIARTE; WESTPHALL, 2010) e (CHAVES; URIARTE; WESTPHALL, 2011) um arcabouço de monitoramento para nuvens privadas é apresentado utilizando ferramentas estáveis de monitoramento como o Nagios[1] e estendendo-o para proporcionar a necessária integração e autonomia para a Nuvem

Apesar de vários trabalhos abordando o tema de monitoramento, havia uma lacuna conceitual teórica importante. A definição de uma taxonomia de monitoramento é um elemento-chave na computação em nuvem. Apesar da sua importância central o monitoramento vem recebendo pouca atenção da comunidade científica. Esta taxonomia é encontrada em (ACETO et al., 2013a) que buscou avaliar propriedades, características e desafios para o completo monitoramento de ambientes heterogêneos e rapidamente variáveis como a CN.

O monitoramento da nuvem possui uma elevada relevância tanto para provedores de serviços em Nuvem quanto para consumidores de serviços de Nuvem. Para o primeiro é uma ferramenta chave de controle e gerenciamento de recursos computacionais, tanto hardware quanto software, para o consumidor proporciona parâmetros de performance tanto para a plataforma quanto aplicações. A monitoração contínua da Nuvem e de seus SLAs provê informações aos provedores e consumidores sobre, por exemplo, a carga gerada nos sistemas ou indicadores de QoS (ACETO et al., 2013b).

Muitas das atividades essenciais de uma Nuvem dependem de dados e de uma plataforma de monitoramento, que necessita ser aderente às características essenciais da Nuvem, como a possibilidade de autoatendimento por parte de consumidores, escalabilidade e elasticidade dos recursos, serviço mensurado e bilhetagem baseado no consumo.

A computação autônoma, definida brevemente no capítulo anterior apresenta elementos que podem ser adaptados e adotados pela

[1] www.nagios.org

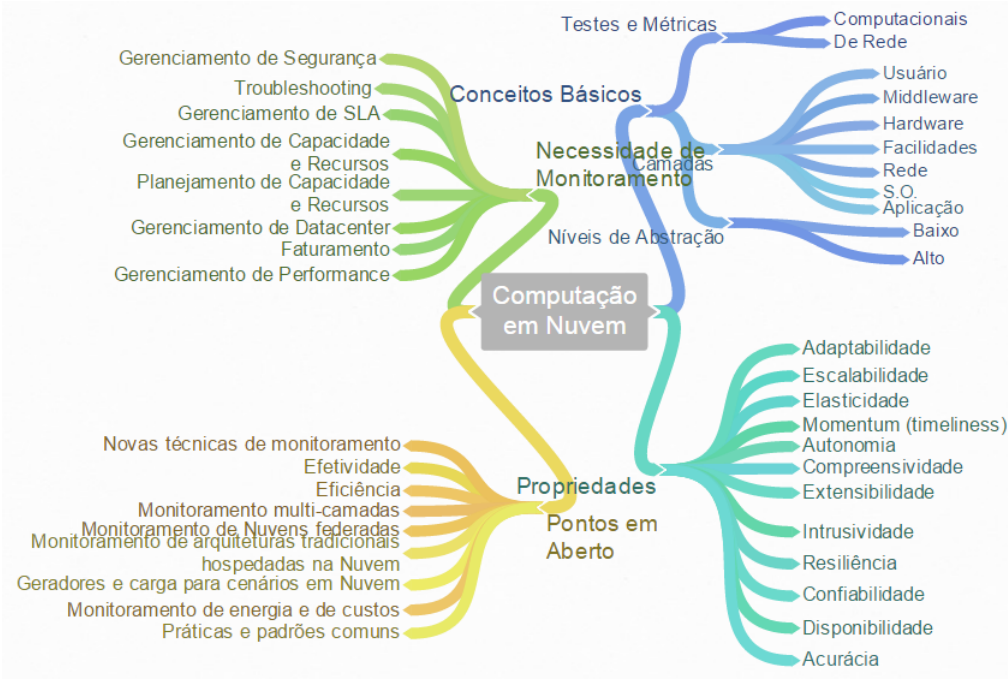

Figura 6 – Taxonomia para o monitoramento de computação em nuvem (ACETO et al., 2013b).

computação em nuvem, como o auto-monitoramento, tornando a computação em nuvem um campo de interesse para a implementação de sistemas autônomos (BUYYA; CALHEIROS; LI, 2012).

Este capítulo visa abordar conceitos inerentes ao monitoramento de Nuvens, proposto por (ACETO et al., 2013b). As principais características, propriedades, bem como, a necessidade do monitoramento e linhas de pesquisa, apresentados na Figura 6, serão abordados, devido à sua relevância para a definição do problema e a proposta do presente trabalho.

3.1 CONCEITOS DO MONITORAMENTO DE COMPUTAÇÃO EM NUVEM

O monitoramento da Nuvem é necessário para mensurar continuamente e determinar comportamentos de uma infraestrutura ou aplicação em termos de performance, confiabilidade, consumo de energia, habilidade de cumprir SLAs, segurança, entre outros elementos (KUTARE et al., 2010). A seguir serão brevemente abordados os conceitos de camadas da Nuvem, níveis de abstração e métricas adotados por (ACETO et al., 2013b).

O modelo em camadas adotado por (ACETO et al., 2013b) é o trabalho da Cloud Security Alliance, onde a Nuvem pode ser modelada em sete camadas: facilidades, rede, hardware, sistema operacional, *middleware*, aplicação e usuário.

- Facilidades: estrutura física do provedor da Nuvem, abrange os centros de dados, abastecimento de energia, refrigeração, segurança física e demais elementos que compõem o cenário físico de um provedor.
- Rede: considera os caminhos de dados na Nuvem e entre a Nuvem e seus usuários.
- Hardware: equipamentos de processamento e rede.
- Sistema operacional: consideram-se aqui os sistemas operacionais hospedeiro e sistemas operacionais em instâncias virtuais.
- *Middleware*: é a camada entre o sistema operacional (hospedeiro ou virtual) e a aplicação de usuário. Geralmente presente em provedores de plataforma e software como serviço.
- Aplicação: é a aplicação executada pelo usuário da Nuvem.
- Usuário: usuário final que acessa a Nuvem.

No contexto da computação em Nuvem, estas camadas constituem-se nos alvos para os agentes de monitoramento. Além das camadas de monitoramento, temos as métricas, que se dividem em métricas de sistema e métricas de cliente, no contexto de métricas internas e métricas externas a um sistema em Nuvem.

Ainda sobre métricas, tem-se as clássicas métricas baseadas em processamento e as métricas baseadas em entrada e saída (I/O bound) ou mais especificamente rede.

3.2 PROPRIEDADES DE MONITORAMENTO EM NUVEM

Para o correto monitoramento em um sistema complexo e heterogêneo como a Nuvem, um sistema de monitoramento distribuído e que contemple agentes e provas em várias camadas e métricas necessita atender a um conjunto de propriedades.

- *Escalabilidade*: um sistema de monitoramento é escalável se ele pode comportar um grande número de agentes e provas. Tal propriedade e muito importante na computação em nuvem devido ao elevado número de parâmetros e o grande conjunto de recursos a ser monitorado. Esta importância é ampliada com a adoção da virtualização, onde ao invés de apenas uma instância física tem-se varias instâncias virtuais com características e aplicações distintas (ACETO et al., 2013b).

- *Elasticidade*: um sistema de monitoramento é elástico se ele pode se adaptar a mudanças dinâmicas nas entidades monitoradas, de forma que recursos criados e destruídos sejam corretamente mensurados, adicionados e removidos do sistema de monitoramento.

- *Adaptabilidade*: a capacidade de um sistema de monitoramento se adaptar a variações nos recursos e na carga de rede e computacional, de forma a não ser invasivo, ou seja, afetar o funcionamento da Nuvem, degradar performance de aplicações de usuários ou alocar recursos em excesso, reduzindo a função de lucratividade do provedor.

- *Pontualidade (timeliness)*: um sistema de monitoramento é pontual (o termo *timeliness* também pode ser traduzido como conformidade ou exatidão) se os eventos detectados estão disponíveis a tempo do seu uso proposto.

- *Autonomia*: um sistema de monitoramento autônomo é capaz de auto-gerenciar seus recursos distribuídos reagindo automaticamente a mudanças imprevistas, enquanto abstrai de consumidores e provedores a complexidade intrínseca.

- *Compreensividade*: um sistema de monitoramento compreensivo é aquele que consegue suportar diferentes tipos de recursos, distintos tipos de dados de monitoramento e vários fornecedores.

- *Extensibilidade*: um sistema é extensível se o seu suporte pode ser estendido facilmente.

- *Intrusividade*: um sistema de monitoramento é intrusivo se a sua implementação exigir considerável modificação na Nuvem.

- *Resiliência*: um sistema de monitoramento é resiliente quando a persistência do serviço entregue é confiável em face a mudanças.

- *Confiabilidade*: é a capacidade de executar uma dada tarefa sob condições específicas durante um dado período de tempo.

- *Disponibilidade*: um sistema de monitoramento é disponível se ele provê os serviços de acordo com a sua especificação quando requisitado.

- *Acurácia*: um sistema é considerado acurado quando as medidas que ele provê são acuradas, ou seja, elas representam o mais próximo possível da realidade.

3.3 PLATAFORMAS DE MONITORAMENTO

Além das propriedades, (ACETO et al., 2013c) também avalia as principais plataformas de monitoramento para computação em nuvem, tanto comerciais quanto de código-aberto.

O objetivo da avaliação das ferramentas foi a verificação da aderência às características de um sistema de monitoramento para a Nuvem listadas na seção anterior. Diversas plataformas de monitoramento atualmente utilizadas também na computação em nuvem surgiram como ferramentas de monitoramento para redes tradicionais, não possuindo, por especificação características intrínsecas de sistemas em Nuvem, como elasticidade e escalabilidade, entre outras.

As figuras 7 e 8 apresentam os resultados da avaliação de sistemas de monitoramento para a Nuvem efetuado por (ACETO et al., 2013c). Podemos verificar claramente que tanto as plataformas comerciais quanto as de código aberto não apresentam aderência significativa ao modelo da Nuvem.

3.4 CONCLUSÃO

Este capítulo buscou apresentar o estado da arte no monitoramento de nuvens, apresentando alguns trabalhos referenciais na área e algumas abordagens ao monitoramento de nuvens.

Platform	Scalability	Elasticiy	Adaptabilitiy	Timeliness	Autonomicity	Comprehensiveness	Extensibility	Intrusiveness	Resilience	Reliability	Availability	Accuracy
CloudWatch [95]		✓		✓			✓					
AzureWatch [137]	✓		✓		✓		✓					
CloudKick [96]	✓		✓									
CloudStatus [48]				✓								
Nimsoft [97]	✓					✓						
Monitis [99]						✓						
LogicMonitor [100]	✓	✓				✓						
Aneka [101]	✓	✓										
GroundWork [129]						✓						

Figura 7 – Plataformas de código aberto para o monitoramento da Nuvem (ACETO et al., 2013b).

Ao avaliar as lacunas apresentadas pelos sistemas atuais de monitoramento e as características propostas por Aceto et al., verifica-se a baixa aderência das plataformas atuais às novas demandas proporcionadas pela computação em nuvem no que tange ao volume de dados de monitoramento, tomada de decisão, apresentação de relatórios e instantâneos (*snapshots*) do estado da Nuvem e demais características.

As características de um sistema de monitoramento serão avaliadas a seguir, de acordo com os problemas encontrados nas plataformas atuais.

- Escalabilidade: os sistemas atuais baseiam-se em mecanismos tradicionais de armazenamento de dados e recuperação da informação, o que limita o volume e o período histórico armazenado.
- Elasticidade: a remoção e adição de elementos em tempo real sem intervenção humana mantendo o histórico para controle e faturamento. Neste contexto a integração com as plataformas de nuvem é fundamental, para a detecção e remoção de instâncias.

Platform	Scalability	Elasticiy	Adaptabilitiy	Timeliness	Autonomicity	Comprehensiveness	Extensibility	Intrusiveness	Resilience	Reliability	Availability	Accuracy
Nagios [104]							✓					
OpenNebula [105]	✓		✓									
CloudStack ZenPack [109]				✓								
Nimbus [110]					✓							
PCMONS [111]							✓					
DARGOS [128]			✓				✓	✓				
Hyperic-HQ [138]	✓					✓						
Sensu [139]		✓					✓					

Figura 8 – Plataformas comerciais de monitoramento para a Nuvem.

- Adaptabilidade: para não ser invasivo o sistema deve possuir um auto-gerenciamento, ou seja, poder monitorar os recursos que está consumindo e evitar que eles sobrecarreguem a nuvem.
- Pontualidade (*timeliness*): a demora e latência da entrega dos dados de monitoramento podem ocasionar a falta de informações no momento necessário para formar o instantâneo da nuvem.
- Autonomia: existe a necessidade de transposição dos sistemas atuais monolíticos e sem inteligência agregada para sistemas autônomos que percebam a si mesmos e consigam inferir e se adaptar ao dinamismo da nuvem.
- Compreensividade: os sistemas atuais são geralmente implementados para suportar um conjunto definido de modelos e marcas, ou são tão genéricos que necessitam desenvolvimento adicional a um esforço elevado para se adaptar com novos dispositivos.
- Extensibilidade: a ausência de APIs que sejam simples e em for-

matos abertos como SOAP, REST ou WSDL fazem da extensibilidade em vários sistemas um problema elevado na adaptação para as particularidades da nuvem. A padronização e criação de tais APIs são fundamentais para facilitar a extensão mantendo uma camada de abstração.

- Intrusividade: o uso de agentes nas instâncias virtuais ou plataformas sob o controle do consumidor são elementos intrusivos; a necessidade de modificações na nuvem para se adequar à plataforma de monitoramento também constitui um problema sério na implantação de novas ferramentas de monitoramento.

Dessa forma conclui-se este capítulo apresentando os principais problemas e características necessárias na construção de arcabouços de monitoramento para a Computação em Nuvem, tendo como base a taxonomia proposta por Aceto e trabalhos relacionados.

4 PROPOSTA DE UMA ARQUITETURA DE COMPUTAÇÃO AUTÔNOMA E COGNITIVA PARA O MONITORAMENTO DE NUVENS

A crescente adoção da computação em nuvem como modelo de entrega de serviços e recursos é um fato consolidado. Desde seus primórdios, onde haviam severos questionamentos em relação à confiabilidade, segurança e disponibilidade da nuvem, dúvidas sobre sua adoção no meio acadêmico (FOSTER et al., 2008), questionamentos sobre ser apenas mais uma onda tecnológica e apenas mais um *buzzwor* do mercado, muito se passou e atualmente a computação em nuvem é a base para novos saltos que vem transformando a forma como os seres humanos consomem e se beneficiam da computação.

O próximo grande paradigma a ser superado é a efetiva automação e construção de sistemas heterogêneos auto-gerenciados. Conforme abordado no Capítulo 2, essa é a visão da computação autônoma, e, mais recentemente ampliada na visão da computação cognitiva (KEPHART; CHESS, 2003), (IBM; ZIKOPOULOS; EATON, 2011).

A realização destas visões passa por avanços na utilização de soluções por software para problemas complexos anteriormente resolvidos apenas por hardware, soluções estas facilmente integráveis e com interfaces simples para controle. Neste contexto é relevante salientarmos as redes definidas por software, as SDN, que nada mais são do que que a transferência dos planos de controle do dispositivo de rede para software, podendo ser controlado e configurado remotamente, de forma centralizada (ALAETTINOGLU, 2013). Outra contribuição, já abordada no Capítulo 2, que vem sendo chamada como Big Data, possibilita a distribuição do processamento sob grandes conjuntos de dados, utilizando hardware comum e arcabouços específicos como *Map Reduce* (DEAN; GHEMAWAT, 2008).

A presente proposta se insere neste contexto de crescente automação e construção de sistemas inteligentes, que constituem-se em uma exigência para os sistemas distribuídos de larga escala com restrições de tempo e necessidade de decisão e reação dinâmicas e instantâneas para manter um estado correto (DOBSON et al., 2010a).

A computação em nuvem, com suas diferentes camadas, modelos de serviço e toda a complexidade abstraída do usuário final é um sistema que será amplamente favorecido por uma arquitetura autônoma não apenas de monitoramento, mas sim que automatize e auto-otimize seus recursos (BUYYA; CALHEIROS; LI, 2012).

Tendo em vista a taxonomia de monitoramento apresentada por (ACETO et al., 2013b) e abordada no capítulo anterior, a proposta da dissertação visa preencher as lacunas na teoria apresentadas e propor uma arquitetura aderente às propriedades de monitoramento inerentes à Nuvem. Existe uma considerável quantidade de trabalhos desenvolvidos na definição de Nuvens auto gerenciadas, porém ainda existe um espaço a ser explorado no monitoramento de infraestruturas, visando predizer possíveis violações de SLA. A maioria dos sistemas disponíveis são baseados em grade ou arquiteturas baseadas em serviço, o que, conforme já apresentado não são compatíveis devido às diferenças do modelo de serviço (EMEAKAROHA et al., 2012).

Entretanto, o monitoramento é apenas uma das etapas do ciclo autônomo, abordado no Capítulo 2. Para a melhor compreensão e situação do problema abordado, a visão geral de uma arquitetura autônoma para gerenciamento de nuvens será abordada utilizando um estudo de caso.

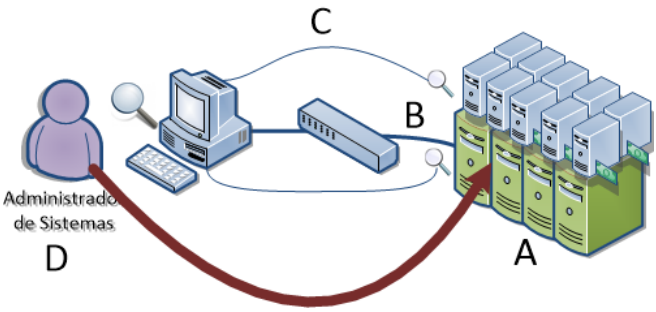

Figura 9 – Visão geral de uma nuvem privada.

A Figura 9 apresenta a visão geral de um ambiente tradicional de computação em nuvem:

A Representa a nuvem, neste contexto uma nuvem privada contendo elementos em todas as sete camadas apresentadas no Capítulo 2, sendo uma nuvem privada, virtualizada e gerenciada com o auxílio de uma plataforma de nuvem como o Cloudstack, Openstack, entre outros.

B Representam as provas e agentes de monitoramento instalados nas instâncias físicas e virtuais, que controlam e enviam informações a uma estação de monitoramento.

C Representam as informações de monitoramento enviadas à estação de monitoramento, consistindo em logs, dados SNMP, dados de monitoramento da plataforma. Geralmente estes dados são oriundos e agregados de diversas fontes, e apresentados por várias ferramentas distintas, por exemplo, Nagios[1] para monitorar as instâncias, Zenoss[2] para monitoramento da plataforma de nuvem, Cacti[3] para plotar gráficos de performance e ferramentas de analise de logs como Splunk[4], Logstash[5], Kibana[6] entre outros. Além disso podemos ter também informações de segurança, logs de IDS, IPS e de roteadores e gerenciadores de carga.

D Apresenta o principal agente de tomada de decisão e controle, o elemento humano. O administrador da nuvem deve permanecer atento às informações de monitoramento, pois devido à complexidade e tamanho dos ambientes elas são as principais ferramentas e muitas vezes as únicas para a tomada de decisão e controle do ambiente.

Em uma arquitetura autônoma, o elemento humano permanece presente, mas como o gestor definindo objetivos de alto nível para o ambiente, de acordo com seu papel. Estes objetivos podem ser representados por SLOs ou objetivos de maximização de lucros em detrimento de manutenção de SLAs se a perda de reputação e multas oriundas da violação forem menores que o prejuízo em executar uma ação autônoma para manter o estado da Nuvem.

A proposta de uma arquitetura autônoma visa remover a maior parte da intervenção humana do processo de gerenciamento da nuvem, e prover informação mais acurada sobre o seu estado. A figura 10 apresenta a visão geral de uma arquitetura autônoma para gerenciamento de nuvens.

Conforme pode ser visualizado na Figura 10 os elementos salientados representam:

A Representa a base de monitoramento, foco principal da proposta e sua inserção na arquitetura autônoma. Seguindo os princípios da computação autônoma, cada elemento do modelo MAPE-K deve ser um elemento autônomo completo auto-gerenciado. A

[1] http://www.nagios.org/
[2] http://www.zenoss.com/
[3] http://www.cacti.net/
[4] http://www.splunk.com
[5] http://logstash.net/
[6] http://www.elasticsearch.org/overview/kibana/

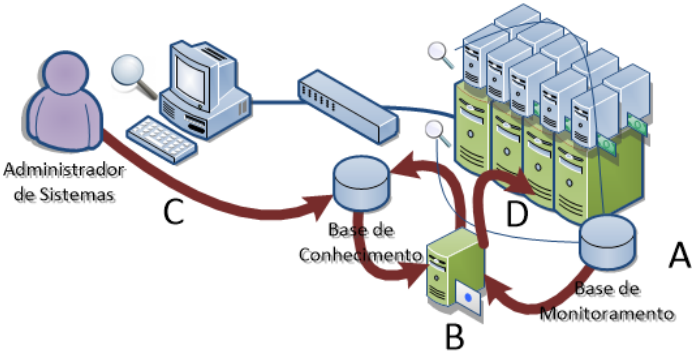

Figura 10 – Visao geral de uma nuvem privada autônoma.

base de monitoramento apresenta a diferença dos modelos tradicionais pois ao invés de cada ferramenta de monitoramento ter sua base, na arquitetura autônoma todos os dados de monitoramento pertinentes são armazenados em um único local, ou uma cópia é enviada ao sistema autônomo de monitoramento.

B Representa as etapas de processamento do ciclo MAPE-K, ou seja, a Análise, Planejamento e Execução. De forma geral, a Análise consiste na avaliação dos dados de monitoramento armazenados no item A, inicialmente na clusterização e classificação dos dados, e posteriormente análise com base em regras de negócio da base de conhecimento, validação de SLAs e construção do estado da nuvem. Após a análise, ocorre o planejamento, que recebe como saída os resultados da análise, por exemplo, violações de SLA encontradas e efetua o planejamento das ações conforme regras da base de conhecimento. Após a definição do planejamento, a etapa de execução efetivamente aplica as ações no sistema.

C Representam as regras de negócio e a base de conhecimento alimentadas pelo agente humano, estas regras podem ser os limiares de monitoramento, regras de SLA, objetivos da nuvem e critérios de alocação de recursos e desalocação. O sistema é autônomo, porém os objetivos e regras para sua operação são de responsabilidade do operador humano, que é quem define como e os parâmetros de operação da nuvem.

D Finalmente, o sistema autônomo executa ações sobre a nuvem,

finalizando o processo e repassando o *feedback* da execução para a base de conhecimento. Este *feedback* será avaliado na próxima iteração do sistema, ao comparar se o resultado esperado foi obtido ou se uma nova configuração é necessária. Essa análise sobre os resultados do sistema autônomo constituem a auto-gerência, também chamada de meta-gerência, que avalia a acurácia e a assertividade do sistema autônomo.

Apesar dos consideráveis avanços nos sistemas autônomos, buscando o auto-gerenciamento, a sua completa realização permanece inalcançada. Segundo (DOBSON et al., 2010b) os pesquisadores devem desenvolver uma nova abordagem de engenharia de sistemas para desenvolver sistemas efetivamente compreensivos.

Relembrando a arquitetura autônoma apresentada anteriormente, o ciclo autônomo MAPE-K está presente. O foco principal da presente dissertação é propor um modelo de monitoramento abrangendo a etapa de monitoramento, representada pela letra M no ciclo autônomo. A Figura 11 apresenta uma visão geral da arquitetura proposta.

Inicialmente o objetivo da dissertação era a construção de um ambiente autônomo completo abordando todas as etapas do ciclo autôno mo (MAPE-K). Porém a medida que a pesquisa foi avançando considerou-se que a tarefa, apesar de extremamente relevante seria inviável pelo curto espaço de tempo disponível. Mesmo a definição de um subconjunto, ou seja, a etapa de monitoramento apresenta desafios consideráveis que não serão abordados. O trabalho irá focar e propor um modelo de arquitetura para o monitoramento e entrará na etapa de análise ao avaliar métodos de predição e detecção de violações de SLA e SLOs.

As próximas seções irão aprofundar cada elemento da arquitetura, apresentando suas interfaces de comunicação, modelo de dados e detalhando seu funcionamento interno.

4.1 MONITORAMENTO COGNITIVO

Conforme o Capítulo 2, onde os conceitos de computação cognitiva são abordados, o foco da mesma é trazer a computação mais próximo do ser humano, através de conceitos da cognição humana construindo informação compreensível ao cérebro humano.

A computação cognitiva refere-se também a iniciativas de construção de sistemas inteligentes, a partir do armazenamento de grandes conjuntos de dados e a sua mineração com o auxílio de aprendizado de

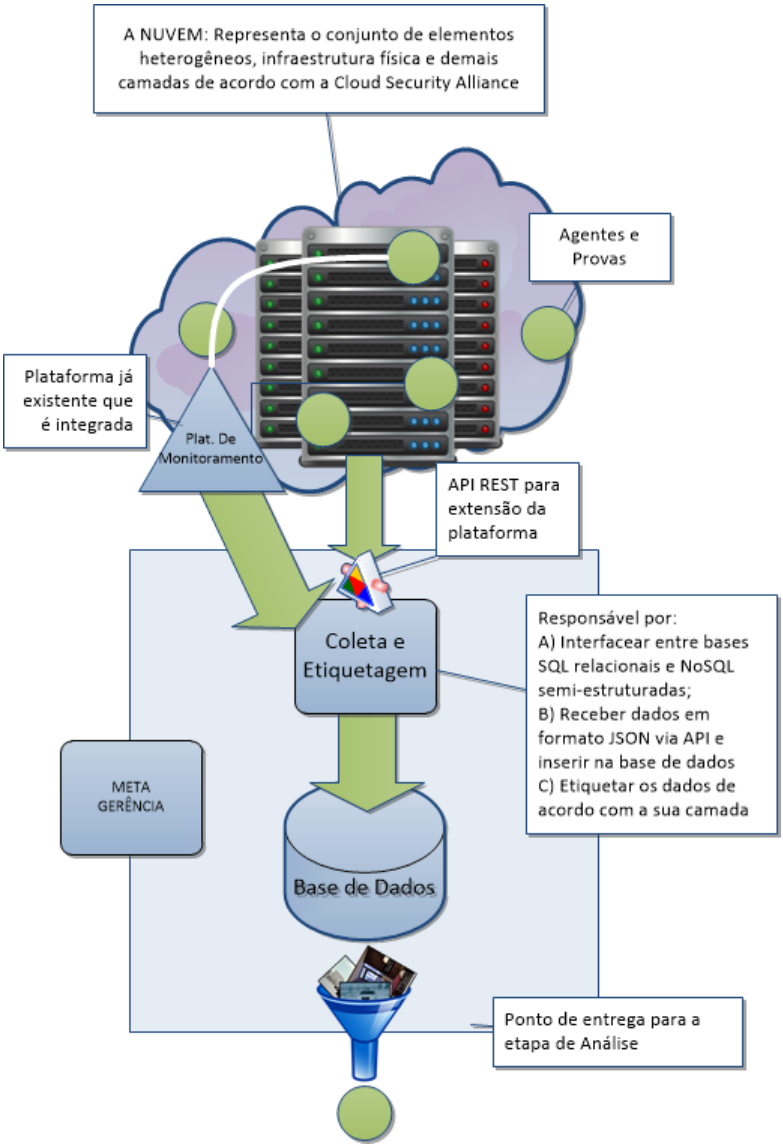

Figura 11 – Visao geral da arquitetura de monitoramento.

máquina e paralelismo.

Com base nestes conceitos o termo monitoramento cognitivo foi desenvolvido, com o objetivo de transmitir a ideia de um sistema que colete informações de diferentes fontes, de forma genérica e armazene os dados em um modelo flexível que possibilite a sua análise por arcabouços e algoritmos de aprendizado de máquina, agregação e clusterização gerando informações relevantes para os agentes e operadores da Nuvem.

Um aspecto importante a ser observado no monitoramento em CN é a temporalidade intrínseca dos dados coletados. Esta característica impõe algumas peculiaridades na estrutura de dados e no armazenamento das informações, da mesma forma que impõe desafios na recuperação da informação (VIEIRA et al., 2014).

4.1.1 Agentes, Provas, Sensores e Fluxos

De forma genérica, um agente ou sensor é um componente do sistema que faz a conexão entre o mundo exterior e o sistema de gerenciamento (VIEIRA et al., 2014).

No presente contexto, um agente ou prova é um elemento coletor de dados em um ponto ou camada específico da pilha da nuvem. Os conceitos entre provas e agentes podem divergir de acordo com a definição do autor, mas no presente trabalho representam coletores de dados.

A natureza desses agentes é diversa e vai além do escopo desta dissertação. O universo dos agentes coletores de sistemas em ambientes de rede ou distribuídos já têm sido explorado e encontra-se em um estado constituído.

Em linhas gerais um agente ou prova é um elemento coletor de dados, sem processamento ou transformação de dado intrínseco. No contexto do trabalho os agentes ou provas têm por função a coleta de informações úteis ao monitoramento da Nuvem, como exemplificado na Tabela 6.

Tabela 6 – Dados e métricas coletados por provas.

Camada da Nuvem	**Métricas (exemplos)**
Facilidades	Consumo de energia, temperatura, estado dos climatizadores.
Rede	Logs de comutadores e roteadores, banda passante nos links de dados de borda, latência até roteadores de borda
Hardware	Informações SNMP sobre o hardware, inclusive temperatura, velocidade de ventoinhas, estado dos componentes.
S.O.	Informações SNMP, WMI (Windows) sobre o sistema (CPU, memória, I/O, capacidade de disco,etc.), coleta de logs de sistema (arquivos de log em sistemas Linux, eventos em ambientes Windows).
Middleware	Estado do VMM, número de instâncias em execução.
Aplicação	Apenas sob autorização do usuário que define as métricas.

O objetivo é possibilitar a inclusão de uma extensa gama de agentes e provas ao modelo a partir de integração com outras plataformas e protocolos existentes.

4.1.2 Modelo de Dados

O modelo de armazenamento de dados de monitoramento constitui-se em um dos elementos centrais da proposta. Este modelo é necessário em virtude do problema do monitoramento da Nuvem, problema este abordado em (ACETO et al., 2013b), referindo-se especialmente às atuais soluções e plataformas existentes para a monitoração de Nuvem. Neste trabalho vemos também as propriedades características de um sistema em Nuvem, entre elas a escalabilidade, intrusividade e pontualidade que estão ligadas à forma com que os dados de monitoramento são armazenados.

O modelo proposto baseia-se em uma abordagem temporal-intervalar. Esta abordagem foi adotada devido à temporalidade presente nos dados de monitoramento, visto que métricas são coletadas em um espaço t de tempo, a ser definido de acordo com a granularidade da mensuração.

É intervalar pois para diferentes tempos sequenciais de leitura, a informação pode ser a mesma, sendo redundante armazenar dois registros cuja única variação é o tempo de leitura, tendo como requisito ser um intervalo completo.

Para tal avaliou-se inicialmente as estruturas de armazenamento das ferramentas de monitoramento de código aberto avaliadas por (ACETO et al., 2013c). A Tabela 7 apresenta as principais plataformas de monitoramento e seus respectivos motores de armazenamento. É predominante a utilização de bancos relacionais.

É com base nos dados coletados e armazenados na base de dados que as demais etapas da arquitetura autônoma irão se basear como fonte de dados para a tomada de decisões.

Segundo (ALMEIDA, 2014), cujo trabalho tem como base a presente proposta, o armazenamento e consulta de dados em séries temporais de forma eficiente é algo para o qual o modelo relacional padrão não é adequado. O modelo proposto não vai causar a perda de informação, pois é flexível o suficiente a ponto de armazenar a mesma informação que tabelas distintas armazenavam sem perda de dados, bem como a possibilidade de receber dados oriundos de arquivos de registros, ferramentas de monitoramento, dados SNMP entre outros. De acordo com (KAI et al., 2013), os valores de métricas precisam ser armazenados persistentemente para a análise do ciclo MAPE-K. Monitorar este sistema distribuído pode produzir uma grande quantidade de valores de métricas. Assim, o sistema de armazenamento deve ser escalável e flexível, com a capacidade de coletar milhares de métricas a partir de milhares de nós e aplicativos a uma alta frequência.

Segundo (ALMEIDA, 2014), uma vez que a maioria dos valores das métricas tem propriedades de séries temporais (vários valores por objeto (métrica) numa dada verificação no tempo), foi adotada a abordagem de séries temporais intervalar. Os dados de séries temporais tem características distintas. Estas propriedades podem ser exploradas por estruturas de dados customizadas para armazenamento e consultas mais eficientes. Sistemas relacionais não suportam nativamente esses tipos de formatos de armazenamento, de modo que essas estruturas são muitas vezes serializadas em uma representação binária e armazenados como uma matriz de *bytes* não indexados. Operadores personalizados são então obrigados a inspecionar esses dados binários. Os dados armazenados em um pacote como este é comumente chamado de *blob* (DIMIDUK et al., 2013).

Ademais, é pensado na utilização de conceitos de computação autônoma e cognitiva relacionados à arquitetura de monitoramento pro-

posta, durante a fase de Análise no ciclo MAPE-K, o que implica em tomar decisões importantes durantes as consultas de relações temporais. A atenção a isto é necessária para a alocação de recursos e atender requisições de usuários, como também, na otimização da utilização de recursos da nuvem. Por fim, alinhar-se com a aplicação das propriedades de auto-gerenciamento: autoconfiguração, auto-cura, auto-otimização e auto-proteção, garantindo assim a qualidade de serviço (QoS) e eficiência energética.

A Tabela 7 apresenta os bancos de dados suportados por cada plataforma.

Pode-se observar que a maioria das ferramentas atuais para monitoramento de nuvem tem o seu armazenamento de dados em um modelo relacional. O modelo e os bancos de dados relacionais têm seu fundamento nas propriedades clássicas ACID. Estas propriedades garantem a durabilidade das transações, consistência dos dados sob o risco da perda de performance. Embora estas propriedades sejam vitais para sistemas que necessitem de alta confiabilidade nas transações e consistência de dados, existem problemas onde a necessidade maior é por rapidez e performance de leitura e disponibilidade dos dados, sacrificando a consistência.

Problemas em que temos grandes volumes de dados semi ou não estruturados, variados e com necessidade de rápido processamento são candidatos a problemas *big data*. Neste sentido uma breve reflexão sobre os dados de monitoramento de CN faz-se necessária.

- Volume: os dados de monitoramento, sejam eles oriundos de SNMP, WMI, ou demais instrumentos de fornecimento de informações do sistema, como logs de sistema e aplicativos, servidores web constituem-se em fonte rica de informações para o monitor, mas dependendo do tamanho do ambiente podem gerar quantidades grandes de dados.

- Variedade: as diversas fontes de dados de monitoramento faz com que seus dados sejam oriundos de várias fontes como arquivos (logs), informações de bibliotecas SNMP, sensores externos, etc.

- Velocidade: um sistema de monitoramento para ser efetivo necessita proporcionar uma visão instantânea do ambiente, ou seja, a velocidade com que os dados são coletados, armazenados e analisados é crucial para uma experiência satisfatória de monitoramento.

Ao avaliar as propriedades acima e os dados de monitoramento,

Figura 12 – Visão geral do modelo temporal.

podemos identificar a aderência das informações monitoradas ao problema *big data*. A partir desta constatação buscou-se modelar um modelo de dados que representasse da melhor forma os dados de monitoramento e sua heterogeneidade.

O modelo proposto é temporal e intervalar, cujo formato de dado está apresentado na Figura 12.

Tabela 7 – Mecanismo de armazenamento de dados das ferramentas de monitoramento de código aberto.

Ferramenta de Monitoramento	**Bases de dados suportadas**	**Tipo**
Nagios	Interno, MySQL, PostgreSQL	Arquivo e Relacional
OpenNebula	SQLITE, MySQL	Relacional
CloudStack Zenoss	MySQL	Relacional
Nimbus	SQLITE	Relacional
PCMONS	Nagios	Relacional ou arquivos
DARGOS	MySQL	Relacional
Hyperic Open Source	MySQL, Oracle, PostgreSQL	Relacional
Sensu	Redis	NOSQL

A Tabela 7 apresenta o mecanismo de armazenamento de dados de diferentes plataformas de monitoramento de código aberto. Com exceção da plataforma Sensu, que utiliza um banco de dados não relacional, todas as outras plataformas se baseiam no armazenamento de dados em sistemas relacionais, que, conforme ja analisado anteriormente não possuem a escalabilidade, elasticidade e flexibilidade que os dados de monitoramento tem, especialmente no contexto crescente

de necessidade de coleta, armazenamento e análise de dados tendo em vista modelos cada vez mais autônomos de gerenciamento.

Vale salientar que o Sensu utiliza-se de um modelo NoSQL de chave valor[7], diferente do modelo apresentado pela presente proposta, que baseia-se no conceito de BigTable, ou seja, um modelo multi-colunar (*multi-row*).

Segue uma descrição dos elementos do modelo proposto e apresentado na Figura 12:

- Timestamp Inicial: Momento inicial do evento armazenado no formato UNIS Timestamp, ou seja, segundos desde 01 de janeiro de 1970, hora de Greenwich[8].
- Timestamp Final: Momento final de medição. Pode também representar um intervalo, caso a métrica já tenha sofrido agregação *a priori*.
- Origem: nome DNS do elemento monitorado, ou seja, o *hostname*.
- Data: Dados monitorados, em formato JSON.
- Tag[n...]: Sequência de colunas de Tag, com um formato chave-valor.

T_Inicial	T_Final	Origem	Data	Tag1	Tag2	Tag[n...]
2014/07/01 15:00:12	2014/07/01 15:01:14	prometheus.lrg.ufsc.br	{ cpu0_load: 0.64, cpu1_load: 0.24, cpu2_load: 0.87 }	Layer= virtualization	CPU	SLA=OK

Figura 13 – Exemplo de entrada de dados do modelo temporal.

A Figura 13 apresenta um exemplo de formato de dado recebido pelo sistema de monitoramento.

4.1.3 Coleta e Etiquetagem

De acordo com a visão geral apresentada anteriormente, a fase de coleta e etiquetagem de dados constitui-se na primeira fase do modelo.

[7] Veja mais em: http://redis.io/
[8] Veja mais em: http://www.unixtimestamp.com/index.php

Devido à característica de não intrusividade, optou-se por não considerar provas e agentes como elementos internos ao sistema, mas sim elementos externos porém acopláveis (*plugins*).

A Figura 14 apresenta a visão geral e os elementos que constituem a etapa de coleta e etiquetagem.

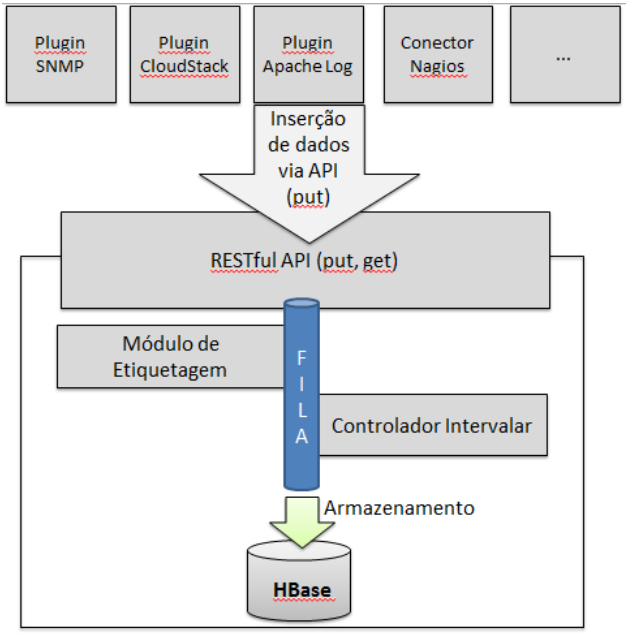

Figura 14 – Coleta e etiquetagem.

Como pode ser visto na Figura 14 o sistema de coleta recebe as métricas a partir de conectores, que podem ser de provas ou agentes diretamente (WMI, SNMP, Logs - syslog) ou conectores de plataformas como Nagios, CloudStack. Estes conectores necessitam ser desenvolvidos tendo em vista o modelo de dados apresentado.

Porém devido ao formato simplificado do modelo qualquer métrica pode ser facilmente convertida e adaptada. Devido ao escopo não entraremos em detalhes sobre a camada de conectores com outras plataformas e coletores, mesmo porque o tema já está consolidado na indústria e academia.

O uso da arquitetura REST deve-se à sua facilidade de implementação e fácil publicação de dados através de requisições HTTP, via

métodos PUT e POST.

A notação para inclusão de dados segue o formato JSON, e o formato básico de inserção de dados segue o formalismo abaixo:

$$\{metrica_A : valor, metrica_B : valor\}$$

Após o recebimento do dado no formato apresentado do modelo de dados, o dado é apresentado ao fluxo de dados da Figura 15. Após o recebimento do evento, existe uma validação JSON do campo DATA, do campo ORIGEM e do TIMESTAMP-INICIAL. Após a validação as tags básicas são geradas, sendo elas LAYER = camadas do monitoramento e METRICA, por exemplo CPU para métricas referentes à CPU (consumo, número de threads, ...). Após a geração de Tags é verificado se já houveram métricas inseridas na base com os mesmos valores para a mesma ORIGEM em um TIMESTAMP-INICIAL ou TIMESTAMP-FINAL anterior sem sobreposição de intervalo. Caso sim, se todos os campos forem idênticos, o valor é atualizado com o TIMESTAMP-INICIAL sendo o TIMESTAMP-FINAL atual do registro. Caso contrário, a entrada é considerada nova e inserida na base de dados.

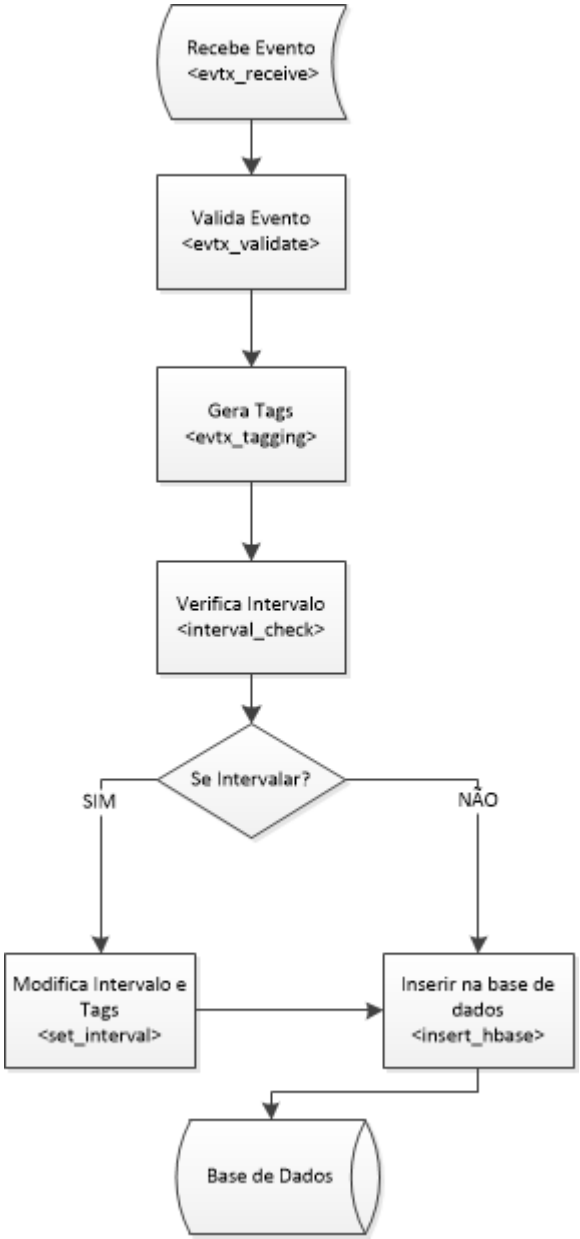

Figura 15 – Fluxo de dados.

4.2 ANÁLISE

A etapa de análise encontra-se inserida no ciclo autônomo, como a etapa posterior ao monitoramento.

Após a coleta dos dados do sistema e seu armazenamento, a análise verifica métricas relevantes aos diferentes papéis ou atores de uma nuvem e extrai conclusões do estado do sistema e avalia o estado de métricas relevantes a cada ator.

Existem um conjunto amplo de métodos analíticos que podem ser aplicados ao conjunto de dados armazenados no modelo proposto na fase de monitoramento, com o objetivo de agregar e correlacionar métricas para extrair informações relevantes ao contexto da CN. Dentre estes, três tipos de métodos analíticos foram selecionados para o monitoramento em CN (VIEIRA et al., 2014):

- Diagnóstico: métodos que visam sintetizar o fluxo temporal de eventos originado de sensores em um estado da Nuvem, extraindo uma visão geral da Nuvem (*dashboard*).
- Causa-raiz: o objetivo deste tipo de análise é determinar quais eventos são a causa principal do atual estado da Nuvem.
- Predição: métodos preditivos visam antecipar os estados futuros da nuvem a partir de análise probabilística de eventos com base no histórico, redes neurais ou métodos de decisão.

Além das considerações sobre os tipos de métodos analíticos, faz-se necessária uma reflexão das características que um método de análise deve possuir para o monitoramento de CN:

- devem existir métodos capazes de suprir um subconjunto de métricas representando parte da Nuvem. Por exemplo, possibilitar a construção de um estado da Nuvem limitado ao escopo de um elemento como uma instância virtual ou aplicação;
- a incerteza faz parte do comportamento de dados de monitoramento, portanto a acurácia de um método passa pela inclusão da incerteza em sua modelagem, como, por exemplo utilizando conjuntos difusos (*fuzzy*) ou métodos probabilísticos para representar a incerteza.
- a temporalidade deve ser considerada, geralmente com a utilização de séries temporais;

- deve ser multi-critério, ou seja, eventos aparentemente não correlacionados devem ser considerados ao analisar certo fenômeno;

- deve possibilitar a sua análise em tempo real, uma característica fundamental da CN onde dados são coletados em períodos de tempo pequenos e sequenciais, sendo a entrada para que decisões sejam rapidamente tomadas para atender certa necessidade;

- a análise deve prover métodos que aprendem do comportamento histórico, e possibilitem prever comportamentos futuros com base neste passado ou avaliar situações semelhantes. Métodos adaptativos que evoluem de acordo com a mutação do seu ambiente devem ser considerados.

No contexto do trabalho buscou-se a construção de um modelo representando o estado da Nuvem.

Segundo (MENDES; WESTPHALL, 2014), o estado do sistema define a circunstância em que o sistema se encontra. Isto pode ser feito a princípio pela medição de parâmetros de configuração ou variáveis de monitoramento. Apesar do estado ser proveniente de uma visão estática, ele em si não e estático, podendo se alterar durante o tempo ou com a adição de nova informação, uma vez que pode não se tratar de um estado simples.

Este estado constitui-se em um instantâneo atual, onde cada elemento $e \in E$, elementos estes podendo ser uma máquina física, um ativo de rede ou uma instância virtual é representado por todas as ocorrências relacionadas na base de monitoramento, construindo-se um vetor de ocorrências $Al_e = (al_1, t_1, al_2, t_2, \cdots, al_n, t_n)$, contendo todas as ocorrências al_x no tempo t_x. Este vetor de ocorrências é então correlacionado e a saída é exibida em um formato visualizável de forma intuitiva.

Na fase de análise, técnicas cognitivas de aprendizado de máquina e extração de conhecimento são utilizadas para extrair informações relevantes de causa-raiz e potenciais violações de serviço ou vulnerabilidades.

Neste contexto partiu-se para a avaliação de métodos probabilísticos e de aprendizado de máquina para avaliar e predizer violações de SLA, levando a determinação do estado do elemento analisado em relação a métrica analisada.

Um importante aspecto acerca das métricas e avaliações acerca do estado, é que elas devem estar associadas a algum tipo de visão ou atributo de interesse, como por exemplo, a verificação da violação ou

não de um SLO relacionado a um SLA, por exemplo, a disponibilidade de um sistema no atendimento à requisições. Buscou-se avaliar a viabilidade de utilização de redes neurais, redes bayesianas e métodos híbridos adicionando-se a incerteza a partir de conjuntos difusos, na detecção de violações de SLOs simples, como por exemplo, a disponibilidade de uma instância virtual e a carga desejável de uma aplicação.

Partindo de um escopo limitado, algumas métricas específicas foram analisadas e extraídas e seus resultados avaliados. Os resultados bem como especificação dos experimentos encontram-se no capítulo destinado à implementação.

4.3 PLANEJAMENTO

A fase de planejamento é a fase onde ocorre a tomada de decisões com base nas informações obtidas na fase de análise (estado do sistema, violação de SLAs, etc.) são analisadas com informações contidas na base de conhecimento que armazena objetivos e contratos alimentados pelo administrador da nuvem.

Devido à amplitude do tema, o foco aqui será apenas na dissertação de algumas possibilidades analisadas para esta etapa.

Conforme (BOUTILIER; DEAN; HANKS, 1999), o planejamento sob incerteza e um problema central de estudo na automação sequencial do processo de decisão, sendo abordado por vários pesquisadores em vários campos do conhecimento, incluindo planejamento em inteligência artificial (IA), análise de decisão, pesquisa operacional, teoria econômica e de controle.

Dentre esses arcabouços para a tomada de decisão, destaca-se o processo Markov, comumente chamado de *Markov Decision Process* (MDP).

De acordo com (VIEIRA et al., 2014), o funcionamento do MDP consiste, em:

- um conjunto de estados S do sistema, no presente contexto, o produto do diagnóstico realizado na fase de análise;
- um conjunto A de possíveis ações a ser tomadas no sistema;
- uma probabilidade da função de transição $P : S \times A \times S \rightarrow R$ que expressa a probabilidade do sistema no estado s, dada uma ação a, de ser conduzido ao estado s', neste ponto, a função de probabilidade sera o produto das predições providas pelos métodos de análise;

- uma função de recompensa $R : S \times A \times S \rightarrow R$ que valida a recompensa de tomar a ação a no estado s e conduzir o sistema ao estado s'.

Existem outras formas de descrever MDP, porém este é o modelo mais interessante para o nosso objetivo, do gerenciamento de monitoramento em Computação em Nuvem.

É importante observar que MDP é conhecido por não funcionar em sistemas com informação incompleta. Para seguir esta abordagem, certos requisitos devem ser considerados:

1. a etapa de monitoramento irá prover toda a informação necessária ao MDP através da utilização de um modelo de dados aderente e representativo e da utilização de técnicas de *Big Data* para extrair a informação relevante no momento necessário;

2. o conjunto dos estados possíveis é finito e tratável;

3. existe um número suficiente de métodos analíticos para suprir as necessidades de predição para atender à função de probabilidade;

4. existe um número suficiente de métodos analíticos para suprir as necessidades da validação de estado e suporte à função de recompensa.

Em conjunto com MDP pode-ser implementar e utilizar a teoria da utilidade esperada para definir qual ação deverá ser tomada, com base na decisão ou peso da utilidade de cada ação. A teoria da utilidade esperada define que o tomador de decisão escolhe entre possibilidades de risco ou incerteza a partir da comparação dos seus valores de utilidade esperada, isso significa, a soma ponderada obtida a partir da soma dos valores de utilidade esperada multiplicados pelas probabilidades do evento (MONGIN, 1997), (MEYER, 1987).

Além da utilidade esperada e MDP, outros modelos e teoremas podem ser aplicados no planejamento de ações a ser tomadas com base em modelos analíticos e preditivos. Este estudo vai além do propósito do trabalho, servindo como base para trabalhos futuros.

4.4 EXECUÇÃO

A execução é a última etapa do ciclo autônomo, e segue-se como passo imediatamente posterior ao planejamento. Seu propósito, como sua própria definição diz, é executar no ambiente as ações planejadas.

Um dos desafios inerentes à execução é o agendamento de execução das ações planejadas. A execução necessita manter um conjunto de ações cronologicamente enfileirados e garantir o gerenciamento de conflitos de ações. Problemas de escalonamento de recursos e execução são um aspecto de estudo a ser levado em conta em uma implementação completa da proposta.

Para exemplificar, um exemplo prático será abordado. Uma dada instância apresentou falha de rede, tornando-se inalcançável no seu domínio. O sistema de monitoramento detectou uma falha e armazenou na base de monitoramento a informação que a instância estava inalcançável. A etapa de análise, ao buscar o estado da instância localizou, no espaço temporal de análise que os últimos eventos relacionavam-se à indisponibilidade da instancia. O método de validação de SLAs, por sua vez, detectou que a falha gerou ou poderá gerar uma violação de SLA, visto que o limiar aceitável de tempo indisponível foi, ou poderá ser ultrapassado em breve. O planejamento, por sua vez, tendo por base a possível violação tomou por decisão remediar a situação, pois a utilidade de recuperar a instância é maior do que a multa ou dano gerado pela não recuperação, e, por fim, o método de execução realiza a reinicialização da instância defeituosa em outro *hypervisor*, sendo que na próxima análise dos dados de monitoramento, a instância será considerada novamente disponível, removendo este alarme da lista de problemas.

4.5 META-GERÊNCIA

A meta-gerência tem por objetivo o monitoramento e otimização do próprio sistema autônomo. De forma simplista é o monitoramento da plataforma de monitoramento. A meta-gerência abrange a autopercepção do sistema autônomo, ou seja, a consciência de seus meta-objetivos e de seus componentes.

No presente trabalho será adotada como modelo de auto, ou meta-gerência um modelo simplificado da arquitetura cognitiva H-Cogaff (SLOMAN, 2001). Este modelo é apresentado em (KENNEDY, 2008), como uma meta arquitetura para sistemas autônomos.

De forma simplificada, a arquitetura H-Cogaff esta representada na Figura 16. Ela apresenta um agente e sua estrutura em duas camadas contendo um nível de objeto e um nível meta. O nível de objeto O_1 exerce tarefas primárias no mundo exterior ao elemento, como coleta de dados do estado do sistema de monitoramento, no contexto do

trabalho.

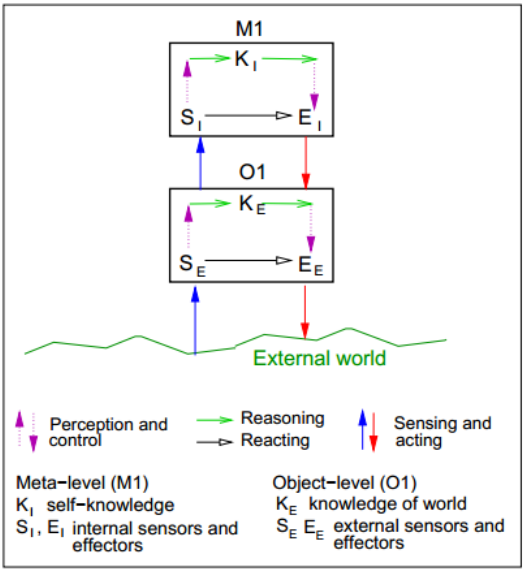

Figura 16 – Um agente cognitivo com meta-gerenciamento (KENNEDY, 2008).

K_E representa o conhecimento do mundo externo e inclui um modelo do seu funcionamento correto, podendo ser um modelo preditivo (envolvendo regras do funcionamento correto, ou meta-objetivos de operação, por exemplo, limites na utilização de recursos pelo sistema de monitoramento), possibilitando uma simulação interna ao avaliar cenários hipotéticos. O_1 inclui uma camada reativa, representada pela seta vermelha, capaz de efetuar ações imediatas no mundo exterior (a plataforma de monitoramento), sem ser necessário uma meta representação de alto-nível.

O nível de meta-gerenciamento representado na Figura 16 por M_1 monitora o processo cognitivo e as ações de O_1 e avalia a qualidade das ações e predições de acordo com um objetivo originado no mundo exterior (agente humano, papeis envolvidos na nuvem). O conhecimento do agente sobre si mesmo e representado pelo símbolo K_1, que contém um mapa dos componentes em nível de objeto, seus estados atuais e comportamento esperado. Um log das ações é guardado

e analisado em relação ao comportamento esperado e operação normal dos componentes para validar o processo de aprendizado e a qualidade de tal conteúdo.

A arquitetura H-Cogaff auxilia na construção de elementos cognitivos, na garantia das propriedades auto (definidas no Capítulo 2), especialmente a auto-percepção do sistema. Algumas propriedades definidas por (ACETO et al., 2013b) em sua autonomia, são relacionadas intrinsecamente com a meta-gerência, sendo que uma breve análise sobre a aplicabilidade das mesmas é abordada.

No contexto de um sistema autônomo de monitoramento as propriedades auto* estão ligadas as propriedades de intrusividade, adaptabilidade e elasticidade, onde um sistema deve ser capaz de:

1. perceber a sua existência (recursos alocados) e expandir-se ou reduzir seu tamanho para atender aos requisitos de monitoramento da Nuvem - elasticidade;

2. não ferir ou entrar em concorrência por recursos primariamente alocados a consumidores - intrusividade;

3. com base no monitoramento da plataforma (mundo exterior), em seus meta-objetivos e na qualidade das predições da fase de análise, ser capaz de se adaptar e redefinir métricas e instantâneos - adaptabilidade.

4.6 CONCLUSÃO

A presente proposta lança as bases, ao propor um modelo de monitoramento autônomo, para os problemas apresentados na Introdução, e expandidos no Capítulo 3, onde a aderência dos atuais arcabouços de monitoramento não se alinha satisfatoriamente às propriedades da CN e os desafios de apresentar métricas relevantes com rapidez sem perder informação.

A seguir a Tabela 8 apresenta a solução proposta para as características de um sistema de monitoramento para CN.

Tabela 8 – Conclusões a partir das propriedades do monitoramento em CN.

Propriedade	**Solução Apresentada**
Escalabilidade	A plataforma pode ser expandida e reduzida conforme a demanda de métricas e dados a serem processados graças a sua construção utilizando elementos como sistemas de arquivos distribuídos e processamento paralelo.
Elasticidade	Por sua arquitetura distribuída e paralela, o sistema pode ser expandido e reduzido conforme a demanda de dados e métricas a serem processadas e armazenadas.
Adaptabilidade	A adaptabilidade é uma característica que atua na reconfiguração de métricas, como por exemplo, redução da frequência das consultas a uma dada informação em um momento de alto uso do recurso. O sistema proposto é adaptável ao possibilitar o recebimento de métricas em diferentes intervalos de tempo (não exercendo controle ativo sobre os intervalos de monitoração).
Pontualidade	Pela sua construção baseando-se em elementos distribuídos e processamento paralelo, espera-se obter pontualidade nos resultados. Propriedade carece validação.
Autonomia	A sua modelagem em si é autônoma, sendo um elemento de uma arquitetura completa que busca implementar o ciclo autônomo, MAPE-K.
Compreensividade	O modelo é compreensivo ao suportar diferentes tipos e fontes de dados, armazenando-os em um formato semi-estruturado.
Extensibilidade	Através de um modelo simples de armazenamento e APIs REST pode ser facilmente estendido.
Intrusividade	O sistema não e intrusivo, pois baseia-se no presente estado na utilização de provas e agentes de terceiros, sendo facilmente acoplado a uma nuvem já existente.

Confiabilidade	A arquitetura nativa do HDFS e Hadoop proporcionam confiança de que os dados poderão ser consultados em virtude de falhas de nós. O sistema porém não e confiável em relação a faltas bizantinas, ou seja, considera os dados como sendo sempre verdadeiros.
Disponibilidade	Sua arquitetura Big Data utilizando Hadoop e HDFS é por projeto tolerante à falhas e de alta disponibilidade.
Acurácia	Não aplicável.

A proposta apresentada atendeu aos objetivos específicos apresentados ao definir um modelo de armazenamento para o monitoramento em CN, com uma abordagem temporal-intervalar, visando a persistência de dados e riqueza na construção da informação, possibilitando trabalhar com dados completos ao invés de amostras, muitas vezes pouco representativas do todo analisado, como no caso de ambientes altamente dinâmicos como é a essência da CN.

5 IMPLEMENTAÇÃO E ANÁLISE DE EXPERIMENTOS

Após a definição da plataforma buscou-se uma implementação parcial mínima como protótipo inicial. Para maior compreensão do porquê certos elementos da arquitetura foram implementados em detrimento de outros uma breve nota histórica merece ser explicada.

A presente arquitetura foco do presente trabalho é o resultado de pesquisas anteriores no âmbito da análise e predição de SLAs no contexto do monitoramento de computação em nuvem.

Com a evolução da pesquisa identificou-se que existia uma lacuna nas plataformas de monitoramento, onde a extração de informações ou era incompleta para o domínio pesquisado, ou carecia de flexibilidade no armazenamento e recuperação de informações. Neste sentido surge a proposta de modelar-se um mecanismo de armazenamento utilizando bancos de dados não convencionais (NoSQL) e a definição de uma arquitetura onde a análise e predição de SLA não fosse mais um elemento isolado, mas sim, parte integrante de um sistema autônomo onde sua saída seria refletida em um processo de planejamento e execução coordenado.

Desta forma, a implementação da proposta foi guiada pela necessidade de se analisar alguns elementos principais da arquitetura. Devido ao escopo teórico e abrangência de temas e complexidade inerente, não objetivou-se uma implementação completa, mas sim, optou-se na validação do modelo NoSQL de dados e na escolha de métodos analíticos de aprendizado de máquina e sistemas probabilísticos estatísticos adicionando-se a incerteza.

Apesar de uma prototipação parcial, esforços foram realizados no sentido de construir a arquitetura apresentada na Figura 17, cujos elementos serão explicados posteriormente.

A Figura 17 apresenta um *cluster* executando o arcabouço Hadoop, que apresenta dois principais componentes, o sistema de arquivos distribuído e altamente redundante HDFS (Hadoop File System) e o arcabouço de programação distribuída MapReduce, que possibilita a distribuição e paralelização de trabalhos de análise em grandes conjuntos de dados. Em conjunto com o Hadoop, abstraído da representação temos o banco de dados NoSQL HBase, que possui integração com o Hadoop e ferramentas adicionais de manipulação de dados (Pig, Hive, ZooKeeper). Este cluster é alimentado por um nó de coleta de dados, que por sua vez possui uma extensão desenvolvida com o objetivo de

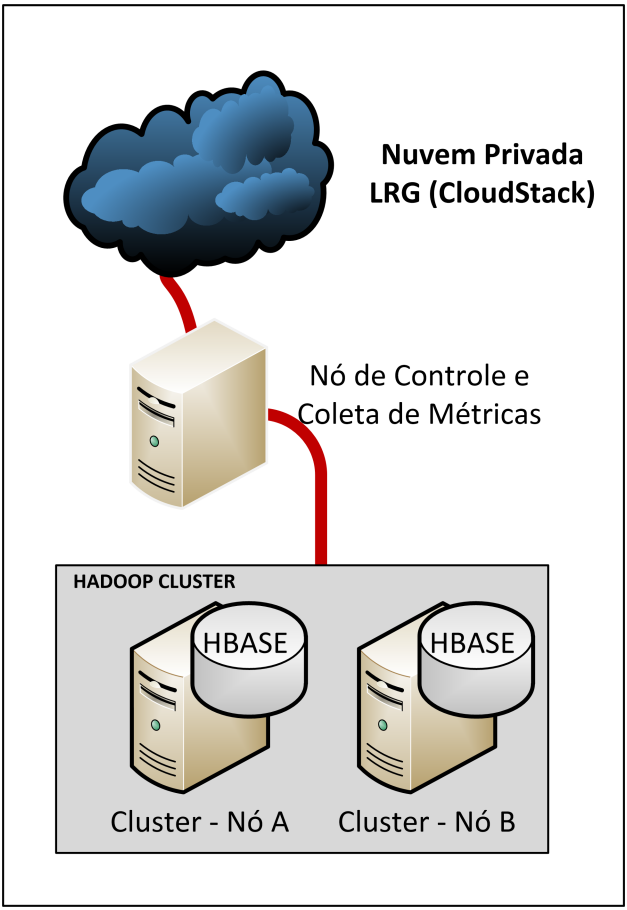

Figura 17 – Visão geral do protótipo.

coletar métricas da plataforma de Nuvem Privada Cloudstack.

O restante do capítulo irá abordar a implementação e testes do mecanismo de armazenamento, apresentado na Figura 17, e a comparação entre métodos de análise de dados para predição de eventos com base no comportamento histórico de um ambiente de testes.

5.1 IMPLEMENTAÇÃO DO MODELO DE ARMAZENAMENTO DE DADOS

Conforme apresentado no Capítulo 4, especificamente na seção 4.1.1, uma base de monitoramento deve aderir as propriedades da taxonomia apresentada no Capítulo 3 e possuir um grande poder de representação.

Devido a estas propriedades, decisões de projetos foram adotadas na direção de uma arquitetura distribuída, baseada em modelos de armazenamento não convencionais e bancos de dados não relacionais. Mais especificamente, optou-se por utilizar-se uma infraestrutura distribuída baseada no arcabouço Apache Hadoop, para assegurar a durabilidade dos dados através da replicação e de um sistema de arquivos redundante (HDFS) e possibilitar a análise distribuída dos dados a partir do arcabouço MapReduce. Esta arquitetura está brevemente explicada na Figura 17.

Para a efetiva implementação do modelo de dados, optou-se por uma solução NoSQL, devido às limitações dos modelos relacionais, já abordada anteriormente. Neste sentido, o banco de dados baseado em colunas (column-store) Apache Hbase foi adotado.

A partir da escolha do banco de dados NoSQL uma série de decisões de implementação foram adotadas, visando adaptar e viabilizar o modelo proposto. Neste sentido alguns desafios surgem.

Para validar o modelo proposto, definiu-se como experimento, a importação dos dados já contidos nas plataformas de monitoramento utilizadas na nuvem de pesquisa do LRG, atualmente sendo a ferramenta de monitoramento nativa do arcabouço de gerenciamento de nuvens Cloudstack e o complemento de monitoramento Zenoss[1]. Ambas soluções baseiam-se em bancos de dados relacionais MySQL, e a tarefa principal está na denormalização das suas tabelas e conversão dos dados para o modelo proposto NoSQL.

A Figura 18 apresenta o modelo relacional a ser importado, onde visualizamos a parcela do banco de dados de controle do Cloudstack referente ao monitoramento de eventos (tabela *event*)e alertas (tabela *alerts*).

Segundo (ALMEIDA, 2014), a escolha da chave de linha e esquema para o HBaseTM é uma das tarefas essenciais para a definição de um bom modelo, a chave de linha, nada mais é do que a chave primária de acesso à tabela. O primeiro problema a se pensar antes de definir

[1] http://www.zenoss.com/

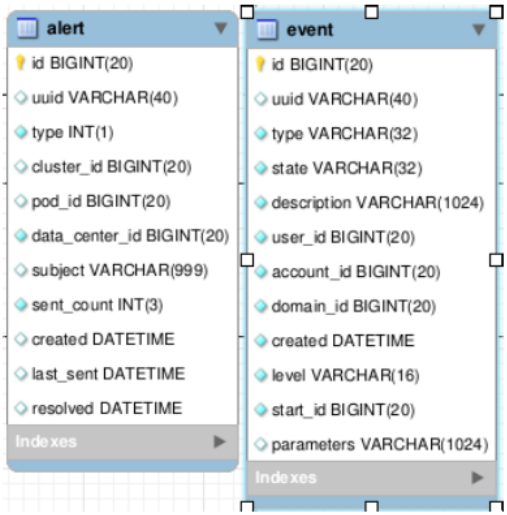

Figura 18 – Tabelas alert e event, conforme obtidas do banco de dados de monitoramento do Apache CloudStack (ALMEIDA, 2014).

a chave de linha é o caso de métricas onde muitas informações são necessárias a serem armazenadas apenas em uma linha, o qual é chamado de *overhead* da linha (GEORGE, 2011). Portanto para adequar a realidade de várias tabelas de um esquema relacional ao HBase é necessário realizar a denormalização de todas as tabelas do banco relacional, para não gerar uma linha com um número de colunas que extrapole o limite de 50MB presente no Hbase (DIMIDUK et al., 2013), e no fim gerarmos outros problemas como, por exemplo baixa performance na busca de informações.

Em linhas gerais, as ações adotadas para a implementação do modelo foram:

1. Denormalização do banco relacional. Para evitar a criação de linhas extremamente grandes devido à modelagem inicial (n tags), gerando problemas de limite e performance, duas tabelas são criadas no HBase, uma contendo as informações das tags, e outra contendo os demais dados.

2. Escolha da chave de linha para acesso aos dados. Neste caso optou-se pelo padrão $OrigemTimestampTag$.

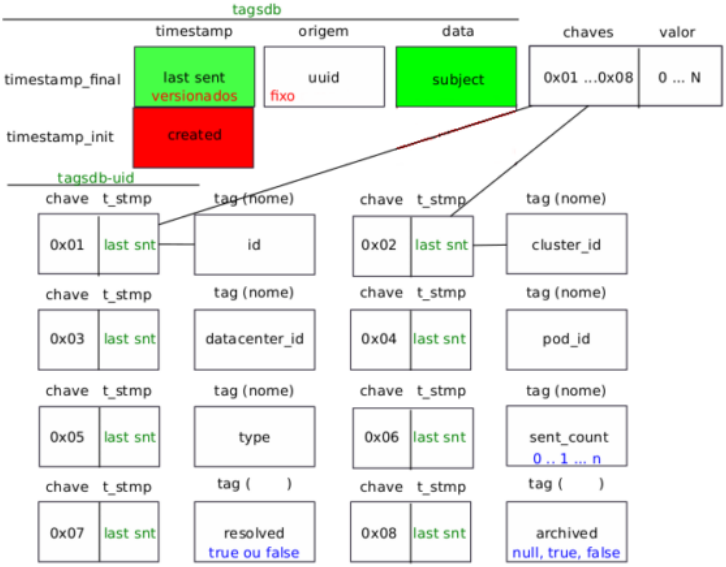

Figura 19 – Mapeamento da tabela *alert* do Cloudstack para o modelo proposto (ALMEIDA, 2014).

3. O mapeamento de campos do modelo proposto na seção 4.1 do capítulo 4 é executado conforme apresentado nas figuras 18 e 19. A figura 18 apresenta o modelo relacional, ou seja, as tabelas contidas no banco de dados relacional utilizado pelo Cloudstack. Já a figura 19 apresenta o modelo NoSQL Hbase.

4. Implementação do modelo NoSQL utilizando o Hbase.

5. População da base HBase com dados reais da base relacional MySQL do Cloudstack e Zenoss.

6. Teste de performance, comparativo entre modelos (relacional e NoSQL).

5.1.1 Análise de Performance

Após a implementação do modelo, construiu-se um pequeno experimento para avaliar a performance obtida com a implementação

NoSQL em relação ao modelo tradicional em MySQL. Para detalhes e código-fonte dos clientes utilizados referir-se a (ALMEIDA, 2014).

O experimento realizado visa avaliar a performance de leitura (READ), escrita (WRITE) e remoção (DELETE) de registros, nos modelos relacional e NoSQL. O ambiente controlado constituiu-se em duas instâncias virtuais de igual capacidade e poder de processamento. Tanto o Hadoop quanto o MySQL foram mantidos em versões *standalone*, ou seja, apenas uma instância, sem replicação ou distribuição do processamento.

A execução do experimento é realizada a partir de clientes implementados em Java utilizando APIs específicas e drivers para a comunicação com MySQL e HBase. Estes clientes, por sua vez efetuam 1000 operações em cada uma das suas bases. Os modelos relacional e NoSQL utilizados são os apresentados nas Figuras 18 e 19, respectivamente.

Utilizou-se como dados as informações ja contidas na base MySQL, de eventos gerados pela Nuvem de pesquisa do LRG. Os mesmos dados foram exportados para o modelo NoSQL.

Os gráficos apresentados nas Figuras 20, 21 e 22 apresentam os resultados das operações em ambos os modelos.

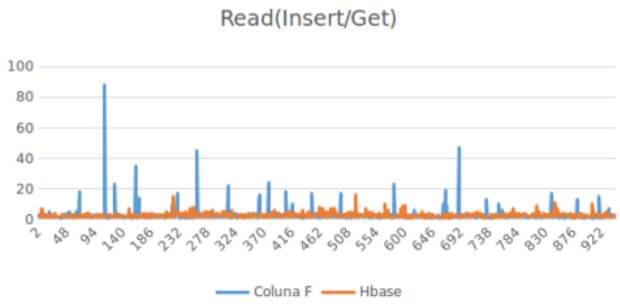

Figura 20 – Resultados de Leitura (ALMEIDA, 2014).

A Figura 20 apresenta os resultados de leitura (READ) obtidos nas implementações relacional e NoSQL. A linha azul representa os resultados obtidos com o MySQL, enquanto a linha laranja os resultados obtidos com o HBase. A escala apresenta o tempo em milissegundos (*ms*) de cada uma das 1000 execuções.

Apesar do pequeno volume de dados, o que de acordo com (DIMIDUK et al., 2013) e (MCCREARY; KELLY, 2013) podem prejudicar a performance do Hadoop e HBase chegando a perder performance em

relação a modelos relacionais, é perceptível a regularidade nas leituras no modelo implementado no HBase em relação ao MySQL. Esta constatação demonstra a aderência do modelo NoSQL a propriedade da pontualidade, onde o dado está disponível quando necessário, e sua recuperação é rápida.

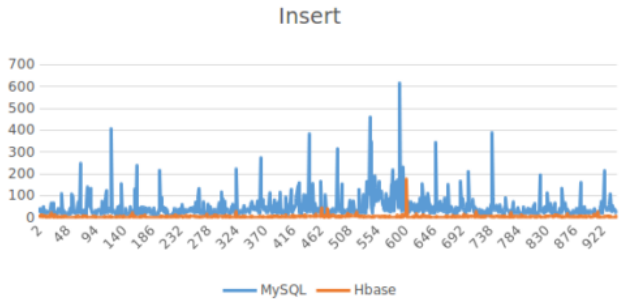

Figura 21 – Resultados de Escrita (ALMEIDA, 2014).

Na Figura 21 os resultados de inserção são apresentados. Novamente vemos que a variabilidade das inserções no MySQL são consistentemente maiores do que no HBase, mesmo considerando um conjunto pequeno de dados. Além disso, nota-se que a latência das inserções são maiores com o MySQL, pondendo ser explicada devido às propriedades ACID que necessitam ser garantidas na inserção.

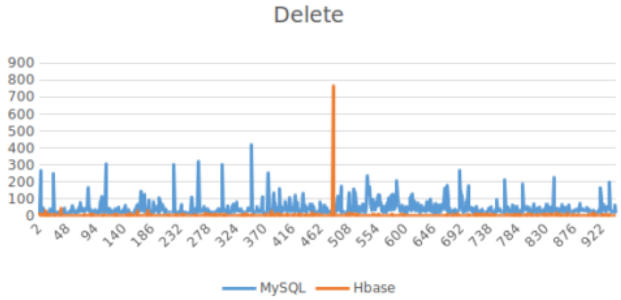

Figura 22 – Resultados de Remoção (ALMEIDA, 2014).

Já a Figura 22 apresenta os resultados das remoções de registros

em ambos modelos. Novamente a consistência do HBase mostrou-se muito superior ao MySQL, da mesma forma a latência. Conforme abordado, esta alta latência do MySQL pode ser explicada pelas verificações de consistência necessárias na remoção de registros em um modelo relacional ACID. Por outro lado vemos um registro que apresentou tempo de resposta bastante elevado no Hbase. Considera-se um ponto discrepante, resultado de alguma operação concorrente na instância no momento da execução.

5.2 IMPLEMENTAÇÃO DE MODELOS PREDITIVOS PARA VIOLAÇÃO DE SLA

Esta seção visa abordar o problema da análise de dados de monitoramento, direcionando a um escopo preditivo na detecção de violações de SLA com base em métricas de monitoramento. Buscou-se avaliar diferentes métodos estatísticos e de aprendizado de máquina com o objetivo de detectar violações de SLA com base no comportamento histórico de uma dada aplicação e regras fixas de violações de SLA e objetivos de garantia de serviço - SLOs.

Neste intuito três soluções foram avaliadas, sendo elas redes neurais multi-camada (MLP), redes bayesianas e sistemas híbridos utilizando redes neurais e conjuntos difusos.

Para os três casos foram expostos os mesmos conjuntos de dados de monitoramento, com os objetivos de detectar a violação de SLA de acordo com regras pré-estabelecidas; avaliar a qualidade do treinamento dos sistemas de aprendizado; verificar a acurácia na detecção e provisionamento de recursos (não aplicável ao sistema bayesiano, onde buscou-se apenas a validação da detecção de SLA).

5.2.1 Implementação do Modelo utilizando Redes Neurais Multi-Camada e Redes Neurais Difusas

Segundo (CARVALHO, 2009), o desenvolvimento de uma aplicação em redes neurais deve seguir as etapas de coleta de dados e separação em conjuntos, configuração da rede, treinamento, teste e integração.

De acordo com o trabalho anterior (SCHUBERT, 2011), a topologia da rede contará conforme a Figura 23 com seis entradas, sendo elas os requisitos não funcionais de QOS mais os indicadores do estado do sistema representados pelo load (carga) do sistema, uso de memória

e de disco. A saída da rede contará com o resultado dos volumes de memória, núcleos de processamento e disco para o provisionamento, além do valor para o SLA, este valor é representado por uma variável booleana que determina se o SLA foi violado.

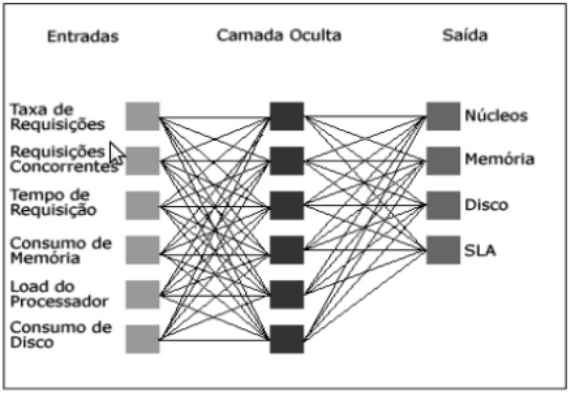

Figura 23 – Modelo neural multi-camada (SCHUBERT, 2011).

As entradas da rede serão os atributos de qualidade de serviço e o estado dos recursos de performance do sistema avaliado. Estes atributos que atuam sobre os recursos do sistema foram definidos levando-se em conta os valores que têm influência direta na performance e na disponibilidade do sistema. Atributos definidos como entrada da rede neural:

- Taxa de requisições: a taxa de requisições é a quantidade de requisições por unidade de tempo. No caso a métrica utilizada foi requisições por segundo (rps).
- Requisições concorrentes: é o volume de requisições simultâneas que o ambiente está recebendo em um momento.
- Tempo de requisição: é o tempo necessário para a requisição ser processada e exibir seu retorno. Mensurada em segundos.
- Consumo de memória: é o volume de memória dinâmica volátil (RAM) consumida pelas requisições no sistema em um momento específico. Exclui a memória utilizada pelo sistema operacional.
- Load do processador: o load de processador é o uso da capacidade de processamento. Os processos em um sistema ficam em espera

ou em execução. O load representa portanto, a soma de todos os processos que estão atualmente em execução mais os processos que estão aguardando na fila de execução para serem executados. O load pode variar de 0 (sem processos rodando ou em fila) a 1.0, ou seja, a fila de execução está cheia, mas não existem processos aguardando o término de um processo para sua execução e valores maiores que 1.0 para o load significam que a fila de execução está cheia e existem processos aguardando execução.

- Consumo de disco: consumo em megabytes das requisições sobre o espaço de armazenamento do sistema.

Após o mapeamento das entradas do sistema, faz-se necessário efetuar a definição das saídas esperadas. Com o intuito de assegurar o SLA e demonstrar o provisionamento de recursos de forma dinâmica, as saídas devem avaliar as entradas e saídas devem demonstrar se houve quebra ou não do SLA, e caso houver sugerir os recursos necessários para que um novo ambiente seja criado e instanciado paralelamente ao que não atende mais aos requisitos de SLA.

- Núcleos: determina a quantidade de núcleos de processamento que devem ser alocados para a nova instância - ou instâncias - virtuais.
- Memória: determina a quantidade de memória a ser alocada para o novo ambiente.
- Disco: informa a quantidade de disco que deve ser alocada ao novo ambiente.
- SLA: valor booleano (verdadeiro ou falso) que determina se o SLA foi violado (verdadeiro) ou não (falso).

Com as entradas e saídas definidas, a primeira análise teve seu foco em redes neurais multi-camada. Para treinar a rede, dados representando a entrada atual e a saída desejada foram utilizados (um valor booleano indicando se o estado operacional - SLA - foi violado, o número de processadores, memória e armazenamento necessário para acomodar o novo estado). Para obter os dados de entrada, scripts em PHP, Shell e Perl foram configurados em um servidor Web Apache e submetido a testes de carga sob diferentes aspectos. O período observado foi de 30 dias. Maiores detalhes sobre a implementação e cenários são encontrados em (ROLIM et al., 2012), (SCHUBERT, 2011) e (SCHUBERT, 2009).

A partir dos dados coletados, uma rede neural foi construída utilizando-se a ferramenta MatLab com seu conjunto de ferramentas para redes neurais. Para testar e validar o poder de generalização do modelo proposto, foi utilizado o método de validação cruzada em 10 vezes (10-fold cross-validation), construindo uma matriz de confusão após o processo.

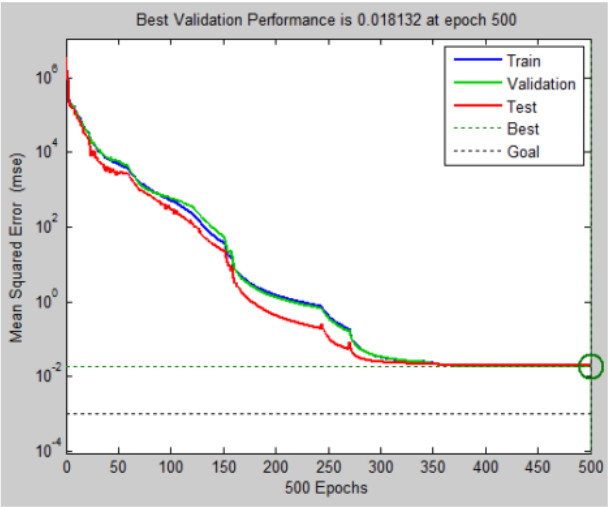

Figura 24 – Treinamento rede neural multi-camada (ROLIM et al., 2012).

A melhor topologia constituiu-se em uma rede neural artificial com 6 nós de entrada, duas camadas intermediárias com 7 e 12 nós respectivamente e a tangente hiperbólica como função de ativação, e finalmente 4 saídas representando os valores preditos. A taxa de aprendizado utilizada foi $1e^{-}3$. Com esta configuração o erro médio quadrado (MSE) foi de 0.0188 (RMSE=0.1371) após 500 épocas de treinamento, representado pelo gráfico da Figura 24.

Para construir o modelo neuro-fuzzy, o mesmo conjunto de dados de entrada e saída utilizado para o modelo neural multi-camada foi utilizado. Este método utilizou o modelo multi adaptativo de inferência (MANFIS), onde os parâmetros internos da rede são configurados automaticamente na ferramenta MatLab. Entretanto, o construtor da rede ainda necessita selecionar o melhor método de partição dos dados de entrada (ROLIM et al., 2012).

No caso da MLP, a melhor configuração foi atingida com um raio

de agregação (cluster radius) de 0.15 para SLA, memória e processamento, e 0.5 para a armazenamento. Para propósitos de comparação, utilizamos um conjunto de treinamento com 500 épocas em ambos os casos (MLP e MANFIS). Com esta configuração, o RMSE resultante da MANFIS é de 0.0707, Figura 25 em contraste com um RMSE de 0.1371 no MLP.

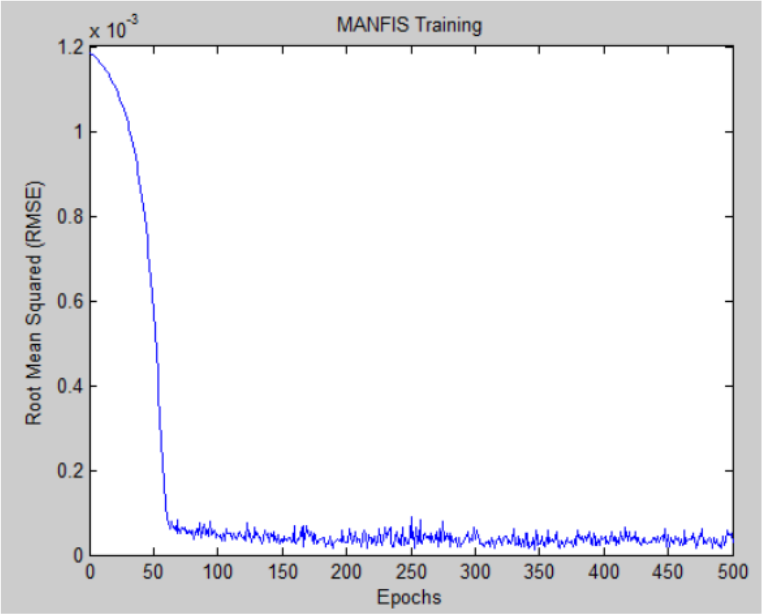

Figura 25 – Treinamento rede MANFIS (ROLIM et al., 2012).

Ademais, em relação ao RMSE menor do modelo MANFIS em relação ao MLP, se observarmos a convergência dos gráficos vemos que a convergência do modelo MANFIS ocorre a uma velocidade maior do que MLP em seu treinamento. No modo de treinamento foi possível concluir que o modelo MANFIS apresenta um poder maior de predição devido ao seu baixo RMSE. Para validar o poder das duas abordagens no contexto do monitoramento de CN, um cenário de teste foi adotado.

A Tabela 9 apresenta os cenários de teste utilizados, estes cenários contém, basicamente, os valores de entrada e saída esperados a partir de mecanismos de predição sob diferentes cenários simulando a carga de um servidor real em situações de normalidade e excesso de capacidade.

Tabela 9 – Cenários de teste de acordo com (ROLIM et al., 2012).

Métrica	**Cenário I**	**Cenário II**	**Cenário III**
Requisições Concorrentes	85	287	958
Requisições por Segundo	225	163	328
Tempo por requisição	62	193	230
Uso de Memória Uso de Armazenamento	64.2 85	287 287	434 434
Carga do Sistema	0.8	3.8	10.4
SLA Esperado	0	1	1
Memória Provisionada Esperada	0	347.84	562.56
Armazenamento Provisionado Esperado	0	4383	5054
Núcleos Provisionados	0	4	10

Estes cenários foram gerados a partir de dados históricos obtidos do ambiente de testes para as entradas e as saídas foram obtidas com base nas funções de alocação:

$$M_Al = 128MB + MpReq$$

$$S_Al = 4096MB + SpReq$$

$$C_Al = Reqs * 0.02$$

A função M_Al representa a memória a ser alocada e é constituída por 128 MB representando a instância base (sistema operacional e processos básicos) mais a quantidade de memória necessária para atender o número de requisições. A função S_Al, por sua vez, representa a função de armazenamento, onde 4096 MB representam o armazenamento base da instância mais espaço para cada requisição, ja a função C_Al é responsável pela alocação de núcleos, onde cada requisição, após análise feita representa um incremento de 0.02 no carga do sistema. As constantes MPReq e SPReq foram fixadas arbitrariamente em 1MB.

Os valores de entrada da Tabela 9 foram utilizados como entradas das redes neurais (MLP e MANFIS). As saídas obtidas estão apresentadas na tabela da Figura 26.

Os resultados apresentados na tabela da Figura 26, possibilitam algumas análises:

- Cenário I: os valores obtidos da predição mostram um servidor com carga normal, sem violações de SLA. As predições obtidas

	MLP Provisioning			*MANFIS Provisioning*		
Scenario / *Attributes*	*Scenario 1*	*Scenario 2*	*Scenario 3*	*Scenario 1*	*Scenario 2*	*Scenario 3*
SLA Status Predicted / Expected	**0** / 0	**1** / 1	**1** / 1	**0** / 0	**1.009** / 1	**1.00** / 1
Memory Predicted / Expected	**0.0002** / 0	**431.2512** / 347.84	**408.38**/ 562.56	**0.0027** / 0	**345.30** / 347.84	**563.49**/562.56
Storage Predicted / Expected	**0** / 0	**4383** / 4383	**4557** / 5054	**0.0018** / 0	**4383.7** / 4383	**5058.25** /5054
Cores Predicted / Expected	**0** / 0	**3** / 4	**9** / 10	**0.0087** / 0	**6.12** / 4	**10.1732** / 10

Figura 26 – Resultados obtidos dos diferentes cenários (ROLIM et al., 2012).

com a MLP e MANFIS são aceitáveis tendo em vista uma pequena margem de precisão.

- Cenário II: os atributos de SLA violado e armazenamento cujos valores foram preditos corretamente por ambas abordagens. Na abordagem MLP, o número de CPUs foi arredondado para baixo, e a memória predita 23% acima do esperado, por outro lado, a rede MANFIS teve uma previsão de 15% a mais para processadores e a memória 1% acima do esperado.

- Cenário III: ambas abordagens previram corretamente a violação de SLA. Entretando, a MLP errou no número de núcleos, a memória foi predita 27% abaixo do esperado e o armazenamento aproximadamente 10% a menos do esperado. Na abordagem MANFIS, memória, armazenamento e CPUs foram preditos com 1% a mais do esperado, o que não constitui uma variação relevante e demonstra uma maior acurácia do modelo MANFIS.

Em resumo, ambas abordagens mostraram-se satisfatórias na previsão da violação de SLA, porém a abordagem MANFIS, devido ao tratamento eficiente da incerteza ao abordar o hibridismo difuso, mostrou-se mais eficiente na proposta de alocação de recursos.

5.2.2 Implementação do Modelo utilizando Redes Bayesianas

Tendo em vista a necessidade de um modelo que avalie o estado atual da instância e detecte flutuações no SLA, propõe-se um modelo que utilize redes bayesianas como mecanismo de diagnóstico e detecção

de violações das cláusulas dos SLAs.

As redes bayesianas foram escolhidas pois, a aleatoriedade contida no problema das violações de SLAs será abordada, pois, efetivamente, dependem da atenção e disposição humana em diagnosticar e dar consequência às violações. Ainda, as redes bayesinas disponibilizam um robusto ferramental matemático para propagação das crenças bem como servem para implementação para sistemas autônomos, permitindo inclusive, a utilização de técnicas de aprendizagem de máquina aprendizagem da topologia e probabilidades a priori (FRIEDMAN; NACHMAN; PEÉR, 1999).

Nó	Estado	Limiar
MEMÓRIA	OK	menor que 80% da capacidade
	WARNING	maior igual a 80% e menor que 90% da capacidade
	CRITICAL	maior igual a 90% da capacidade
PROCESSAMENTO	OK	menor que 90%
	WARNING	maior igual a 90% e menor que 100%
	CRITICAL	igual a 100%
ARMAZENAMENTO	OK	menor que 90%
	WARNING	maior igual a 90% e menor que 95%
	CRITICAL	maior igual a 95%
CONSUMO DE BANDA	OK	menor que 90%
	WARNING	maior igual a 90% e menor que 95%
	CRITICAL	maior igual a 95%
HOST	UP	Host Ativo
	DOWN	Host Inativo
SLA	VIOLADO	Contrato Quebrado
	MANTIDO	Contrato Preservado
SLA GERAL	VIOLADO	Contrato Quebrado
	MANTIDO	Contrato Violado

Figura 27 – Nós da rede bayesiana (SCHUBERT; MENDES; WESTPHALL, 2013).

A tabela da Figura 27 apresenta as entradas da rede. Os nós de entrada são MEMÓRIA, PROCESSAMENTO, ARMAZENAMENTO, CONSUMO DE BANDA e HOST. A coluna Estado apresenta os limiares permitidos para a detecção de uma violação de SLA. Já o limiar delimita os intervalos aceitáveis para cada estado.

A rede *bayesiana* foi construída utilizando o arcabouço de desenvolvimento de redes bayesianas Netica, uma das principais ferramentas para a construção de sistemas de inferência probabilísticos, segundo (HOPE; NICHOLSON; KORB, 2002).

O treinamento da rede se deu a partir da utilização de um conjunto de 100 casos que representam o monitoramento constante de uma instância virtual, contendo os dados em fatias de tempo de 5 minutos e

em situações simuladas de processamento e disponibilidade. Relatórios obtidos a partir do treinamento baseado em Gradiente Descendente (BALDI, 1995) e apresentados na Tabela 10.

Tabela 10 – Iterações de treinamento da rede.

Iteração	Probabilidade	Mudança %
0	6,50737	
1	5,89169	9,4613
2	5,73948	2,9229
3	5,58664	2,3226
4	5,56284	0,4260
5	5,53936	0,4221
6	5,52387	0,2797
7	5,51191	0,2165
8	5,51007	0,0335
9	5,5082	0,0338
10	5,50659	0,0294
11	5,5058	0,0142
12	5,50542	0,0069
13	5,50533	0,0016
14	5,50526	0,0013
15	5,50522	0,0007
16	5,5052	0,0003
17	5,50516	0,0007
18	5,50516	0,0001
19	5,50516	0,0001
20	5,50516	0,0001
21	5,50514	0,0000
22	5,50515	-0,0000

A diferença no cálculo de probabilidades atingiu 0.0 pontos após 22 ciclos de treinamento, ou seja, o estado ótimo. A partir deste ponto a continuação no treinamento não trará melhorias na performance da rede. Probabilidades iniciais calculadas com base no treinamento estão detalhadas na Tabela 11.

Após a construção e treinamento da rede *bayesiana*, um conjunto de inferências será realizado para validar a rede. A Tabela 11 repre-

Tabela 11 – Probabilidades dos nós da rede.

Nó	**Estado**	**Limiar**
MEMÓRIA	OK	0.34
	WARNING	0.32
	CRITICAL	0.34
PROCESSAMENTO	OK	0.36
	WARNING	0.36
	CRITICAL	0.28
ARMAZENAMENTO	OK	0.33
	WARNING	0.34
	CRITICAL	0.33
CONSUMO DE BANDA	OK	0.34
	WARNING	0.35
	CRITICAL	0.31
HOST	UP	0.8
	DOWN	0.2
SLA	VIOLADO	0.4311
	MANTIDO	0.5890
SLA GERAL	VIOLADO	0.4400
	MANTIDO	0.5600

senta a matriz de regras contendo as regras de inferência e o resultado esperado dados os valores de entrada nos nós. As regras de inferência da Tabela 11 representam as definições dos SLAs para a instância.

Tabela 12 – Matriz de confusão.

Caso	Memoria	Processamento	Armazenamento	Banda	Host	SLA Geral
1	WARNING	CRITICAL	WARNING	WARNING	UP	MANTIDO
2	OK	OK	OK	OK	DOWN	VIOLADO
3	OK	WARNING	CRITICAL	CRITICAL	UP	VIOLADO
4	CRITICAL	CRITICAL	WARNING	OK	UP	MANTIDO
5	WARNING	WARNING	WARNING	WARNING	UP	MANTIDO

Com base na matriz de confusão apresentada na Tabela 11, foram efetuadas inferências sobre a rede bayesiana implementada. A Tabela 12 apresenta os resultados obtidos a partir da simulação dos valores da matriz de confusão.

Tabela 13 – Resultados dos casos de teste da Matriz de Confusão

Caso	SLA Geral	Mantido %	Violado %
1	MANTIDO	66,2	33,8
2	VIOLADO	19,6	80,4
3	VIOLADO	28,2	71,8
4	MANTIDO	84	16
5	MANTIDO	75,5	24,5

A partir dos resultados pode-ser perceber que apesar de ter corretamente predito todos os casos, as probabilidades associadas com os eventos de manutenção ou violação apresentaram variações, especialmente no caso 1.

5.3 CONCLUSÃO

Este capítulo apresentou a validação prática de elementos relevantes da proposta. Em primeiro lugar buscou-se a implementação da base de monitoramento, onde os dados são armazenados.

Os testes e resultados apresentados demonstraram-se válidos na construção de um modelo NoSQL e sua superioridade performática em relação a modelos tradicionais relacionais.

Na etapa de análise, varios trabalhos anteriores foram coleta-

dos e resumidos visando apresentar soluções para a predição de SLA e alocação de recursos na CN.

6 CONCLUSÃO

A adoção e o uso massivo da CN tornou-se uma realidade. A transformação causada pelo seu advento e consolidação tem transformado a cadeia produtiva e de pesquisa, apresentando novos desafios e possibilitando a execução de processos antes inviáveis computacional ou economicamente.

A presente dissertação buscou abordar, a automação de todo o processo de monitoração, análise, planejamento e execução de tarefas tendo como base a computação autônoma, e a extração de informações da miríade de sensores e pontos de coleta de dados existentes.

Como apresentado, o monitoramento tem um papel essencial na CN. Seu papel vai muito além da visão tradicional de um sistema de alertas ou diagnóstico reativo. Em sua taxonomia, (ACETO et al., 2013c) apresentou uma série de propriedades inerentes ao monitoramento em CN, detalhadas na Seção 3.2, os quais buscou-se atingir com a presente proposta.

O trabalho buscou contribuir com o avanço no estado da arte ao propor uma abordagem autônoma, utilizando os conceitos da computação cognitiva, técnicas de armazenamento NoSQL e temporalidade para abordar o problema do monitoramento em nuvem.

Devido à complexidade do tema e à abrangência, buscou-se focar no armazenamento e acesso aos dados, e a persistência destes dados de forma resiliente e redundante. O paralelismo foi abordado ao utilizar uma plataforma compatível com computação distribuída (Apache Hadoop, HBase e o arcabouço Map Reduce). Ademais, foi integrada à fase de análise trabalhos anteriores de validação e predição de violações de SLA utilizando redes bayesianas e um comparativo entre redes bayesianas, redes neurais e redes neuro-fuzzy.

A amplitude do tema dificultou a definição de um escopo que satisfizesse aos requisitos para propor um modelo aderente à taxonomia de Aceto (ACETO et al., 2013c) ao mesmo tempo possibilitando um detalhamento adequado. Optou-se portanto na definição de uma arquitetura e de um modelo de dados geral, agregando a ele métodos de análise de dados avaliados em trabalhos anteriores. A implementação de um protótipo completo tornou-se inviável devido ao limite de tempo e recursos disponíveis.

6.0.1 Principais Contribuições

Conforme será abordado na seção de Trabalhos Futuros, que aborda as inúmeras possibilidades acadêmicas e de negócio possíveis a partir deste modelo, o presente modelo apresentou uma visão e propôs soluções para o problema do monitoramento em ambientes de computação em nuvem. As principais contribuições foram:

- Abordar o estado da arte no monitoramento de computação em nuvem.
- Apresentar os conceitos e a visão da computação cognitiva e análise de grandes conjuntos de dados (*Big Data*).
- Propor uma arquitetura autônoma de monitoramento e gerenciamento de CN, como evolução dos atuais sistemas e infraestruturas como serviço.
- Apresentar um modelo de dados para armazenamento de dados e métricas de monitoramento
- Apresentar um modelo aderente às propriedades de monitoramento de CN propostas por Aceto (ACETO et al., 2013c).
- Propor uma arquitetura de monitoramento aderente ao ciclo autônomo MAPE-K
- Efetuar uma avaliação entre métodos de aprendizado de máquina e sistemas especialistas para a predição e detecção de violações de SLA.

Considera-se que a presente dissertação contribuiu satisfatoriamente com o avanço do estado da arte, a partir da avaliação da proposta apresentada, os estudos preliminares alcançados e o amplo leque de possibilidades para trabalhos futuros apresentados.

Porém, é notório salientar que o tema é extenso, e em cada etapa do monitoramento, sejam elas definição de agentes, configuração e ajuste de métricas, coleta e agregação de fontes, armazenamento e posterior análise, existem desafios e temas ainda não explorados exaustivamente que necessitam ser atacados com o devido rigor científico.

6.1 TRABALHOS FUTUROS

A partir da proposta apresentada surgem diversas possibilidades de extensão e aprimoramento do modelo.

- Implementar o modelo de dados como parte dos projetos CloudStack e (ou) OpenStack, como complemento ao *dashboard* de monitoramento atual
- Desenvolver extensões para coleta de dados de diferentes fontes (SNMP, WMI) e logs de aplicativos (formato de Logs do Apache, Tomcat).
- Concluir o ciclo autônomo, implementando as fases de análise, planejamento e execução, juntamente com a definição da base de regras de negócio, onde os objetivos gerais são gerenciados.
- Criar o meta gerenciamento e a garantia da integridade das propriedades auto do modelo autônomo.
- Tratar a coleta de dados e o processamento de fluxos com estruturas mais eficientes do que filas (como por exemplo o mecanismo Kinesis (AMAZON..., 2014)).

Estes são apenas alguns desafios para trabalhos futuros baseados na presente proposta.

REFERÊNCIAS

ACETO, G. et al. Cloud Monitoring : definitions , issues and future directions. 2013.

ACETO, G. et al. Cloud Monitoring: a Survey. *Computer Networks*, 2013. <http://wpage.unina.it/pescape/doc/CMv02_jan2013_R1.pdf>.

ACETO, G. et al. Survey cloud monitoring: A survey. *Comput. Netw.*, Elsevier North-Holland, Inc., New York, NY, USA, v. 57, n. 9, p. 2093–2115, jun. 2013. ISSN 1389-1286. <http://dx.doi.org/10.1016/j.comnet.2013.04.001>.

ALAETTINOGLU, C. Software defined networking. 2013.

ALMEIDA, M. G. d. Armazenamento big data no monitoramento em nuvem. 2014.

AMAZON Kinesis. 2014. <http://aws.amazon.com/pt/kinesis/>.

ARMBRUST, M. et al. *Above the Clouds: A Berkeley View of Cloud Computing*. [S.l.], Feb 2009. <http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html>.

ARTIFICIAL intelligence meets business intelligence. 2014. <http://www.research.ibm.com/cognitive-computing>.

ASSUNCAO, M. D. et al. Big data computing and clouds: Challenges, solutions, and future directions. *arXiv preprint arXiv:1312.4722*, 2013.

BALDI, P. Gradient descent learning algorithm overview: a general dynamical systems perspective. *Neural Networks, IEEE Transactions on*, v. 6, n. 1, p. 182–195, 1995. ISSN 1045-9227.

BODIK, P. et al. Statistical machine learning makes automatic control practical for internet datacenters. In: *Proceedings of the 2009 conference on Hot topics in cloud computing*. [S.l.: s.n.], 2009. p. 12–12.

BOUTILIER, C.; DEAN, T.; HANKS, S. Decision-theoretic planning: Structural assumptions and computational leverage. *arXiv preprint arXiv:1105.5460*, 1999.

BUYYA, R.; CALHEIROS, R. N.; LI, X. Autonomic Cloud computing: Open challenges and architectural elements. *2012 Third International Conference on Emerging Applications of Information Technology*, Ieee, p. 3–10, nov. 2012. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6407847>.

BUYYA, R.; PANDEY, S.; VECCHIOLA, C. Cloudbus toolkit for market-oriented cloud computing. In: *Proceedings of the 1st International Conference on Cloud Computing*. Berlin, Heidelberg: Springer-Verlag, 2009. (CloudCom '09), p. 24–44. ISBN 978-3-642-10664-4.

BUYYA, R. et al. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems*, Elsevier, v. 25, n. 6, p. 599–616, 2009.

CARVALHO, A. *Redes Neurais Artificiais*. 2009. <http://www.icmc.usp.br/pessoas/andre/research/neural/index.htm>.

CHAVES, S. A. de; URIARTE, R. B.; WESTPHALL, C. B. Implantando e monitorando uma nuvem privada. In: *VIII Workshop em Clouds, Grids e Aplicaç oes*. [S.l.: s.n.], 2010.

CHAVES, S. D.; URIARTE, R.; WESTPHALL, C. Toward an architecture for monitoring private clouds. *Communications Magazine, IEEE*, v. 49, n. 12, p. 130–137, December 2011. ISSN 0163-6804.

DEAN, J.; GHEMAWAT, S. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, ACM, v. 51, n. 1, p. 107–113, 2008.

DIMIDUK, N. et al. *HBase in action*. [S.l.]: Manning Shelter Island, 2013.

DOBSON, S. et al. Fulfilling the vision of autonomic computing. *Computer*, p. 35–41, 2010. <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5398781>.

DOBSON, S. et al. Fulfilling the vision of autonomic computing. *Computer (New York)*, v. 43, n. 1, p. 35–41, 2010.

DUNCAN, D. et al. The structure of the new it frontier: Cloud computing–part i. *Strategic Faciities Magazine-Pacific and Strategic Holdings Pte Ltd: Singapore*, p. 67–72, 2009.

EMEAKAROHA, V. C. et al. Towards autonomic detection of SLA violations in Cloud infrastructures. *Future Generation Computer Systems*, Elsevier B.V., v. 28, n. 7, p. 1017–1029, jul. 2012. ISSN 0167739X. <http://linkinghub.elsevier.com/retrieve/pii/S0167739X11002184>.

FOSTER, I. et al. Cloud Computing and Grid Computing 360-Degree Compared. *2008 Grid Computing Environments Workshop*, Ieee, p. 1–10, nov. 2008.

FOUNDATION, A. S. *What can Apache CloudStack do?* 2014. <http://docs.cloudstack.apache.org/en/master/concepts.html/what-can-apache-cloudstack-do>.

FRIEDMAN, N.; NACHMAN, I.; PEÉR, D. Learning bayesian network structure from massive datasets: the «sparse candidate «algorithm. In: MORGAN KAUFMANN PUBLISHERS INC. *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*. [S.l.], 1999. p. 206–215.

GEORGE, L. *HBase: the definitive guide*. [S.l.]: "O'Reilly Media, Inc.", 2011.

GRIMES, J. M.; JAEGER, P. T.; LIN, J. Weathering the storm: The policy implications of cloud computing. *available at nora. lis. uiuc. edu/images/iConferences/CloudAbstract1310 9FINAL. pdf*, 2009.

GUTIÉRREZ, S. A.; BRANCH, J. W. A comparison between expert systems and autonomic computing plus mobile agent approaches for fault management. una comparación entre los enfoques basados en sistemas expertos y computación autónoma más. *Dyna*, v. 78, n. 168, p. 173–181, 2011.

HOEFER, C. N.; KARAGIANNIS, G. Taxonomy of cloud computing services. In: *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*. [S.l.: s.n.], 2010. p. 1345–1350.

HOPE, L.; NICHOLSON, A.; KORB, K. *Knowledge engineering tools for probability elicitation*. [S.l.], 2002.

HUEBSCHER, M. C.; MCCANN, J. A Survey of Autonomic Computing. *ACM Computing Surveys*, v. 40, n. 3, p. 1–28, ago. 2008. <http://portal.acm.org/citation.cfm?doid=1380584.1380585>.

IBM; ZIKOPOULOS, P.; EATON, C. *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*. 1st.

ed. [S.l.]: McGraw-Hill Osborne Media, 2011. ISBN 0071790535, 9780071790536.

KAI, L. et al. Scm: A design and implementation of monitoring system for cloudstack. In: *Cloud and Service Computing (CSC), 2013 International Conference on*. [S.l.: s.n.], 2013. p. 146–151.

KELLY, J.; HAMM, S. *Smart Machines: IBM's Watson and the Era of Cognitive Computing*. Columbia University Press, 2013. (Columbia Business School Publishing Series). ISBN 9780231537278. <http://books.google.com.br/books?id=tS8CAQAAQBAJ>.

KENNEDY, C. M. Distributed meta-management for self-protection and self-explanation. *SCHOOL OF COMPUTER SCIENCE RESEARCH REPORTS-UNIVERSITY OF BIRMINGHAM CSR*, University of Birmingham; 1999, v. 3, 2008.

KEPHART, J.; CHESS, D. The vision of autonomic computing. *Computer*, n. January, p. 41–50, 2003. <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1160055>.

KHAJEH-HOSSEINI, A.; SOMMERVILLE, I.; SRIRAM, I. Research challenges for enterprise cloud computing. *CoRR*, abs/1001.3257, 2010.

KUTARE, M. et al. Monalytics: online monitoring and analytics for managing large scale data centers. In: ACM. *Proceedings of the 7th international conference on Autonomic computing*. [S.l.], 2010. p. 141–150.

LOCKHEED, M. type, *The Intersection of Open Source and the Cloud*. 2010.

MANYIKA, J. et al. *Big data: The next frontier for innovation, competition, and productivity*. maio 2011.

MCAFEE, A.; BRYNJOLFSSON, E. et al. Big data: the management revolution. *Harvard business review*, v. 90, n. 10, p. 60–66, 2012.

MCCREARY, D.; KELLY, A. *Making Sense of NoSQL*. [S.l.: s.n.], 2013.

MELL, P.; GRANCE, T. The NIST definition of Cloud Computing. 2011.

MENDES, R. d. S.; WESTPHALL, C. M. Um modelo probabilÃstico de auto-proteÃ§Ã£o em nuvens de computador. 2014.

MEYER, J. Two-moment decision models and expected utility maximization. *The American Economic Review*, JSTOR, p. 421–430, 1987.

MONGIN, P. Expected utility theory. *Handbook of economic methodology*, Edward Elgar London, p. 342–350, 1997.

MORENO-VOZMEDIANO, R.; MONTERO, R.; LLORENTE, I. Key challenges in cloud computing: Enabling the future internet of services. *Internet Computing, IEEE*, v. 17, n. 4, p. 18–25, July 2013. ISSN 1089-7801.

NURMI, D. et al. The Eucalyptus Open-source Cloud-computing System. 2008.

OPENNEBULA. 2013.

OPENSTACK. 2013.

ROLIM, C. O. et al. Comparison of a multi output adaptative neuro-fuzzy inference system (manfis) and multi layer perceptron (mlp) in cloud computing provisioning. In: *29th Brazilian Symposium on Computer Networks and Distributed Systems, Paris*. [S.l.: s.n.], 2012.

RUSSOM, P. Big data analytics. *TDWI Best Practices Report, Fourth Quarter*, 2011.

SCHUBERT, F. AplicaÃ§Ã£o de algoritmos de provisionamento baseados em contratos de nÃvel de serviÃ§o para computaÃ§Ã£o em nuvem. 2009.

SCHUBERT, F. AplicaÃ§Ã£o de Algoritmos de Provisionamento Baseados em Contratos de NÃvel de ServiÃ§o para ComputaÃ§Ã£o em Nuvem. *... Simposio Brasileiro de ...*, 2011.

SCHUBERT, F.; MENDES, R.; WESTPHALL, C. B. Redes bayesianas para a deteÃ§Ã£o de violaÃ§Ã£o de sla em infraestrutura como serviÃ§o. 2013.

SLOMAN, A. Varieties of affect and the cogaff architecture schema. In: *Proceedings of the AISB 01 symposium on emotions, cognition, and affective computing. The Society for the Study of Artificial Intelligence and the Simulation of Behaviour*. [S.l.: s.n.], 2001.

SPRING, J. Monitoring cloud computing by layer, part 1. *IEEE Security and Privacy*, 2011.

STERRITT, R.; BUSTARD, D. Autonomic computing - a means of achieving dependability? In: *Engineering of Computer-Based Systems, 2003. Proceedings. 10th IEEE International Conference and Workshop on the*. [S.l.: s.n.], 2003. p. 247–251.

TURING, A. M. Computing machinery and intelligence. *Mind*, LIX, n. 236, p. 433–460, 1950. <http://mind.oxfordjournals.org/content/LIX/236/433.short>.

ULLAH, S.; ZHENG, X. Cloud computing research challenges. *CoRR*, abs/1304.3203, 2013.

VIEIRA, K. M. M. et al. Autonomic intrusion detection system in cloud computing with big data. 2014.

WEI, Y.; BLAKE, M. B. Service-oriented computing and cloud computing: Challenges and opportunities. *IEEE Internet Computing*, v. 14, n. 6, 2010.

ZHANG, Q.; CHENG, L.; BOUTABA, R. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, Springer-Verlag, v. 1, n. 1, p. 7–18, maio 2010. ISSN 1867-4828. <http://dx.doi.org/10.1007/s13174-010-0007-6>.

ZHANG, R.; BIVENS, A. Comparing the use of bayesian networks and neural networks in response time modeling for service-oriented systems. *... of the 2007 workshop on Service-oriented ...*, p. 67–74, 2007. <http://dl.acm.org/ft_gateway.cfm?id=1272467&type=pdf>.