UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

REDE BAYESIANA EMPREGADA NO GERENCIAMENTO DA SAÚDE DOS SISTEMAS NA COMPUTAÇÃO EM NUVEM

RENATO DOS SANTOS ALVES

ORIENTADOR: PROF. DR. CESAR AUGUSTO CAVALHEIRO MARCONDES

São Carlos – SP

Julho/2016

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

REDE BAYESIANA EMPREGADA NO GERENCIAMENTO DA SAÚDE DOS SISTEMAS NA COMPUTAÇÃO EM NUVEM

RENATO DOS SANTOS ALVES

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação, área de concentração: Sistemas Distribuídos e Redes de Computadores

Orientador: Prof. Dr. Cesar Augusto Cavalheiro Marcondes

São Carlos – SP

Julho/2016

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária UFSCar
Processamento Técnico
com os dados fornecidos pelo(a) autor(a)

Alves, Renato dos Santos
A474r Rede Bayesiana empregada no gerenciamento da saúde dos sistemas na computação em nuvem / Renato dos Santos Alves. -- São Carlos : UFSCar, 2016.
113 p.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2016.

1. IaaS. 2. Mineração de dados. 3. Computação em nuvem. 4. Rede bayesiana. 5. Saúde de sistemas. I. Título.

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a defesa de Dissertação de Mestrado do candidato Renato Santos Alves, realizada em 10/08/2016.

Prof. Dr. César Augusto Cavalheiro Marcondes
(UFSCar)

Prof. Dr. Hermes Senger
(UFSCar)

Prof. Dr. Eleri Cardozo
(UNICAMP)

A Deus,
a meus gestores Fiori, Marcio e Célio,
e ao meu orientador Cesar Marcondes.

AGRADECIMENTOS

Agradeço a Deus pela minha saúde, e a minha avó por ensinar o real significado da palavra AMOR.

Sou grato ao Prof. Dr. Cesar Augusto Cavalheiro Marcondes pelo incentivo e força que me proporcionou neste período, e que teve papel fundamental na elaboração deste trabalho. Agradeço também a todos aqueles que, direta ou indiretamente, contribuíram no decorrer desta caminhada.

Não precisamos de mais dinheiro, não precisamos de mais sucesso ou fama, não precisamos do corpo perfeito, nem mesmo do parceiro perfeito. Agora mesmo, neste momento exato, dispomos da mente, que é todo o equipamento básico de que precisamos para alcançar a plena felicidade.

Dalai Lama

RESUMO

A computação em nuvem é um modelo de computação conveniente, pois permite a ubiquidade, com acesso sob demanda a um conjunto de recursos configuráveis e compartilhados, que podem ser rapidamente provisionados e disponibilizados com o mínimo de esforço ou interação com o fornecedor do serviço. IaaS é uma maneira diferente de entregar computação em nuvem, onde a infraestrutura de servidores, sistemas de rede, armazenamento e todo o ambiente necessário para o funcionamento do sistema operacional até aplicação são contratados como serviços. Entretanto, empresas tradicionais ainda possuem dúvidas com relação à transferência de seus dados para fora dos limites da corporação. A saúde de sistemas em computação em nuvem é algo fundamental para o negócio, e dada a complexidade dos sistemas é difícil garantir que todos os serviços e recursos funcionem adequadamente. A fim de garantir um gerenciamento mais adequado da saúde dos sistema e serviços na nuvem, propôs-se nesse trabalho uma arquitetura de diagnóstico de saúde de sistema de nuvem. A arquitetura foi modularizada, especializando funções de monitoramento, mineração de dados e inferência com rede Bayesiana. Nessa arquitetura, são fundamentais os registros de eventos de monitoramento dos sistemas e recursos computacionais, pois os dados registrados são minerados para identificar padrões de falhas. Para mineração dos dados de monitoramento foram propostos dois algoritmos: um para realizar a tarefa de pré-processamento dos dados e outro para realizar a transformação dos dados. Como produto da mineração dos dados, foram obtidos conjuntos de dados que foram o insumo para criar a rede Bayesiana. Por meio de algoritmos de aprendizagem estrutural e paramétrica foram criadas redes Bayesinas para cada sistema e disponibilizados por meio da computação em nuvem. A rede Bayesiana tem o objetivo de auxiliar na tomada de decisão com prevenção, previsão, correção de falhas nos sistemas e serviços, permitindo assim gerenciar a saúde e o desempenho dos sistemas de forma mais adequada. Para verificar a aderência da arquitetura ao diagnóstico de falhas, validou-se a precisão de inferência da rede Bayesiana com o método de validação cruzada.

Palavras-chave: computação em nuvem, IaaS, monitoramento, mineração de dados, aprendizagem estrutural, aprendizagem paramétrica, rede bayesiana, inferência, saúde de sistemas.

ABSTRACT

Cloud computing is a convenient computing model, because it allows the ubiquity with on-demand access to a set of configurable and shared features, that can be rapidly provisioned and made available with minimal effort or interaction with the service provider. IaaS is a different way to deliver cloud computing, where infrastructure servers, networking systems, storage, and all the necessary environment for the operating system to run the application are hired as services. Meanwhile, traditional companies still have doubts in relation to the transferring of their data outside of the limits of the corporation. The health of cloud computing systems is fundamental to the business, given the complexity of the systems it is difficult to ensure that all services and resources will work properly. In order to ensure a more appropriate management of the systems and services in the cloud, an architecture is proposed. The architecture has been modularized through specializing monitoring functions, data mining, and inference with Bayesian network. In this architecture are essential records of event monitoring systems and computing resources because the recorded data is mined to identify fault patterns a given system after the result of one or more events in the environment. For mining the monitoring data we proposed two algorithms, one for performing preprocessing of data and another to perform data transformation. As a data mining product obtained, data sets that were the input to create a Bayesian network. Through structural and parametric learning algorithms Bayesinas networks for each systems and services offered by cloud computing were created. The Bayesian network is intended to assist in decision making with prevention, prediction, error correction in systems and services, allowing to manage the health and performance of the most appropriate way systems. To check the compliance of the fault diagnosis of this architecture, we validate accuracy of inference of Bayesian network with cross-validation method using data sets generated by monitoring systems and services.

Keywords: cloud computing, IaaS, monitoring, data mining, structural learning, parametric learning, bayesian network, inference, health systems.

LISTA DE FIGURAS

LISTA DE TABELAS

ALGORITMOS, PROGRAMAS E COMANDOS

SUMÁRIO

Capítulo 1

Introdução e Motivação

A computação em nuvem (em inglês, *cloud computing*) é um dos avanços mais significativos no campo da tecnologia nos últimos anos, tendo rápida aceitação entre as empresas, incluindo as pequenas e médias (PMEs), mudando a forma como as pessoas fazem o processamento e o gerenciamento das informações, fornecendo serviços de computação a um baixo custo e com garantia de qualidade, reduzindo as despesas de capital (CAPEX) e operacionais (OPEX) (YISHAN; CHEN; JIA, 2009). Por ser um paradigma de computação distribuído em larga escala e dirigido pela economia em escala, a computação em nuvem tomou destaque por abstrair e virtualizar um pool de recursos com funções e serviços de computação dinamicamente escaláveis, permitindo a integração necessária para automatizar e conectar entre si todos os processos de uma organização, sendo entregues sob demanda para clientes externos através da Internet. As soluções em nuvem impulsionam a eficiência, reduzem custos e aceleram a adoção de tendências tecnológicas como mobilidade, análise de dados e *Big Data*.

Entretanto, um dos grandes desafios da computação em nuvem está na qualidade dos serviços entregues. Os sistemas e serviços na nuvem devem ter garantias de disponibilidade, segurança e a confidencialidade da comunicação entre o provedor e os usuários, pois dado que são sistemas distribuídos e de larga escala, falhas podem ocorrer com frequência.

As garantias fornecidas pelo provedor para seus consumidores podem ser definidas através de contratos em nível de serviço – SLAs (Service Level Agreements). A maioria dos provedores de nuvem oferece garantias de tempo de disponibilidade em seus SLAs. Um dos problemas relacionados ao SLA é a dificuldade para expressar e implementar o contrato em nível computacional, por exemplo, como fornecer garantias de disponibilidade para uma transação se esta envolve um fluxo de dados através

da Internet?

As empresas tradicionais ainda possuem dúvidas em relação à adoção da computação em nuvem. Um dos motivos é a confiabilidade dos serviços, outro fator importante está relacionado à transferência de seus dados para fora dos limites da corporação. A Amazon AWS é uma das principais plataformas de computação em nuvem, hospedando inúmeros aplicativos e sites, que sofrem com problemas de confiabilidade e indisponibilidade. Um dos casos mais recentes aconteceu em 2015, no dia trinta e um de julho, em que a AWS relatou problemas com seus serviços que estão localizados em toda a costa oeste dos Estados Unidos, ficando os servidores indisponíveis por mais de dez horas.

Gerenciar os sistemas e serviços na nuvem ainda é complexo e custoso dado o grande número de variáveis existentes em um "ecossistema" composto por sistemas operacionais, banco de dados, serviços de rede e colaboração, armazenamento de dados, sistemas de gestão empresarial e vários recursos computacionais físicos, tais como memória, CPU e interfaces de rede. A identificação dos problemas e falhas na nuvem pode estar em vários fatores que se inter-relacionam. Essas características são a motivação de estudo e pesquisa, em que ferramentas de monitoramento, metodologias e modelos, tais como, mineração de dados e redes Bayesianas são mecanismos que podem permitir um gerenciamento mais adequado para a saúde dos sistemas e serviços oferecidos nos diversos modelos de computação em nuvem.

1.1 Descrição do problema

A propriedade determinante dos ambientes corporativos é a complexidade que envolve a comunicação entre diferentes tecnologias, usuários, culturas e politicas internas. Com a Internet, a computação em nuvem possibilitou o avanço e a adoção de novas tecnologias em direção aos ambientes corporativos, ao tornar ubíquo os acessos aos sistemas e serviços. Porém, a computação em nuvem teve como consequência uma enorme implicação nos atributos de comunicação e gerenciamento, dos recursos computacionais e dos sistemas e serviços de cada organização.

Neste panorama de computação em nuvem, a falta de mecanismos para monitoramento pode resultar no aumento do risco de indisponibilidade dos sistemas e serviços, além de problemas como recursos computacionais mal utilizados e sobrecarregados, congestionamento do tráfego e problemas com segurança. A computação em nuvem

por um lado, é o facilitador na adoção de novas tecnologias, mas, por outro, pode tonar-se uma questão problemática a ser resolvida, pois com o surgimento de novas aplicações e sistemas, muitas vezes heterogêneos, o monitoramento das redes de computadores, dos serviços e sistemas pode tornar-se um verdadeiro desafio.

Dependendo do tamanho da complexidade na arquitetura das aplicações, sistemas e serviços na nuvem, tarefas consideras simples podem tornar-se bastante complexas, levando a um custo alto de gerenciamento. Desta forma, o controle de sistemas e de serviços na nuvem não pode ser realizado apenas por esforço humano.

A utilização de soluções semiautomatizadas ou automatizadas pode tonar a atividade de gerenciamento mais eficaz e adequada ao panorama dos sistemas e serviços na computação em nuvem. Tendo isso em vista, é um desafio a proposição de soluções que realizem tais atividades de gerenciamento na nuvem. Algumas pesquisas moveram-se sobre o campo de diagnósticos das falhas de sistemas e redes de computadores e demonstraram restrições, em que em Yishan, Chen e Jia (2009), foi proposto um método para inferência Bayesiana de diagnósticos das falhas e redes de computadores, mas este método não foi validado em condições reais, não ficando demonstrada a eficácia do método. Entretanto, o trabalho (LI; ZHAO; YIN, 2009), apesar de ter o método avaliado, apenas foi utilizado o conhecimento de um especialista para a modelagem do método de diagnóstico das falhas em sistemas, podendo ser um problema para sistemas e serviços na nuvem, dada a grande quantidade de variáveis que podem falhar, contribuindo para a falha no ambiente, e um especialista não seria capaz de gerar um modelo para o gerenciamento da saúde desses ambientes. Outros trabalhos, como os de Mi et al., (2011), Bermudez et al., (2013) e Bin, Zhijian e Yu (2012), exploram suas respectivas formas de detecção das falhas e dos problemas de desempenho dos sistemas na computação em nuvem, mas não apontaram como resolver tais problemas ou falhas identificados.

Identificamos na literatura que há um grande esforço para se criar soluções de diagnóstico das falhas em sistemas e serviços de rede, mas muitos problemas e restrições são encontrados pelo caminho na realização de experimentos e pesquisas, tais como a obtenção de um ambiente realístico para o experimento e a interpretação de dados desestruturados. Além desses aspectos, pesquisas que se movem no campo da computação em nuvem, devem atentar-se para problemas na arquitetura dos sistemas e serviços.

1.2 Hipótese do projeto

Partindo do pressuposto de que sistemas e serviços na computação em nuvem podem falhar e que sua falha pode causar a perda de informações e a degradação do desempenho, é importante realizar a monitoração e o controle desses processos por meio de ferramentas que auxiliam na identificação de problemas. Porém, para um ambiente computacional muito complexo, em que muitas variáveis podem indicar falhas nos sistemas, é difícil identificar e indicar a causa e os efeitos de um determinado problema.

Por isso, adotando-se um ambiente realístico de computação em nuvem, onde os sistemas e serviços são monitorados, propôs-se como hipótese para este trabalho uma arquitetura de gerenciamento da saúde dos sistemas, que tem por objetivo agrupar todas as informações geradas pelo monitoramento desses sistemas, e a partir dessas informações gerar o conhecimento necessário para tomadas de decisões, indicando a necessidade de manutenção ou de ajuste no controle de escalabilidade dos recursos computacionais, a fim de garantir o correto funcionamento dos sistemas de informação disponíveis em tempo integral na infraestrutura da nuvem.

O objetivo desta arquitetura é processar dados não-estruturados do log de monitoramento dos sistemas e serviços, gerar conhecimento com a relação das variáveis dos sistemas e seus eventos de falhas, para então, a partir deste conhecimento gerado, criar uma rede Bayesiana, a fim de auxilar no gerenciamento da saúde dos sistemas na nuvem. A rede Bayesiana será modelada com dados de monitoramento e com o conhecimento de um especialista nos sistemas. Com isso, o objetivo é que a rede Bayesiana expresse relações precisas das variáveis dos sistemas, ou seja, expresse a relação causa e efeito de um determinado problema de saúde nos sistemas, que pode ser desde uma indisponibilidade dos serviços a um problema localizado como uma falha de disco.

Essa arquitetura tem como proposta permitir, através dos dados de desempenho, usar um modelo capaz de estimar o comportamento dos sistemas, dada uma nova evidência do estado de um ou mais atributos dos sistemas e serviços. Modelar esta arquitetura será uma tarefa complexa, que requer desenvolvimento de algoritmos para processamento e estruturação dos dados, e alto grau de abstração, dada a quantidade de recursos totalmente heterogêneos disponíveis em serviços e sistemas na nuvem.

1.3 Organização do trabalho

Esta dissertação está estruturada da seguinte maneira: neste capítulo, apresentaram-se a introdução, o contexto do problema e a hipótese do projeto para este trabalho. No capítulo 2, é apresentada a fundamentação teórica do trabalho, que conceitua noções gerais sobre computação em nuvem, descrevendo seus serviços, características e funcionamento. O capítulo 2 conceitua também noções gerais sobre mineração de dados, monitoramento e probabilidade com enfoque para redes Bayesianas e suas respectivas características. No capítulo 3, é apresentada uma revisão bibliografia de pesquisas relacionadas com o tema da dissertação. No capítulo 4, é delineada a proposta do trabalho, onde é apresentada a arquitetura de gerenciamento da saúde de sistemas na nuvem. No capítulo 5, a arquitetura é experimentada em um ambiente realístico, apresentando em cada módulo algoritmos desenvolvidos e utilizados. No capítulo 6, são apresentados os resultados dos experimentos. No capitulo 7, são apresentadas as conclusões e possibilidades para futuras pesquisas decorrentes deste trabalho.

Capítulo 2

FUNDAMENTAÇÃO TEÓRICA

A literatura revela aspectos importantes sobre *computação em nuvem* e *redes Bayesianas*, tais aspectos apresentados neste capítulo, levando em consideração conceitos, características, serviços, modelos e ferramentas que poderão contribuir para a pesquisa proposta.

2.1 Computação em nuvem

Com o crescente aumento da velocidade dos processadores e das redes, e com a proliferação da Internet, modernos sistemas estão espalhando-se cada vez mais sobre o conceito de computação em nuvem. Computação em nuvem alude a forma de uso dos sistemas de informação, como na entrega de aplicações, como serviço sobre a Internet, e no uso de software em datacenters para prover esses serviços. Os softwares entregues como serviço são denominados Software as a Service (SaaS), já o hardware e o software do datacenter são denominados como "Nuvem"(FOX et al., 2009).

Essa abordagem de computação é uma evolução natural dos modelos tradicionais e oferece vários benefícios, tais como eficiência e agilidade, com destaque para os benefícios de aspecto econômico e técnico, tais como controle de custos e gerenciamento do ambiente. Economicamente, a computação em nuvem torna-se viável pela facilidade e rapidez ao prover recursos sem a necessidade de investimentos prévios, uma vez que, nesse modelo, apenas se paga pelo que é efetivamente consumido (Utility computing), ou seja, há uma mudança de "investimento"para "despesa"que gera economia fiscal. Além disso, há também a redução do custo operacional, pois toda a infraestrutura é propiciada pelo fornecedor que garante serviços confiáveis e escaláveis

ao negócio, restando mais tempo para se pensar em inovação e em novos projetos.

A tecnologia central da computação em nuvem é a virtualização (SMITH; NAIR, 2005a), e sua definição está na execução de máquinas virtuais sobre máquinas *bare-metal*[1]. As máquinas virtuais ou *virtual machines* (VMs), como também são conhecidas, são implementadas usando um sistema denominado *monitor de máquina virtual* ou apenas *hypervisor*. O *hypervisor* é executado sobre máquina *bare-metal*, e os sistemas operacionais das máquinas virtuais são administrados pelo *hypervisor*.

A computação em nuvem possui benefícios dadas as propriedades do *hypervisor* e VMs. Alguns benefícios são escalabilidade, consolidação de servidores, computação utilitária, elasticidade, gerenciabilidade, agilidade, alta disponibilidade, monitoramento e controle (TSURUOKA, 2016). Além desses benefícios, a computação em nuvem pode fornecer um conjunto de servidores *"Disaster Recovery"* e *"Data Storage"*, com a vantagem da geodistribuição, onde os dados podem ser replicados de uma região a outra em questões de minutos, sendo essas vantagens focadas na continuidade do negócio.

2.1.1 Modelos de implantação

Sabe-se que a *"Nuvem"* proporciona vários benefícios, porém dado um modelo há diferentes tipos de responsabilidades e garantias. *Nuvens* podem ser classificadas nos modelos de implantação como: *privada*, *pública*, *comunitária* e *híbrida* (MELL; GRANCE, 2011).

Na nuvem privada, o objetivo principal não é vender capacidade pela Internet, mas, sim, dar aos usuário locais uma infraestrutura ágil e flexível para suportar *workloads*[2] de serviços dentro de seu próprio domínio administrativo (SOTOMAYOR et al., 2009). Já a nuvem pública é uma alternativa a nuvem privada, com oferta de recursos em escala massiva ao público. A nuvem pública possui toda sua infraestrutura gerenciada pelo próprio provedor, com oferta de serviços na modalidade *pay-as-you-go*, ou seja, pague pelo que usar. Uma nuvem privada, no entanto, pode dar suporte a uma nuvem híbrida, através da complementação da capacidade da infraestrutura local com a capacidade computacional de uma nuvem publica (SOTOMAYOR et al., 2009). Na nuvem comunitária, a infraestrutura é para o uso exclusivo de uma comunidade de organizações, que colabora para obter um ambiente computacional de maior capa-

[1]Bare-metal nesse contexto está se referindo apenas a recursos físicos de servidores como CPU, memória e interfaces de rede.

[2]Uma quantidade de dados esperada para realização de processamento.

cidade.

2.1.2 Modelos de serviços

Na definição de NIST, computação em nuvem é um modelo para habilitar o acesso por rede ubíquo, conveniente e sob demanda de um conjunto compartilhado de recursos de computação (como redes, servidores, armazenamento, aplicações e serviços) que possam ser rapidamente provisionados e liberados com o mínimo de esforço de gerenciamento ou interação com o provedor de serviços (MELL; GRANCE, 2011).

Provedores de nuvem pública oferecem recursos de tecnologia da informação como serviços para usuários, como corporações e desenvolvedores. Em nuvens públicas, os serviços que são entregues podem ser divididos em três modelos, tais como Software como Serviço ou SaaS (Software as a Service), Plataforma como Serviço ou PaaS (Platform as a Service), Infraestrutura como Serviço ou IaaS (Infrastructure as a Service). A Figura 2.1 ilustra quais serviços são entregues em cada modelo.

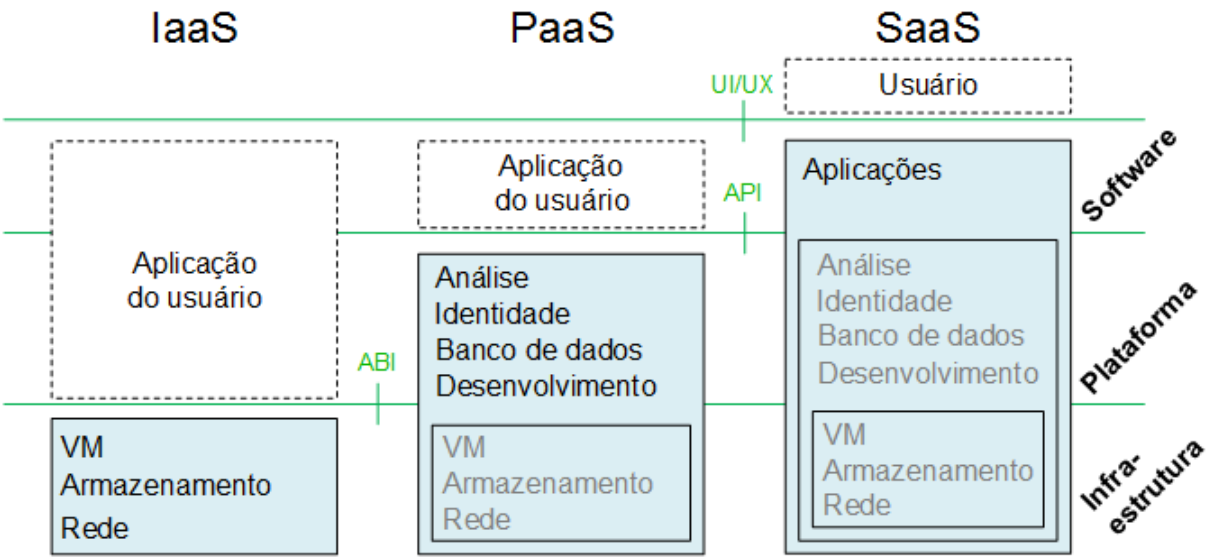


Figura 2.1: Modelos de serviços em nuvem pública

No modelo IaaS, tipicamente são fornecidas máquina virtuais (VMs) para os usuários finais, e eles não precisam realizar a manutenibilidade do hardware, porém é de sua responsabilidade a administração dos sistemas operacionais e aplicações. Os recursos computacionais oferecidos nesse modelo de serviço estão dispostos em centros de dados (*datacenters*). Esses recursos computacionais, tais como servidores, armazenamento e dispositivos de rede, estão interligados e são transparentemente gerenciados por meio da virtualização, permitindo que suas capacidades sejam compartilhadas en-

tre instâncias virtuais (BUYYA; VECCHIOLA; SELVI, 2013). Nesse modelo, as aplicações dos usuários fazem uso da *Interface Binária de Aplicação* (ABI - Application Binary Inteface) para obter acesso ao hardware (recursos computacionais disponíveis) por meio da invocação indireta de serviços do sistema operacional (SMITH; NAIR, 2005b). No modelo IaaS, estão entre os principais provedores de serviço o Amazon, Rackspace e CloudSigma (IaaS..., 2016).

Outra forma de prover soluções na nuvem é com o uso de PaaS, uma plataforma que permite construir aplicações, processar, analisar, armazenar e manipular dados. Exemplos incluem banco de dados, gerenciamento de identidade e plataformas de análise de dados. Os usuários interagem com esses serviços por meio de APIs[3]. Os usuário de PaaS não gerenciam ou controlam a infraestrutura da nuvem, mas devem construir e manter suas próprias aplicações. Alguns fornecedores que agregam serviço PaaS na nuvem são Microsoft Azure, Google App Engine e Salesforce (PAAS..., 2016).

No modelo SaaS, o cliente não gerencia ou controla a infraestrutura da nuvem nem mesmo a camada de plataforma, como é demonstrado na Figura 2.1. Dessa forma, resta ao usuário apenas uma camada de interação, a interface, UI (User Interface), que pode levar o usuário a ter várias experiências, UX (User experience) com a tecnologia. Exemplos mais comuns de aplicações SaaS vão desde um Webmail a redes sociais, tais como Gmail e Facebook.

2.1.3 Infraestrutura como Serviço

Nos últimos anos, testemunhou-se o crescimento nos serviços baseado na nuvem, com vários tipos de abstrações. IaaS tem como objetivo abstrair a complexidade computacional da infraestrutura física, oferecendo hardware e software associados como serviço. Tudo isso em *data centers* remotos, permitindo aos clientes reduzirem custos com o gerenciamento de hardware. IaaS é a evolução do tradicional serviço de *Hosting*[4], em que a contratação do serviço não requer nenhum compromisso a longo prazo e permite um modelo de provisionamento sob demanda, em que uma organização não precisa preocupar-se com questões operacionais – da mesma forma que a rede elétrica permite suprir a necessidade energética, e sequer há necessidade de conhecer seus de-

[3] API (Application Programming Interface) que significa em tradução para o português "Interface de Programação de Aplicativos", é um conjunto de rotinas e padrões de programação para acesso a um aplicativo de software ou plataforma baseado na Web.

[4] O termo hosting serve para denominar servidores físicos que suportam as atividades de outros computadores da empresa.

talhes técnicos. A preocupação do cliente com a nuvem restringe-se à definição dos requisitos e utilização dos recursos, pois todos os detalhes técnicos do *back-end*[5] da nuvem são encapsulados. Além disso, IaaS oferece serviços sob demanda, na modalidade pague pelo que usar. Outras características e componentes de uma IaaS incluem: elasticidade, flexibilidade, escalabilidade dinâmica, virtualização de desktop, conectividade com a Internet, serviços baseados em políticas, automatização de tarefas administrativas.

A nuvem IaaS pode ser implementada em diferentes modelos, como nuvem pública, privada e híbrida (BUYYA; VECCHIOLA; SELVI, 2013). Para se utilizar a nuvem pública, basta ter um cartão de crédito para realizar o cadastro no serviço, pois todo o serviço utilizado será faturado no cartão cadastrado. Neste modelo, o tradicional são grandes corporações, tais como Amazon, Google e Microsoft, oferecerem serviços de computação pela Internet. Por outro lado, a nuvem privada geralmente fica alocada na própria organização, possui infraestrutura própria e não tem dependência de terceiros. Neste modelo, estão disponíveis diversas ferramentas de código aberto, entre as quais o VMware, Eucalyptus, Nimbus, OpenNebula e OpenStack (VOGEL; GRIEBLER; ROVEDA, 2015); (SEMPOLINSKI; THAIN, 2010); (SAHASRABUDHE; SONAWANI, 2014). Já a nuvem híbrida é formada a partir de uma mesclagem entre os modelos privado e público.

Com essa oferta flexível no serviço, arquitetos de infraestrutura optam por IaaS, dado que toda a infraestrutura do datacenter é oferecida como uma abordagem de *outsourcing*, ou seja, se uma aplicação pode ser virtualizada, então ela pode ser carregada e executada em um ambiente IaaS. Na definição de NIST, IaaS são recursos computacionais (infraestrutura física) oferecidos ao contratante. Esses recursos, em geral, consistem em máquinas virtuais que oferecem armazenamento, processamento e rede para que o contratante possa instalar e executar os softwares, que vão desde o sistema operacional até variedades de aplicações (MELL; GRANCE, 2011).

2.2 Monitoramento

A arquitetura de computação em nuvem permite que os sistemas utilizem o máximo de recursos computacionais distribuídos, a fim de realizar a entrega de bens e serviços com alto nível de qualidade. A viabilização da continuidade do negócio com

[5]Back-End, nesse contexto, refere-se a toda infraestrutura física que dá suporte à entrega do serviço.

redução de custos é possível com maior controle na gestão dos sistemas e recursos computacionais.

Com esse viés computacional, fornecedores de TI (Tecnologia da Informação) agem como provedores de serviços e formalizam seu relacionamento com o cliente por meio de um instrumento denominado Acordo de Nível de Serviço (Service Level Agreement - SLA). O SLA possui uma descrição formal e explícita dos produtos (bens, serviços e sistemas contratados) e índices a serem atingidos para o cumprimento dos compromissos acordados.

O uso de SLA é uma prática muito difundida no mercado, onde áreas da empresa funcionam como unidades de negócio. Isto significa que cada área é responsável por gerir seu próprio orçamento, provendo bens e serviços para os clientes internos. Sem o SLA, o fornecedor não tem clareza do escopo para o qual foi contratado, e o cliente pode correr o risco de receber bens ou serviços em desacordo com suas expectativas. Alguns dos serviços de TI tratados com clientes no SLA são: agilidade de resposta no atendimento, segurança da informação, disponibilidade máxima e qualidade dos serviços. Porém a garantia de qualidade e desempenho dos serviços ainda dependem dos analistas responsáveis de gerenciar a usabilidade e a estabilidade dos principais componentes da nuvem.

Como descrito acima, para a garantia da qualidade dos sistemas computacionais na nuvem, também é necessário o uso dos mecanismos de monitoramento e gerência, que permitam detectar problemas de congestionamento de tráfego, recursos mal utilizados e sobrecarregados, bem como falhas de sistemas e problemas de segurança.

2.2.1 Sistemas de monitoramento

Hoje, com o aumento do grau de complexidade e do número de sistemas e das redes que os suportam, faz-se necessário o emprego de sistemas de gerenciamento e monitoramento que proporcionem qualidade de serviço, proatividade, integração com processo de serviços e negócios. Esses sistemas de monitoramento são também conhecidos como ferramentas DCIM (Data Center Infrastructure Management) e permitem registrar e analisar informações de status relacionados a servidores e sistemas (HARRIS; GENG, 2015). Existem várias ferramentas DCIM de uso comercial, código aberto e licenças de software livre. Algumas soluções suportam o uso e o desenvolvimento de extensões ou plugins, a fim de adicionar novas funcionalidades. Algumas das ferra-

mentas e seus recursos serão apresentados nos próximos parágrafos.

Nagios é uma das ferramentas de monitoramento mais usadas (KATSAROS; KUBERT; GALLIZO, 2011). Este sistema de monitoramento foi desenvolvido para suportar escalabilidade e flexibilidade, fornecendo informação sobre toda a infraestrutura de Tecnologia da Informação, permitindo a detecção e a notificação de falhas, bem como a gerência dos sistemas e dispositivos de rede. Novas funcionalidades podem ser adicionadas à ferramenta através do desenvolvimento de extensões ou plugins. Nagios foi projetado para trabalhar em sistemas operacionais linux ou unix, e seu código é aberto. Com o Nagios é possível monitorar vários tipos de dispositivos e sistemas, reportando problemas e fornecendo relatório de acordo com os resultados.

Zabbix é uma ferramenta de monitoramento que oferece como *front-end* uma interface web com diferentes visões e mapeamentos (LEMES, 2014). É utilizado um banco de dados MySQL para armazenar informações históricas das configurações. O *back-end* núcleo da ferramenta é desenvolvido na linguagem C, enquanto o *front-end* a interface web é desenvolvido em PHP. Alguns dos protocolos suportados para o monitoramento são TCP, ICMP e SNMP.

Cacti é uma ferramenta que possui como característica facilidade de implementação e usabilidade simplificada, e destaca-se pelos recursos gráficos inerentes à desempenho de um dado dispositivo conectado à rede (KUNDU; LAVLU, 2009). O Cacti é constituído de uma interface web desenvolvida na linguagem PHP. Essa interface dá acesso à base de dados MySQL, que também é gerida pelo Cacti. As informações são inseridas na base de dados através da ferramenta RRDTool (*Round Robin Database Tool*), que armazena e apresenta dados que se alteram com o passar do tempo, como é o caso da largura de banda consumida por uma interface Ethernet. A utilização do Cacti não possui custo por se tratar de um software livre.

2.2.2 Monitoramento com o Nagios

Nagios permite monitorar a saúde de máquinas, redes e serviços de sistemas (SCHUBERT et al., 2008). O início de seu desenvolvimento ocorreu no ano de 1999. O nome original do projeto foi *Netsaint*. O término do projeto ocorreu no ano de 2002, mas teve a continuação com o novo nome *Nagios*. O autor é *Mr. Ethan Galstad*, que atualmente também é presidente da companhia *Nagios Enterprises* (GALSTAD, 2009).

Nagios é um sistema de monitoramento muito popular. Este fato foi confirmado

por NATARAJAN, que lançou uma discussão em um fórum para fãs de distribuição Linux. O resultado da discussão mostrou, por maioria de votos, que o Nagios é a ferramenta de monitoramento mais popular, como pode ser visto na Figura 2.2.

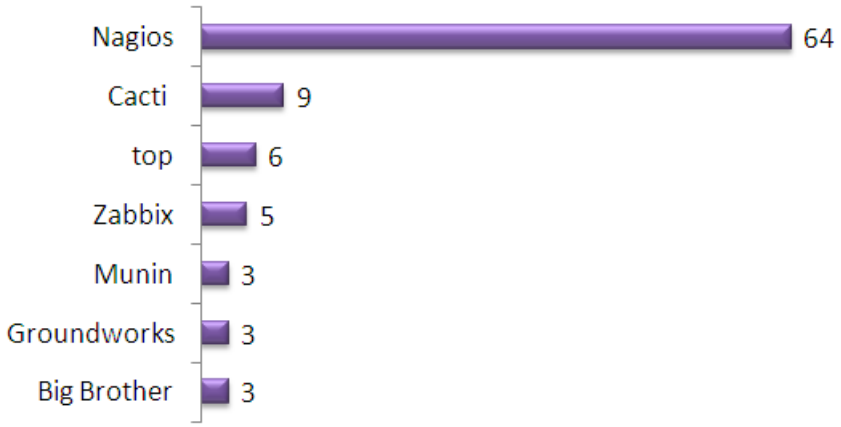


Figura 2.2: Votação para sistemas de monitoramentos favoritos (NATARAJAN, 2009).

Com o Nagios, vários tipos de dispositivos e sistemas podem ser monitorados tais como consumo de memória, espaço em disco, carga da CPU, número de processos do sistema e muitas outras informações customizadas (BARTH, 2006). O Nagios fornece uma interface web de fácil interação, permitindo navegar entre *hosts* e *serviços*, exibindo detalhes das informações de monitoramento, mas de acordo com BARTH, a grande força do Nagios, comparado a outras ferramentas, está em sua estrutura modular.

```
.
|-- cgi.cfg
|-- htpasswd.users
|-- nagios.cfg
|-- objects
|   |-- commands.cfg
|   |-- contactgroups.cfg
|   |-- contacts.cfg
|   |-- hostextinfo.cfg
|   |-- hostgroups.cfg
|   |-- hosts.cfg
|   |-- hosttemplates.cfg
|   |-- services.cfg
|   |-- servicetemplates.cfg
|   `-- timeperiods.cfg
`-- resource.cfg

1 directory, 14 files
```

Figura 2.3: Diretório ../nagios/etc/ com os principais arquivos de configuração.

O *Nagios daemon* é o principal processo que compõe a estrutura do Nagios. Após a inicialização do *daemon*, os arquivos de configuração da Figura 2.3 são carregados,

e o processo de monitoramento é iniciado. O principal arquivo de configuração é o *nagios.cfg*, a partir do qual, por meio de parametrizações, são feitas as relações com os demais arquivos de configurações existentes. A comunicação do *Nagios daemon* com as variáveis de ambiente do sistema é implementada através de arquivos-texto, onde as saídas do sistema são armazenadas em arquivos (e.g. arquivos de log), bem como as entradas do sistema são lidas nos arquivos (e.g. arquivos de configuração).

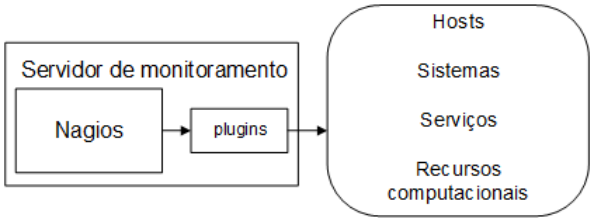


Figura 2.4: Monitoramento baseado em *plugins*.

O Nagios sozinho não é capaz de realizar a checagem da saúde dos *hosts* e *serviços*, nem mesmo notificar quando houver modificações de status (SCHUBERT et al., 2008). Esse controle de checagem dos status da saúde dos *hosts* e *serviços* dos sistemas é realizado por programas externos, que também são conhecidos por *plugin*. Os *plugin* são incorporados ao Nagios e estão localizados entre o Nagios e os *hosts* e *serviços* monitorados, como pode ser visto na Figura 2.4. Os plugins não são distribuídos juntamente com o Nagios, mas um conjunto de plugins está disponível para download e pode ser encontrado no seguinte site *http://www.nagios.org/download*. Esses *plugin* permitem monitorar hosts, serviços, dispositivos, recursos computacionais, protocolos e sistemas com o Nagios. Por exemplo, nesse conjunto de *plugin*, existe um *plugin* para testar conexões TCP, chamado *check_tcp*, que pode ser usado para determinar se um serviço que utiliza protocolo TCP pode ser alcançado ou não pela rede.

Os plugins são responsáveis pela realização de testes e pelo processamento da saída correspondente ao retorno dos testes realizados em hosts, serviços e recursos computacionais de sistemas monitorados. Para testes realizados em *hosts*, o teste pode ser descrito em três estados: OK, DOWN ou UNREACHABLE; e para testes realizados em *serviços*, o teste pode ser descrito em quatro estados: OK, WARNING, CRITICAL ou UNKNOWN. A Tabela 2.1 contém a relação dos códigos de retorno dos plugins e seus respectivos estados para *hosts* e *serviços*.

O plugin pode ser criado por uma variedade de linguagens de programação, mas é importante seguir todos os requisitos do Nagios, para um adequado desenvolvimento. Com esse propósito, um guia de referência para desenvolvedores está disponível em

Tabela 2.1: Código de retorno dos plugins, dado um estado do host ou serviço

Código de Retorno	Estado do Serviço	Estado do Host
0	OK	UP
1	WARNING	UP ou DOWN
2	CRITICAL	DOWN/UNREACHABLE
3	UNKNOWN	DOWN/UNREACHABLE

https://nagios-plugins.org. Junto a esse guia, alguns trabalhos mostram que a criação de um plugin pode ser uma tarefa extremamente simples.

Fernández et al. (2014) demonstraram pelo fluxograma da Figura 2.5 a típica execução de um plugin do Nagios, onde na fase inicial (Start), se necessário, são definidos limiares para os estados WARNING e CRITICAL, por passagem de parâmetros ou assumindo os limites-padrão definidos no *plugin*. Na segunda fase (Meassurement), são aferidas medições e, se a checagem falhar, o *plugin* termina sua execução com um código de saída "3". Porém, se a checagem do *plugin* não falhar, o resultado é então verificado em relação às faixas WARNING e CRITICAL, e, dependendo do resultado, o plugin imprime a mensagem adequada e termina com o código de saída correspondente.

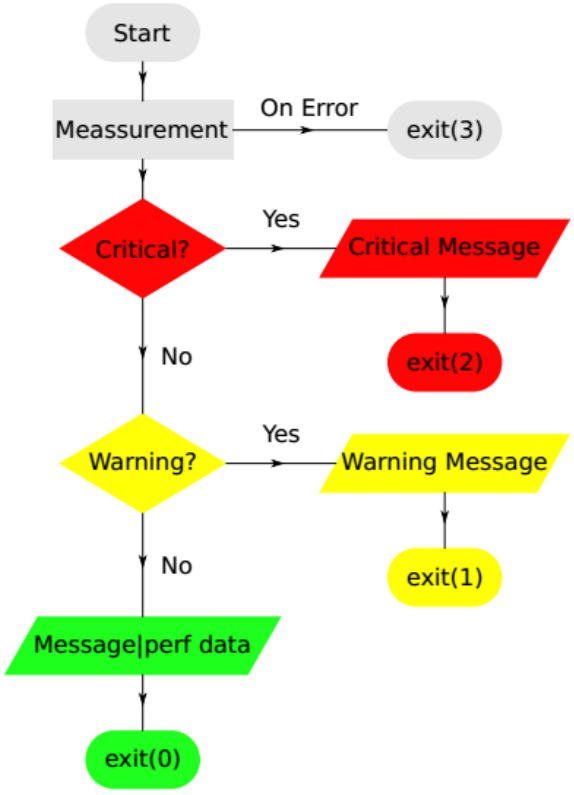


Figura 2.5: Fluxograma básico de um *plugin* genérico do Nagios (FERNÁNDEZ et al., 2014)

Um *plugin* Nagios pode ser um simples programa *bash script*, que deve seguir duas regras básicas de um plugin Nagios. Primeira regra: o plugin deve retornar o estado do recurso checado. Segunda regra: deve ser feito o uso do hífen "-" para separar o estado do recurso monitorado de seu detalhe textual, que possui dados explicativos. No Programa 2.1, é apresentado o código fonte de um *plugin* Nagios desenvolvido em *bash script*, com a finalidade de aferir o espaço em disco disponível, e assim imprimir a saída no formato-padrão do Nagios, seguindo as regras do guia de referência para desenvolvedores.

Programa 2.1: Código de um plugin para o Nagios

```
1  #!/bin/bash
2  espaco_usado=‘df -h / | grep -v Use% | awk ’{print $5}’ | sed ’s/%//g’‘
3  case $espaco_usado in
4  [1-84]*)
5  echo ”OK - $espaco_usado% de espaco em disco usado.”
6  exit 0
7  ;;
8  [85]*)
9  echo ”WARNING - $espaco_usado% de espaco em disco usado.”
10 exit 1
11 ;;
12 [86-100]*)
13 echo ”CRITICAL - $espaco_usado% de espaco em disco usado.”
14 exit 2
15 ;;
16 *)
17 echo ”UNKNOWN - $espaco_usado% de espaco em disco usado.”
18 exit 3
19 ;;
20 esac
```

Comando 2.2: Plugin para checar espaço em uso no disco

```
1  # ./checa_uso_disco
2  OK - 39% de espaco em disco usado.
```

O Comando 2.2 demonstra a execução do *plugin checa_uso_disco*, sendo seu respectivo código o Algoritmo 2.1. A execução do *plugin* retornou "OK", para o consumo de disco do servidor monitorado, sendo a saída "39% de espaco em disco usado" o

detalhe textual do recurso monitorado.

Quando o Nagios agenda a execução de um plugin, o plugin pode retornar dois tipos de dados na saída-padrão. Ambos os campos estão na mesma linha. Esses dois campos são separados pelo operador pipe (|). Tudo o que estiver antes do operador pipe, o Nagios considera *texto de estado*, e é inserido no campo de estado do Nagios, que vai para interface de usuário. Já tudo o que vier depois do pipe, é formatado como dados de desempenho. O Nagios processa as saídas dos plugins usando o seguinte formato: <ESTADO DO SERVIÇO> - <Detalhe textual do serviço> | <Dados de desempenho>.

Abaixo, segue uma saída típica para plugin com dados de desempenho:

OK - load average: 0.35, 0.29, 0.20 | load1=0.350;5.000;10.000;0;load5=0.290;4.000;6.000;0; load15=0.200;3.000;4.000;0;

Para que a saída-padrão dos plugins seja arquivada, o Nagios possui macros[6] que realizam o respectivo processamento e armazenamento dos eventos em arquivos de log. De modo geral, a fim de registrar todas as saídas dos plugins e os eventos do Nagios, a diretiva *log_files* do arquivo *nagios.cfg*, indica onde o Nagios deve registrar suas informações; neste caso, é especificado o caminho do arquivo *nagios.log*, responsável por armazenar todas as saídas e os eventos do Nagios.

O arquivo *nagios.log* registra todos os eventos de monitoramento do Nagios, incluindo checagens de notificações, comandos externos e eventos. Isso faz com que o arquivo cresça rapidamente, dificultando a rastreabilidade de eventos no arquivo de log. Para dividir os eventos registrados no arquivo *nagios.log*, em arquivos que representem os eventos, dado um mês, uma semana, um dia ou uma hora de monitoramento, para que assim seja facilitada a rastreabilidade do arquivo de log, é preciso habilitar a "rotação do log", com o uso da diretiva *log_rotation_method*, e manter os arquivos de log armazenados no caminho indicado pela diretiva *log_archive_path*. As duas diretivas são especificadas no arquivo de configuração *nagios.cfg*.

2.2.3 Monitoramento de sistemas

Em um ambiente de *cloud computing*, os sistemas estão expostos a várias condições de hardware e software, que podem impor ou predeterminar uma gravidade na saúde

[6]Macros são definidos na programação como um padrão de entrada que é substituído por um novo padrão de saída.

dos sistemas. Para oferecer diagnóstico da saúde de alguns sistemas como SAP ERP, Linux/Unix e Windows, várias formas de monitoramento são apresentadas nesta seção.

Para o sistema integrado de gestão empresarial SAP ERP[7], existem várias formas de monitoramento. Uma forma simples é controlar o estado das portas de comunicação que correspondem aos serviços SAP em execução, verificando se elas estão abertas e acessíveis. Normalmente, estas portas são a TCP 3200, responsável pela comunicação do SAP GUI (Interface gráfica do usuário) com o SAP ERP, e a porta TCP 3300 usada para comunicação RFC[8] entre os ambientes do sistema SAP ERP. Para essas portas TCP, uma simples checagem pode ser feita com o plugin *check_tcp*. No entanto, caso algum serviço interno do SAP falhe, pode ser que o usuário não consiga realizar a autenticação e o acesso ao sistema, até mesmo se as portas de comunicação TCP estiverem acessíveis. Neste caso, é necessário testar as interações complexas entre os componentes SAP, e isso exige uma comunicação a nível da camada de aplicação do sistema SAP ERP (BARTH, 2006).

Para o monitoramento de servidores Linux e Unix, também existem várias formas de monitoramento. Uma das possíveis abordagens é com o uso do *plugin check_by_ssh*. Como pode ser visto na Figura 2.6, o uso desse *plugin* faz sentido, o daemon SSH (sshd) já está em execução na maioria dos servidores que precisam ser monitorados. Nessa abordagem, o Nagios dever ser capaz de se conectar ao servidor remoto sem o uso de usuário e senha, para isso deve ser configurado um método de chave pública para autenticação por meio do protocolo SSH. Assim, os *plugins check_disk*, *check_load* e *check_swap*, como apresentado na Figura 2.6, são ativados no servidor remoto, retornando o estado do recurso monitorado para o Nagios.

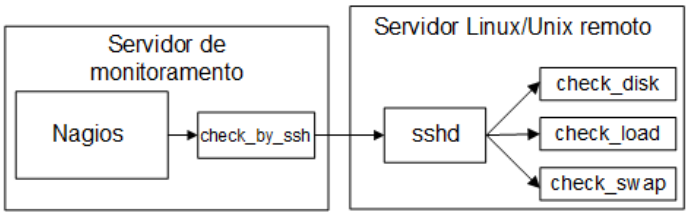


Figura 2.6: Monitoramento de sistemas remotos com o Nagios via SSH

Em sistemas Microsoft Windows, é relativamente simples implementar o monitoramento de recursos, serviços e sistemas. Uma das possibilidades é com o uso do

[7]SAP ERP (até 2003 SAP/R3, até 2007 mySAP ERP) é um sistema integrado de gestão empresarial (ERP) transacional, produto principal da SAP AG, uma empresa alemã, líder no segmento de softwares corporativos, tendo cerca de 86 mil clientes, segundo a própria SAP.

[8]RFC (Remote Function Call) é uma interface padrão do SAP para realizar a comunicação entre sistemas SAP.

agente NSClient++, simples e seguro, sendo desenvolvido para trabalhar especificamente com servidores Microsoft Windows. Este agente funciona como um proxy, intermediando a comunicação entre o plugin Nagios e os serviços privados de um servidor Microsoft Windows, como apresentado na Figura 2.7. Serviços privados, tais como memória disponível, espaço em disco ou carga de CPU, não podem ser monitorados diretamente pelo Nagios. Outros serviços públicos, tais como HTTP, FTP, SMTP e POP3, podem ser monitorados diretamente através de seus respectivos *plugins* Nagios *check_http, check_ftp, check_smtp, check_pop*. O NSClient++ permite o uso de vários protocolos, sendo um deles o NRPE (Nagios Remote plugin Executor), um protocolo Nagios para coletar métricas de *hosts* remotos através de checagem ativa. O diagrama de bloco da Figura 2.7 demonstra que requisições do *plugin* Nagios *check_npre* são primeiramente feitas ao agente NSClient++, em seguida o NSClient++ recupera as métricas e coloca-as em uma pilha interna para que, subsequentemente, sejam fornecidas ao *plugin* Nagios, para realizar seu processamento.

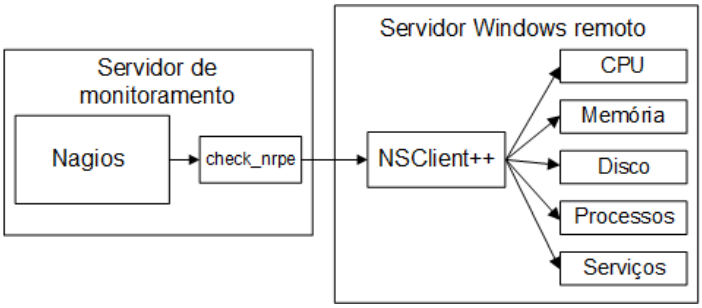


Figura 2.7: Monitoramento de sistemas remotos com o Nagios via NSClient++

2.3 Mineração de Dados

A necessidade de solucionar problemas automatizando a análise de dados com o objetivo de transforma-los em conhecimento, tem crescido sob o rótulo de Mineração de Dados, mais conhecido na comunidade científica como Descoberta de Conhecimento em Banco de Dados (KDD - Knowledge Discovery in Databases). KDD não é uma nova técnica, mas, sim, uma área de pesquisa interdisciplinar, que compõe resultados de pesquisas de diversas comunidades cientificas, tais como tecnologia de banco de dados, estatística, aprendizado de máquina e visualização de dados. Mineração de Dados é a principal etapa no processo KDD, responsável pela seleção dos métodos a serem utilizados a fim de localizar padrões nos dados, pois segundo Fayyad (1996), mineração de dados é o "processo de descoberta de padrões válidos, novos, potenci-

almente úteis e compreensíveis, embutidos nos dados". Segundo Fayyad (1996), KDD pode ser divido em etapas, conforme apresentado na Figura 2.8.

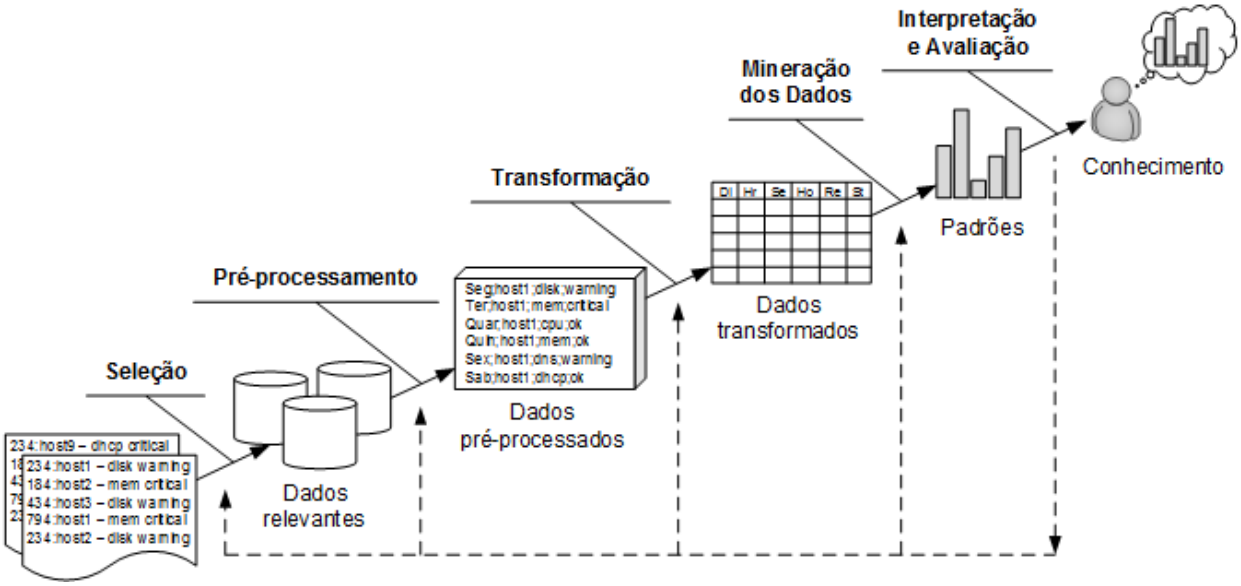


Figura 2.8: Etapas que compõem o processo KDD conceitual de Fayyad.

O processo KDD é interativo e iterativo, envolvendo inúmeros passos com decisões do analista. Nos últimos vinte anos, pesquisadores têm focado seus esforços a fim de melhorar o designer do processo, propondo um modelo simplificado ou genérico unificando, todos os modelos existentes. A pesquisa feita por Kurgan e Musilek (2006) realiza a comparação de modelos KDD difundidos na indústria e no meio acadêmico frente a um *modelo genérico* e mais simplificado. Pode-se observar na Tabela 2.2 a comparação de alguns desses modelos e suas etapas.

O *modelo genérico* é proposto dada uma série de observações nos cinco modelos analisados em Kurgan e Musilek (2006). O *modelo genérico* oferece uma visão consolidada dos modelos analisados. Aqui, resumidamente, estão delineadas algumas das etapas básicas deste *modelo genérico*:

Na primeira etapa, é desenvolvida uma compreensão do domínio da aplicação e do conhecimento prévio relevante do assunto tratado nesse processo de extração do conhecimento.

Na segunda etapa, é selecionado um conjunto de dados, ou de amostra dos dados relevantes do processo de extração do conhecimento. O objetivo é realizar a identificação de problemas na qualidade dos dados.

Na terceira etapa, ocorrem a limpeza, o pré-processamento e a transformação dos dados. Operações básicas eliminam dados redundantes, inconsistentes, e avaliam

Tabela 2.2: Comparação das etapas de modelos KDD (KURGAN; MUSILEK, 2006)

Modelo	Fayyad et al.	Cabena et al.	Modelo Genérico
Área	Acadêmica	Industrial	N/A
No de etapas	9	5	6
Referencia	(FAYYAD, 1996)	(CABENA et al., 1998)	N/A
Etapas	1º Desenvolvimento e entendimento do domínio da aplicação	1º Determinar os objetivos do negócio	1º Entendimento do domínio da aplicação
	2º Criar um conjunto de dados	2º Preparação dos dados	2º Entendimento dos dados
	3º Limpeza dos dados e pré-processamento		3º Preparação dos dados
	4º Redução dos dados e projeção		
	5º Escolher a tarefa de Mineração de Dados		
	6º Escolher o algoritmo de Mineração de Dados		
	7º Mineração de Dados	3º Mineração de Dados	4º Mineração de Dados
	8º Interpretar padrões minerados	4º Elicitar conhecimento do domínio	5º Avaliação
	9º Consolidação do conhecimento descoberto	5º Assimilação do conhecimento	6º Consolidação e implantação do conhecimento

possíveis valores discrepantes ao conjunto de dados, removendo ruídos e adicionando novos atributos, se for apropriado. Deste modo, o conjunto de dados estará organizado segundo as requisições e particularidades da técnica que se aplicará na etapa seguinte.

A quarta etapa consiste na aplicação do método de mineração selecionado, em que os dados são de fato analisados a fim de encontrar padrões consistentes que estabeleçam relações de dependência. Nessa etapa, também são aplicados testes no conhecimento gerado.

Na quinta etapa, é realizada a interpretação dos resultados do conhecimento descoberto, avaliando possível retorno no processo a fim de identificar quais ações alternativas podem ser tomadas para melhorar o resultado.

Na sexta e última etapa, ocorre a implantação da descoberta do conhecimento, que

pode se usado diretamente ou incorporado em outros sistemas, ou até simplesmente documentado e reportado aos interessados. Um importante aspecto nesse processo de descoberta do conhecimento é relativo ao tempo gasto para completar cada etapa. A pesquisa realizada por Kurgan e Musilek (2006) mostrou que Cabena et al. (1998) estimam que 20% de esforço são gastos com a Determinação dos Objetivos do Negócio, 60% são gastos com a Preparação dos Dados, e 10% com a Mineração de Dados (CABENA et al., 1998). Baseado em experiências de aplicações industriais, Shearer (2000) estima que cerca de 50-70% do tempo são gastos com a Preparação dos Dados, 20-30% são gastos com o entendimento dos dados, 10-20% na modelagem (SHEARER, 2000). Cios et al. (2005) estimam que 10-20% do esforço são gastos com o Entendimento do Domínio e dos dados, 30-60% são gastos com a Preparação dos Dados, 10-25% na Mineração dos Dados e 5-10% na Avaliação e Uso do Conhecimento Descoberto (CIOS et al., 2000). Como apresentado pelos estudos citados, exitem várias razões pelas quais a Preparação dos Dados que inclui o pré-processamento demanda tanto tempo. Grandes companhias, ao realizarem o processo, acabam não coletando todos os dados necessários, e os dados coletados frequentemente contêm erros e até mesmo dados redundantes e inconsistentes.

Enquanto a etapa que mais demanda esforço é a de pré-processamento, a etapa mais importante do processo KDD é a de Mineração dos Dados, tal é que rotula a ciência como Mineração de Dados. Porém, dependendo do tipo de dado que está sendo minerado, o processo pode ter outras formas de denominação. Alguns termos relacionados são: *mineração de texto*, *análise de texto*, *mineração de conteúdo*, *mineração de áudio*, *mineração de imagens*, *mineração de metadados* e *mineração de vídeo*.

Quando é conduzida uma pesquisa que envolve usuários, dispositivos de redes e sistemas computacionais, a seleção do método adequado é essencial para obtenção de resultados eficientes. A coleta de dados também envolve a escolha de métodos. Logs transacionais e a análise de logs transacionais são uma abordagem para coleta de dados e um método para análise do desempenho de sistemas e do comportamento dos usuários, que vem sendo utilizado desde 1967 (MEISTER; SULLIVAN, 1967). Logs transacionais ou apenas arquivos de logs são normalmente armazenados em arquivos ASCII ou textos simples, capazes de armazenar as interações que ocorrem entre sistemas, dispositivos de rede e seus usuários.

Existem vários tipos de logs, tais como logs de servidores, logs de firewall, logs de sistemas e logs de aplicação, em que cada um desses logs são gravados vários eventos

dentre uma cadeia de ações.

Mesmo que os logs tenham em comum gravar algumas ações, eles registram diferentes ações. Cada tipo de log segue seu próprio formato que o torna difícil de analisar. Para os analistas que não são treinados, os registros são caóticos. Para analistas treinados, os registros são extremamente difíceis de serem analisados. Dado que arquivos de logs são usualmente grandes, é extremamente difícil a manipulação manual, a fim de encontrar e recuperar informações relevantes para análise.

Existe uma similaridade do que foi revelado sobre *Mineração de Dados* e *Análise de Logs*. Pois, dada uma grande quantidade de logs, existe a necessidade de entender o que os sistema computacionais tentam dizer com estes dados. Sabendo disso, pode-se então chamar de *"Mineração de Logs"* a aplicação das técnicas de Mineração de Dados para análises avançadas de logs.

Os logs podem apresentar características de dados estruturados e desestruturados. Dados estruturados são dados que podem ser facilmente divididos em tabelas e colunas. Um arquivo de log, por si só, é um dado estruturado, mas quando se lida com um número de diferentes logs juntos, eles não são.

Muitos logs, nos dias de hoje, são mais similares a dados desestruturados. Nesse caso, as técnicas do modelo de *Mineração de Texto* (RUIZ, 2016) podem ser úteis. Entretanto, logs de monitoramento possuem características de dados estruturados, e, nesse caso, a utilização das técnicas do modelo de *Mineração de Dados* é a mais indicada para descoberta de conhecimento.

2.4 Redes Bayesianas

O termo "Bayesiano" é derivado de "Bayes", em homenagem a Thomas Bayes, que foi um reverendo presbiteriano e matemático que viveu no início do século XVIII (1702 – 1761) na Inglaterra, e suas contribuições culturais e intelectuais podem ser encontradas, hoje, em milhares de artigos científicos (PENA, 2006).

Existem duas grandes vertentes na estatística: uma frequentista e outra Bayesiana. A vertente Bayesiana aborda aspectos de probabilidade como grau de crença. O primeiro trabalho de destaque da abordagem Bayesiana a problemas de inferência foi publicado por Richard Price, em 1763, da obra póstuma de Thomas Bayes intitulada "An essay towards solving a problem in the doctrine of chances" (Ensaio buscando resolver

um problema na doutrina das probabilidades), onde estava presente a demonstração do famoso Teorema de Bayes.

2.4.1 O teorema de Bayes

Para chegar ao Teorema de Bayes, parte-se de princípios básicos da probabilidade e seus axiomas, e mostra a relação entre uma probabilidade condicional e sua inversa. Assim, a probabilidade de se observar simultaneamente um evento X e um evento Y é dada pela equação (2.1), que é a regra do produto de probabilidade, deduzida pela definição de probabilidade condicional.

$$P(X,Y) = P(X|Y)P(Y) \tag{2.1}$$

A mesma probabilidade é válida também na seguinte equação:

$$P(Y,X) = P(Y|X)P(X) \tag{2.2}$$

Combinando a equação 2.1 e 2.2, pode-se obter a seguinte igualdade: P(X|Y)P(Y) = P(Y|X)P(X). Dividindo os dois lados dessa igualdade por P(X), obtém-se:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} \tag{2.3}$$

em que: P(Y) é a probabilidade *a priori* do evento Y; P(X|Y) é a verossimilhança relativa dada a evidência X e dada a hipótese do evento Y; P(X) é um fator de normalização (probabilidade *a priori* do evento X); P(Y|X) é a probabilidade *a posteriori* do evento Y conhecida a evidência X.

Inicialmente formulada pelo reverendo Thomas Bayes, a equação 2.3, conhecida como Teorema de Bayes, mostra a relação entre uma probabilidade condicional e sua inversa. Nesta equação, a probabilidade de uma hipótese é dada pela observação de uma evidência, e a probabilidade da evidência é dada pela hipótese. O Teorema de Bayes é uma equação de grande importância, pois permite o cálculo ou a atualização de probabilidades condicionais como uma função de probabilidades *a priori*.

2.4.2 Definição de Redes Bayesianas

Uma rede *Bayesiana* (RB) é um grafo direcionado acíclico (DAG), onde cada nó representa uma variável aleatória com sua respectiva TPC (Tabela de Probabilidade Condicional), ou CPT do inglês *Condicional Probability Table*, e os arcos direcionados representam relacionamentos causais diretos entre os nós que se conectam (RUSSELL; NORVIG, 2010).

Na Figura 2.9, o nó *Terremoto* (também chamado como pai, ancestral, antecessor) representa semanticamente uma causa do nó *Alarme* (filhos).

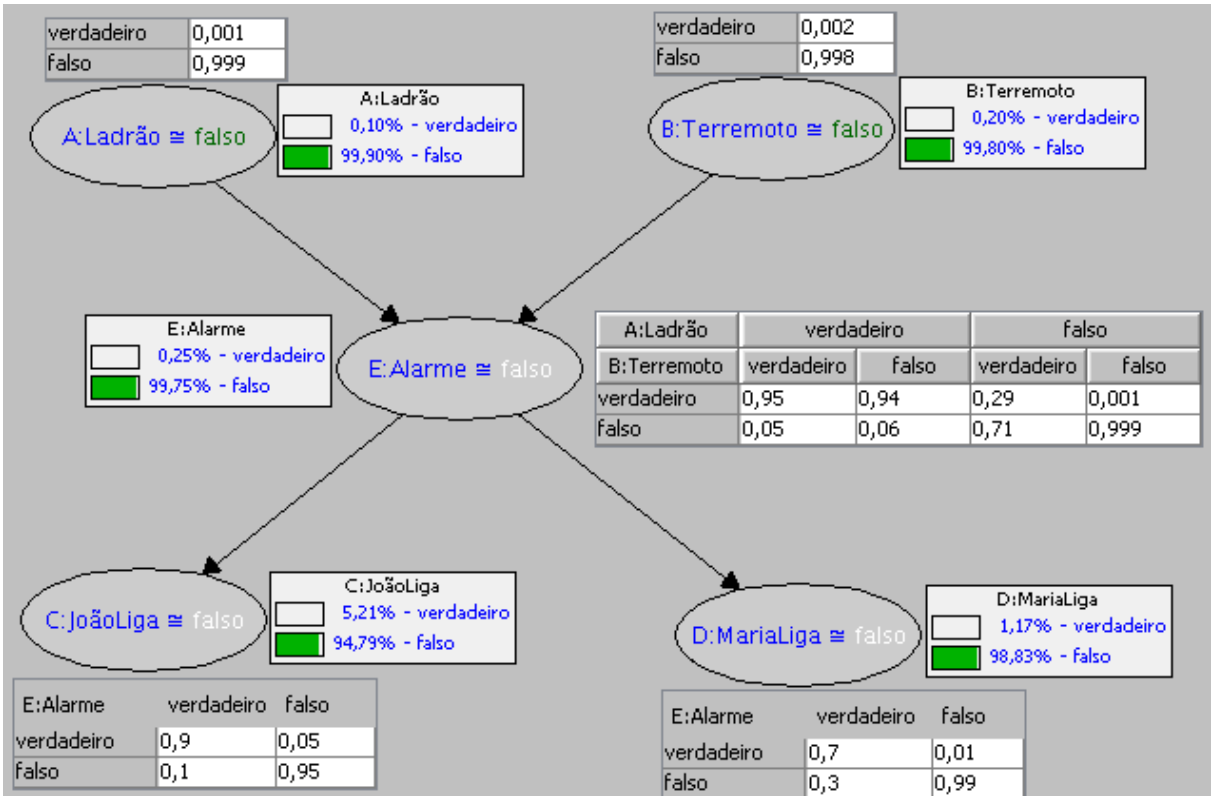


Figura 2.9: Uma típica rede Bayesiana, apresentando a topologia e as CPTs. Adaptação do uso prático apresentado em HOFFMANN (2014).

A especificação completa de RUSSELL; NORVIG (2010) para redes *Bayesianas* é dada a seguir:

- Um conjunto de variáveis aleatórias constitui os nós da rede. As variáveis podem ser discretas ou contínuas.
- Um conjunto de vínculos ou setas, conecta pares de nós. Se houver uma seta do nó X até o nó Y, X será denominado pai de Y.
- Cada nó X_i tem uma distribuição de probabilidade condicional $P(X_i|pa(X_i))$ que

quantifica o efeito dos pais sobre o nó, em que *pa(X_i)* denota os pais do nó X_i.

- O grafo não tem nenhum ciclo orientado, ou seja, é um grafo acíclico.

Cada nó está associado a uma TPC, que quantifica os efeitos dos nós pais sobre o nó, como ilustrado na Figura 2.9. As TPCs de todos os nós representam os parâmetros numéricos ou nominais como θ. Observe que o nó *Terremoto* não possui um pai, assim a tabela de probabilidade é reduzida para uma probabilidade incondicional, conhecida também como probabilidade a *priori P(Terremoto)*.

2.4.3 Cálculo da Distribuição de Probabilidade Conjunta

Baseado no cálculo das tabelas de probabilidade condicional, pode-se obter a distribuição de probabilidade conjunta do domínio a que um determinado conjunto de dados se refere. Assim, considerando-se $X_1, X_2, ..., X_n$ como os nós de uma rede *Bayesiana* e pegando-se da estrutura desta rede situações em que se tem independência condicional, então tem-se a equação (2.4), que permite determinar a probabilidade conjunta de todos os nós.

$$P(U) = P(X_1, X_2, ..., X_n) = \prod_{i}^{n} P(X_i | pa(X_i)) \tag{2.4}$$

em que, P(U) é a probabilidade conjunta para a rede; *pa(X_i)* são os pais do no X_i, e $P(X_i|pa(X_i))$ são as probabilidades condicionais de X_i em relação a seus pais.

Considere a rede Bayesiana da Figura 2.9, extraída de RUSSELL; NORVIG (2010), como exemplo: "Você possui um novo alarme contra ladrões em casa. Este alarme é muito confiável na detecção de ladrões, entretanto ele também pode disparar caso ocorra um terremoto. Você tem dois vizinhos, João e Maria, que prometeram telefonar-lhe no trabalho caso o alarme dispare. João sempre liga quando ouve o alarme; entretanto, algumas vezes, confunde o alarme com o telefone e também liga nestes casos. Maria, por outro lado, gosta de ouvir música alta e, às vezes, não escuta o alarme."

Uma vez definida a topologia, é necessário definir a TPC para cada nó. Cada linha na tabela contém a probabilidade condicional para cada caso condicional dos nós pais. Um caso condicional é uma possível combinação dos valores para os nós pais. Como Terremoto e Ladrão não têm pais, suas tabelas de probabilidades condicionais têm apenas as colunas representando seus estados, na qual estão as probabilidades a *priori* da variável.

2.4.4 Aprendizagem Bayesiana

O aprendizado Bayesiano é adequado em trabalhos sob condições de incerteza, expressando conhecimento probabilístico, relacionando as probabilidades condicionais que ocorrem entre as variáveis de um determinado conjunto de dados.

Dentro do processo de aprendizagem, é necessário calcular as distribuições de probabilidade (parâmetros numéricos) e identificar a estrutura da rede, ou seja, identificar as variáveis e as relações de interdependência dadas pelos arcos (JÚNIOR, 2003).

Conceitualmente, o processo de aprendizagem para rede Bayesiana está dividida em: aprendizagem da estrutura (relações entre as variáveis); e a aprendizagem de parâmetros numéricos (distribuição de probabilidades). Tanto a estrutura como os parâmetros da rede Bayesiana podem ser construídos a partir do conhecimento de um especialista; porém, dependendo do domínio a ser modelado, esta pode ser uma tarefa complexa. Considerando isto, a estrutura e os parâmetros da rede podem ser construídos utilizando a aprendizagem indutiva, em que se utiliza um conjunto de dados de exemplos, e partindo deste, a rede é construída automaticamente.

Na literatura de redes Bayesianas, vários algoritmos de aprendizagem indutiva foram propostos, com o objetivo de encontrar a estrutura da rede que represente adequadamente a relação das variáveis de um determinado domínio, bem como algoritmos que determinem as distribuições de probabilidades dos nós da estrutura da rede Bayesiana.

De acordo com JÚNIOR (2003), o processo de obtenção dos parâmetros numéricos é geralmente mais simples do que o processo de construção estrutural da rede, considerando que a rede Bayesiana já esteja estruturada. Alguns algoritmos para aprendizagem paramétrica são:

- Maximum likelihood (ML). Este paradigma de aprendizagem tem o objetivo de maximizar L(θ) *máxima verossimilhança*. O método de máxima verossimilhança estima os valores dos diferentes parâmetros do modelo estatístico de maneira a maximizar a probabilidade dos dados observados (isto é, busca parâmetros que maximizem a função de verossimilhança).

- Expectation–Maximization (EM). Os parâmetros para RB podem ser aprendidos até mesmo quando o conjunto de dados está incompleto, i.e., os valores de algumas variáveis são desconhecidos. Comumente, o algoritmo Expecta-

tion–Maximization é usado, pois estima os valores em falta, calculando os valores esperados, a fim de atualizar os parâmetros que usam esses valores esperados, como se fossem valores observados.

Para a aprendizagem de estrutura, JÚNIOR (2003) observa que existem várias metodologias na literatura, sendo que cada uma se aplica melhor em um tipo de aplicação. Por serem bastante específicas, não é possível definir qual é a melhor. Dentre os métodos de aprendizagem estrutural, existem duas abordagens diferentes:

- A abordagem baseada em pontuação, do inglês *score-based*, um número de grafos candidatos é gerado, e o que melhor se enquadrar em alguns tipos de pontuação é escolhido. As funções de pontuação são baseadas em diferentes princípios, tais como entropia e informação (CHOW; LIU, 2006), a descrição com comprimento mínimo (BOUCKAERT, 1994), ou abordagens Bayesianas (BUNTINE, 1991), como o BIC *(Bayesian Information Criterion)*.

- Na abordagem baseada em restrições, do inglês *constraint-based*, é derivado um conjunto de propriedades de independência condicional e incondicional dos dados. Em seguida, um DAG é construído, representando essas propriedades de independência com a maior precisão possível. Essa abordagem realiza um estudo qualitativo das relações de dependência e independência entre as variáveis de um domínio como em estudos (CHENG et al., 2002) e (CAMPOS; HUETE, 2000), a fim de encontrar uma rede que represente tais relações.

Capítulo 3

TRABALHOS RELACIONADOS

A literatura especializada tem evidenciado de maneira imperativa a necessidade de acompanhar o desenvolvimento de pesquisas científicas, que se destacam por ser o berço da inovação. Neste capítulo, são apresentados estudos que podem contribuir na pesquisa, ao permitir identificar metodologias, resultados e *gaps* nos experimentos.

3.1 Estado da Arte

O interesse por pesquisas que abordam "estado da arte" deriva da abrangência desses estudos para apontar caminhos que vêm sendo tomados e aspectos que são abordados em detrimento de outros. A realização destes balanços possibilita contribuir com a organização e a análise na definição de um campo, uma área, além de indicar possíveis contribuições para a pesquisa. A análise do campo investigativo é fundamental neste tempo de intensas mudanças associadas aos avanços crescentes da ciência e da tecnologia.

Estados da arte podem significar uma contribuição importante na constituição do campo teórico de uma área de conhecimento, pois procuram identificar os aportes significativos da construção da teoria e da prática, apontando as restrições sobre o campo em que se move a pesquisa, as suas lacunas de disseminação, identificar experiências inovadoras investigadas que apontem alternativas de solução para os problemas da prática e reconhecer as contribuições da pesquisa na constituição de propostas na área focalizada. Neste sentido, algumas pesquisas que possam contribuir para a constituição do trabalho proposto foram relacionadas.

Tabela 3.1: Critérios e subcritérios definidos. Fonte: (FERREIRA; MARTINS; SALGADO, 2015)

Critérios	Subcritérios
Conexão	Banda Larga (BLI) Redundância (RI) Disponibilidade (DISPO)
Custo	Instalação Mensalidade Capacitação da equipe técnica (CET)
Infraestrutura	Segurança Usabilidade Escalabilidade
Eficiência	Suporte técnico (ST) Sincronização de serviços (SS) Amplo acesso

3.1.1 Método para identificar o melhor modelo de computação em nuvem para empresas juniores

O estudo realizado por FERREIRA; MARTINS; SALGADO (2015) - *"Aplicação do método analytic hierarchy process para a adoção de computação em nuvem em empresas juniores"* - faz o uso da metodologia AHP com o objetivo de identificar qual é o melhor modelo de computação em nuvem para uma empresa júnior.

Empresas juniores (JÚNIOR, 2012) são associações sem fins lucrativos e crescem de acordo com sua necessidade, tendo seu lucro reinvestido em benefícios para a própria empresa. Dadas as vantagens de flexibilidade, disponibilidade e escalabilidade dos recursos computacionais, muitas empresas já estudam direcionar seus investimentos para serviços de computação em nuvem. Porém, empresas no Brasil, tal como as empresas juniores, ainda têm dúvida na escolha de qual modelo de implantação é o mais adequado, dentre os quais estão: Público, Privado, Híbrido e Comunitário. Como ferramenta de apoio para realizar a decisão de qual é o melhor método para empresas juniores, o AHP (Analytic Hierarchy Process) foi selecionado com a ferramenta para esta pesquisa.

O AHP é caracterizado pela decomposição de um problema em uma hierarquia, composta por diversos níveis. Essa hierarquia possui estrutura semelhante à de uma árvore, em que o primeiro nível da estrutura seria a raiz, e os critérios e subcritérios podem ser interpretados como as folhas que compõem a árvore (DUMOULIN; GUIMARÃES; NEVES, 2006). O método realiza a comparação paritária entre critérios e subcritérios, trazendo a seguinte questão: Qual a importância de um critério em relação ao outro? (DODGSON et al., 2009). As comparações expressas verbalmente por meio da escrita são

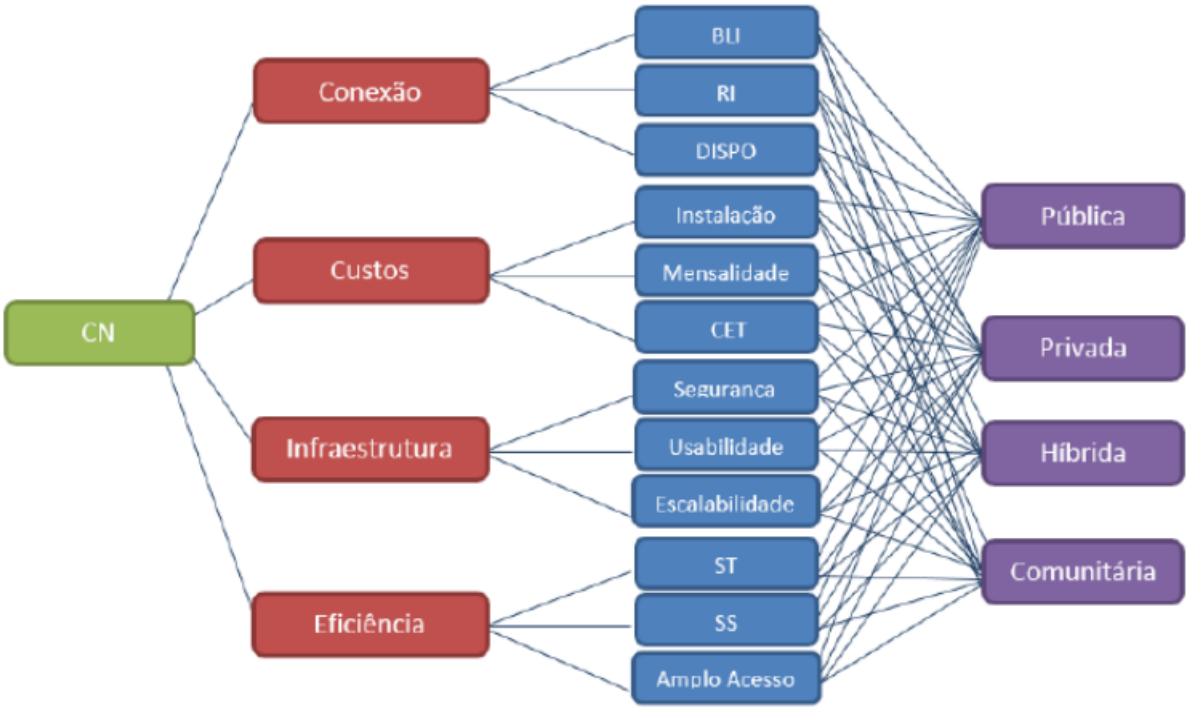


Figura 3.1: Estrutura hierárquica gerada. Fonte: (FERREIRA; MARTINS; SALGADO, 2015)

convertidas em valores numéricos, utilizando a Escala Fundamental de Saaty (SAATY, 2010), para posteriormente realizar os cálculos necessários, a fim de constatar qual é a melhor decisão a ser tomada para a devida ocasião.

Os critérios e subcritérios escolhidos para a criação da árvore do método AHP foram definidos de acordo com os modelos de serviços e de implantação da computação em nuvem. A Tabela 3.1 apresenta a descrição dos critérios e subcritérios. A estrutura hierárquica foi gerada a partir de um objetivo específico e bem definido, representada por CN (Computação em Nuvem), sendo dividida em critérios, subcritérios e alternativas, como apresentado na Figura 3.1. Contendo a definição da estrutura, os autores preparam documentos para a realização das entrevistas, pontuando os critérios e subcritérios para que, posteriormente, fossem realizados cálculos que podem ser encontrados em FERREIRA; MARTINS; SALGADO (2015).

Analisando os dados resultantes dos cálculos realizados, conclui-se que o grupo entrevistado optou pelo serviço de nuvem privada. Apesar de o serviço possuir custos, o modelo de nuvem privada é a melhor alternativa para ser adotada em empresas juniores, isso porque os entrevistados prezaram por recursos computacionais dedicados, segurança, suporte técnico e disponibilidade.

3.1.2 Explorando a computação em nuvem com medições passivas

O estudo realizado por BERMUDEZ et al. (2013) - *"Exploring the cloud from passive measurements: The Amazon AWS case"* apresenta uma caracterização da Amazon Web Services (AWS), através de medições passivas do provedor de nuvem mais proeminente (AWS) que oferece computação, armazenamento e plataformas de distribuição de conteúdo.

Para a caracterização dos serviços da AWS, foi necessário gerar um *dataset* (conjunto de dados) dos serviços em operação. Com esse objetivo, foi empregado o uso da ferramenta *Tstat*[1], um *snifffer*[2] que captura os pacotes de rede, colocando a interface de rede em modo passivo, gerando medidas passivas dos padrões de tráfego da rede para análise posterior. Uma *metodologia de análise* foi criada para analisar pacotes trocados por usuários finais dentro de pontos estratégicos de monitoramento, com o objetivo de avaliar os serviços que são executados na AWS, e como eles são acessados.

A metodologia de análise combinou o uso do banco de dados da organização *MaxMind*[3] para isolar todos os endereços IPs conhecidos da Amazon, e o uso das informações fornecidas pelo DNS para identificar os fluxos de dados dirigidos aos serviços *Amazon EC2*, que é um serviço web que fornece capacidade de computação redimensionável na nuvem. Para desvendar conteúdos ou serviços entregues por cada conexão, foi usado o *DN-Hunter*, um sistema que aproveita as informações fornecidas pelo tráfego DNS para fornecer a identificação dos fluxos de dados e dar visibilidade ao tráfego, mesmo para o tráfego criptografado. A identificação da localização geográfica dos datacenters é inferida com o uso do traceroute e da latência (BERMUDEZ et al., 2012).

Entre as diferentes medições realizadas por *Tsat* e *DN-Hunter*, foram consideradas métricas por fluxo como tempo de resposta e *goodput*, que é a quantidade de dados úteis transmitidos num determinado link durante um certo tempo. Com o objetivo de avaliar o custo sustentado pela rede para troca de dados entre AWS e os usuário finais, foi definida uma média ponderada da distância percorrida por unidade de informação; com isso, diferentes definições de distâncias foram consideradas como: i) a média RTT de conexões TCP; ii) o número de ASs percorridos, que é obtido executando o traceroute a partir de pontos estratégicos do caminho, ou até mesmo, iii) a distância física

[1] http://tstat.tlc.polito.it/
[2] *Snifffer* são programas que têm como princípio capturar pacotes de rede.
[3] https://www.maxmind.com/pt/geoip2-services-and-databases

geodésica. Assim, obtiveram-se diferentes métricas de custo de rede.

Na caracterização espacial dos serviços da AWS, os dados mostram que, mesmo o datacenter da Irlanda sendo mais próximo dos usuários finais da Itália (e Europa) do que o da Virgínia (que sedia instâncias EC2), os clientes da AWS, por causa de uma gestão simples e/ou por razões econômicas, são orientados a implantar seus serviços em um único datacenter na Virgínia. Já investigando o serviço de distribuição de conteúdo, CloudFront, identificou que, devido a mudanças nas políticas de DNS da Amazon, usuários que eram tipicamente servidos pelo MXP (Serviço de cache dos servidores de Milão/Itália) tinham sido redirecionados para ARN (Servidores de Estocolmo/Suécia) durante aquele período analisado. Isso levou à conclusão de que políticas CloudFront são dinâmicas, em contraste com a alocação estática de serviços EC2.

Os resultados mostraram, em termos de *tempo de resposta* para serviços EC2 e *goodput* para serviços S3 de armazenamento na nuvem, que o datacenter mais popular da Amazon AWS é também o de pior performance, isso porque há um grande desequilíbrio entre o workload[4] de diferentes datacenters que hospedam os serviços. Observou-se, também, que empresas dependentes dos serviços EC2 concentram seu conteúdo em um único datacenter, causando o aumento de custo sustentado pela rede para transportar dados aos usuários finais distantes, e aumentando o risco de parada dos serviços em caso de falhas. A respeito do desempenho, os usuários que acessam os serviços de Virgínia (EUA) possuem o pior desempenho, mas não foi possível identificar as causas reais. Foi identificado, também, que o serviço CloudFront apresenta excelente desempenho, mas problemas típicos de outros sistemas de CDN como: DNS genéricos retornando caches de regiões mais afastadas dos usuários finais e conteúdos impopulares gerados pelos usuários, causando a degradação da qualidade do serviço.

3.1.3 Detecção de problemas de desempenho em sistemas na computação em nuvem

O estudo realizado por MI et al. (2011) - *"Performance Problems Online Detection in Cloud Computing Systems via Analyzing Request Execution Paths"* propõe uma abordagem para rapidamente diagnosticar a origem da degradação da performance em sistema de computação em nuvem de larga escala.

É muito desgastante detectar problemas de performance na execução de sistemas

[4]Carga de trabalho

em *cloud computing*. O comportamento e as conexões ocultas entre a enorme quantidade de solicitações de caminhos de execução, em tempo de execução, geralmente contêm informações úteis para a detecção de problemas de desempenho. Nesta abordagem de diagnóstico online da primeira causa de problemas de desempenho para sistemas de computação em nuvem, primeiro agrupam-se as solicitações de usuários dentro de categorias que contêm o mesmo caminho de execução. Estas solicitações de usuários são conhecidas como sequência de métodos que são invocados sequencialmente; depois, aplica-se a extração dos métodos primários com o uso do procedimento estatístico PCA (principal component analysis), que utiliza uma transformação ortogonal, para converter um conjunto de observações de possíveis variáveis correlatas, para um conjunto de valores de variáveis não correlacionadas linearmente, chamados componentes principais. Dessa forma, são comparados os métodos normais e anormais dos métodos primários, para finalmente localizar a principal causa dos problemas de desempenho.

O experimento foi realizado na plataforma de computação em nuvem, da empresa **Alibaba**[5], para a implantação do serviço de compartilhamento de arquivo, com o objetivo de detectar problemas na infraestrutura, focando nas operações *Listar* e *Salvar* Arquivos, observando as variações nas operações de CPU e Disco, bem como a latência no tempo de resposta das operações. Ao procurar métodos que poderiam degradar o desempenho da aplicação, foi identificado que o método de leitura dos metadados do usuário é a principal causa do aumento no tempo de resposta na operação *Listar Arquivo*.

Os resultados mostraram que rastrear as solicitações fim a fim é uma forma de identificar e entender a primeira causa do problema de desempenho, pois identificar os problemas enquanto on-line pode facilitar na busca da causa raiz dos problemas de forma mais eficiente e precisa.

3.1.4 Redes Bayesianas para o diagnóstico de falhas

O estudo realizado por YISHAN; CHEN; JIA (2009) - "The Bayesian Network about Computer Network Failure Diagnosis under OSI" apresenta um procedimento de construção da RB e declara seus mecanismos de raciocínio, a fim de resolver o tradicional problema de diagnóstico de falhas em redes de computadores com baixa eficiência.

[5]http://www.aliyun.com/

Com o rápido desenvolvimento da telecomunicação e suas tecnologias, redes com diferentes sistemas e estruturas são interligadas. Tudo isso torna mais complicada a proteção e a manipulação dos dados de toda a rede. Então, as pessoas tentam simplificar a manutenção da rede com vários métodos. Redes Bayesianas chamam a atenção pela sua vantagem na predição de incertezas. Para a construção do modelo de rede Bayesiana de diagnóstico de falha, foi usado o modelo de referência OSI, que contém sete camadas como fundamento da transmissão de dados em rede. Cada camada tem suas características de falhas. Dessa forma, a RB que é constituída, contém 2 tipos de nós: um que corresponde a *local de falha* e outro a *característica da falha*. As Tabelas 3.2 e 3.3 apresentam uma descrição dos nós com sua respectiva abreviação em inglês.

Tabela 3.2: Local de falha e seu código. Fonte: (YISHAN; CHEN; JIA, 2009)

code	*name*	*code*	*name*
F	failure	PL	Physical layer
DLL	Data linked layer	NL	Network layer
TL	Transmission layer	T3	Top 3
TM	Transmission media	C	Connector
PA	Panel Association	HUB	Hub
TC	TC	MC	Media Convention
NIC	Network Card	BR	Bridge
SW	Switcher	RO	Router
TCP	Transport control protocol	IP	Intent protocol
DNS	DN system	DHCP	DHC Protocol

Tabela 3.3: Caracterítica da falha e seu código. Fonte: (YISHAN; CHEN; JIA, 2009)

F1	Signal diminished	F2	Signal transformed
F3	Noise disturbance	F4	Network disconnect
F5	Slow opening after NIC is installed	F6	NICPL
F7	NIC working but unable to connect with outside	F8	Insufficient capacity
F9	Data lost	F10	Bridge not working
F11	SW not working	F12	RIP failure
F13	OSPF failure	F14	BGP failure
F15	Router hardware failure	F16	TCP test and faults
F17	IP test and faults	F18	Non simultaneity of Service Bank
F19	IP address fault		

Este trabalho demonstrou que, pela combinação de conhecimentos, é conveniente a criação do modelo de diagnóstico de falha em RB. Para o trabalho proposto, a rede Bayesiana da Figura 3.2 foi desenvolvida utilizando o procedimento na respectiva ordem: 1º identificar cada nó da camada OSI, nó de falha e nó característico da falha; 2º definir modelo de grupo de estados, por exemplo, normal ou anormal, e identificá-lo como eventual critério do nó relacionado; 3º construir relacionamento da propagação de falha nos nós relacionados, por exemplo, a anormalidade de DHCP pode resultar da falha do endereço IP, e este pode ser marcado como IP $\rightarrow$ DHCP; 4º construir a tabela de probabilidade condicional (CPT), desenvolvida de acordo com o procedimento acima.

O mecanismo de raciocínio realizado através de RB foi mais uma representação formal das regras para realizar o Diagnóstico de Falhas, na busca da relação causa-efeito da falha, bem como sua probabilidade.

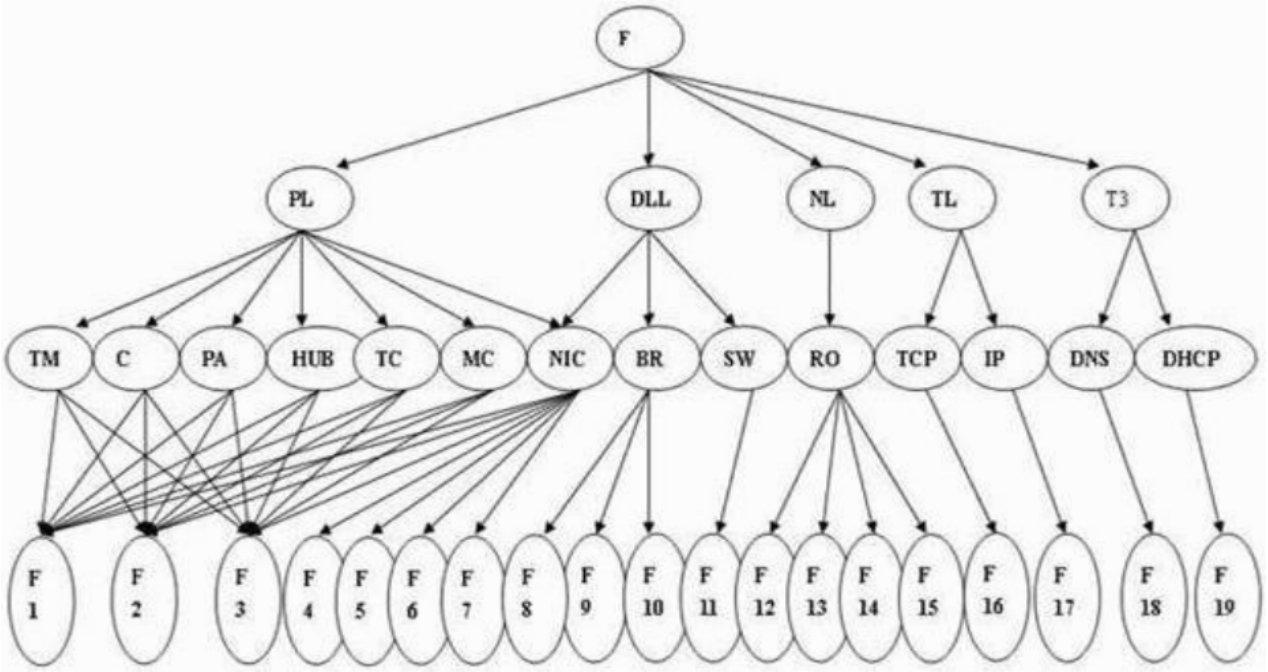


Figura 3.2: Rede Bayesiana para o diagnóstico de falhas em redes de computadores. Fonte: (YISHAN; CHEN; JIA, 2009)

3.1.5 Construção de um modelo de propagação de falhas com Rede Bayesiana

O estudo realizado por LI; ZHAO; YIN (2009) - "A Method of Building the Fault Propogation Model of Distributed Application Systems Based on Bayesian Network" propõe um método para a construção do Modelo de Propagação de Falha em Sistemas de Aplicação Distribuída, baseado em Rede Bayesiana.

A fim de realizar diagnóstico de falhas em sistemas de aplicações distribuídas, o Modelo de Propagação de Falha (MPF) é geralmente o pré-requisito necessário para as tarefas subsequentes, tais como o raciocínio probabilístico, a previsão e a recuperação de falhas. Neste trabalho, a construção do MPF que combinou amostra de dados e conhecimento especializado foi baseada em rede Bayesiana.

Primeiramente, foi realizado o mapeamento MPF do sistema de aplicação distribuída para uma rede Bayesiana, com o conhecimento de um especialista, a fim de fornecer uma referência intuitiva para resultados da construção do método MPF proposto e, assim comparar sua eficácia. Nessa abordagem, os nós que representam os serviços fornecidos pelo sistema são abstraídos do hardware e do software necessário para o serviço, e linhas direcionadas representam as relações de dependências condicionais entre os componentes de serviço.

Para a construção do MPF, foi empregado o método baseado na busca de pontuação (search-scoring) para a construção da estrutura da rede. A chave do método é escolher um apropriado algoritmo de busca e uma adequada função de pontuação. Foi escolhida a função de pontuação BIC (Bayesian Information Criterion) (BAYES..., 2014). O algoritmo selecionado foi o "greedly search" (GS), escolhido como algoritmo de seleção de modelo com o objetivo de escolher o modelo com maior pontuação BIC. Com essas definições, o método modelado segue na Figura 3.3; em primeiro lugar, com o algoritmo "maximum weight spanning tree algorithm" (MWST) (HECKERMAN, 1999), cria-se uma árvore inicial que melhor se ajusta às amostras de dados; em segundo lugar, a árvore inicial é revisada por especialistas. Por fim, o modelo final é aprendido, utilizando o algoritmo GS com a estrutura revista como seu modelo de entrada inicial.

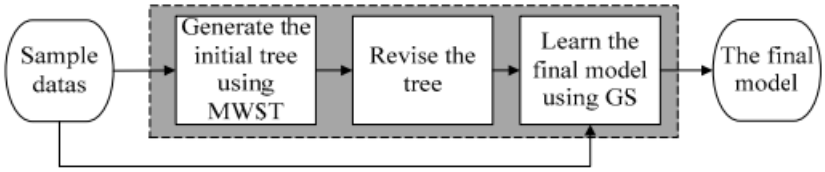


Figura 3.3: O diagrama de fluxo do método de modelagem

O experimento foi realizado usando ferramentas de redes Bayesianas, desenvolvidas com base no Matlab por Kevin P.Murphy e outros. Para o experimento, foi criado um ambiente de aplicação distribuída, em que dados medidos foram coletados através de inferência ativa. Em seguida, esses dados foram pré-processados para adequado resultado na execução do experimento. O resultado do experimento mostra que, depois de integrar o conhecimento de um especialista, o modelo de aprendizagem final é exatamente o mesmo que o real. Por isso, pode-se concluir que o método de modelagem proposto neste trabalho pode ser aplicado para a construção do FPM de sistemas distribuídos.

3.2 Análise de resultados dos trabalhos

Os estudos e os resultados dos trabalhos apresentados contribuíram para a constituição teórica desta dissertação, onde os trabalhos de YISHAN; CHEN; JIA (2009) e LI; ZHAO; YIN (2009) abordaram métodos de diagnóstico de falhas em redes de computadores e sistemas distribuídos com o uso de redes Bayesianas, os trabalhos de MI et al. (2011) e BERMUDEZ et al. (2013) exploraram computação em nuvem, abrangendo seus estudos à metodologias de análise do desempenho, e o trabalho de FERREIRA; MAR-

TINS; SALGADO (2015) apresenta um panorama da adoção de computação em nuvem nas empresas juniores.

O trabalho de YISHAN; CHEN; JIA (2009) obteve como resultado um mecanismo de raciocínio realizado através da rede Bayesiana, atacando o problema de diagnóstico das falhas em redes de computadores, que buscou a relação causa e efeito, já o trabalho (LI; ZHAO; YIN, 2009) (2009) propôs um método para a construção de um modelo em rede Bayesiana a fim de realizar o diagnóstico de falhas em sistemas distribuídos. Como resultado, o experimento mostrou que integrar o conhecimento de um especialista na criação de um rede Bayesiana contribui para uma modelagem mais próxima da realidade.

Os demais trabalhos são focados na *computação em nuvem* como o MI et al. (2011), que propõe diagnosticar rapidamente a origem da degradação de desempenho em sistemas de computação em nuvem, identificaram que rastrear as solicitações fim a fim dos serviços é uma forma de identificar e entender a primeira causa do problema de desempenho. O trabalho (BERMUDEZ et al., 2013), focado principalmente no provedor de nuvem AWS, identificou, através de medições passivas, que existe um grande desequilíbrio entre a carga de trabalho dos diferentes *data centers* que hospedam os serviços EC2 e S3, principalmente porque empresas que dependem desses serviços concentram seu conteúdo em um único *data center*, aumentando o risco de parada dos serviços em caso de falhas.

Nesses trabalhos, foi possível identificar experiências inovadoras que apontam alternativas e contribuições para a pesquisa. Nos trabalhos de YISHAN; CHEN; JIA (2009) e LI; ZHAO; YIN (2009), ficou evidente que a rede Bayesiana é uma ferramenta eficiente para o diagnóstico de falhas, mas é importante na sua construção e validação a contribuição de um especialista do domínio, pois é ele que conhece a relação causa e efeito das falhas apresentadas pelas variáveis. Já MI et al. (2011) e BERMUDEZ et al. (2013) realizaram através de monitoramento o diagnóstico de falhas e a caracterização do desempenho dos serviços na nuvem. Enquanto no trabalho (FERREIRA; MARTINS; SALGADO, 2015), a pesquisa demonstrou que empresas juniores como muitas outras no Brasil dão seus primeiros passos na computação em nuvem com o modelo de nuvem privada, levando em conta fatores como segurança, suporte técnico e recursos computacionais dedicados. Essas pesquisas elucidaram a proposta do trabalho, pois focamos nossa pesquisa no modelo de nuvem privada, monitorada, a fim de, melhorar o gerenciamento da saúde dos sistemas como o uso da rede Bayesiana.

Capítulo 4

PROPOSTA DO TRABALHO

Este trabalho propõe a criação de uma arquitetura para um sistema de suporte para a tomada de decisão com a aplicação de Rede Bayesiana (RB) empregada no gerenciamento da saúde de sistema em cloud computing, a fim de identificar sintomas de falhas nos sistemas e auxiliar no provisionamento dinâmico de recurso para remediar ou solucionar a causa dos problemas em sistemas e serviços em *cloud computing*.

Para fornecer um mecanismo que realize inferência probabilística e permita uma adequada tomada de decisão sobre o gerenciamento da saúde dos sistemas em nuvem, o uso de outras técnicas será utilizado para compor a arquitetura do trabalho proposto.

Antes de indicar um problema de saúde no sistema na nuvem, que pode ser desde uma falha no serviço ou no recurso computacional, é necessário realizar o monitoramento. O monitoramento permite aferir, instantaneamente, se há ou não um problema de saúde nos sistemas que não são monitorados adequadamente. Isso só é possível com ferramentas adequadas. Para este trabalho, é proposto o uso do Nagios como ferramenta de monitoramento. Nessa arquitetura, o Nagios tem o papel de garantir as restrições de SLA no monitoramento de serviços e sistemas, fornecendo seus respectivos dados de saúde. Esses dados serão inseridos em um módulo de suporte de decisão para um processamento adequado, dando a garantia de acurácia à informação para uma correta tomada de decisão.

Para um processamento adequado dos dados de monitoramento, é proposto o uso da metodologia KDD para a mineração dos dados nos logs de monitoramento do Nagios, realizando pré-processamento e transformação dos dados, operações que irão eliminar ruídos, dados redundantes e inconsistentes, permitindo avaliar e excluir possíveis dados discrepantes do conjunto de dados que será alvo de estudo.

O conjunto de dados adquirido após processamento servirá de insumo para a modelagem da rede Bayesiana. Aplicando os algoritmo Bayesianos sobre o conjunto de dados para o aprendizado estrutural e paramétrico da rede, aliado ao conhecimento de um especialista na computação em nuvem, será proposta uma RB para sistemas e serviços em uma nuvem privada de uma grande empresa sucroenergética no interior do Estado de São Paulo.

A arquitetura de gerenciamento será modularizada, sendo a rede Bayesiana uma proposta para o módulo de suporte a decisão, em que, dada uma nova evidência da saúde dos sistemas monitorados, será possível inferir a necessidade ou não do provisionamento de recursos computacionais, para garantir a entrega adequada dos serviços monitorados.

4.1 Arquitetura de gerenciamento proposta

Sob consideração do gerenciamento da saúde de sistemas em *Cloud Computing*, um conjunto de conceitos é proposto, a fim de resolver o problema do gerenciamento da saúde de sistema em *Cloud Computing*. A arquitetura proposta é apresentada na Figura 4.1. Modularizada, a arquitetura é composta por quatro módulos, dos quais o *Nagios* faz parte do módulo de monitoramento, a metodologia *KDD* faz parte do módulo de descoberta do conhecimento, a *Rede Bayesiana* faz parte do módulo de inferência e, por ultimo, para garantir a elasticidade dos recursos computacionais, o módulo de provisionamento.

A arquitetura apresentada na Figura 4.1 foi proposta para serviços que são demandados em uma nuvem no modelo IaaS, em que o bloco *Infraestrutura* corresponde ao provisionamento da infraestrutura para o software, como espaço adequado para o *data center*, gerenciamento de energia, refrigeração, disponibilização de servidores físicos, equipamentos de rede e segurança. O bloco de *Virtualização* oferece vários serviços, dentre os quais, computação, armazenamento e rede, que são disponibilizados através de máquinas virtuais (VMs). O bloco de *Virtualização* traz um aspecto de escalabilidade para nuvem, que pode ter sua configuração dinâmica, com base nas necessidades demandadas pelos clientes. Os serviços e os sistemas demandados pelos cliente estão relacionados no bloco *Sistemas e Serviços*.

Os serviços e os sistemas no modelo IaaS são instalados em máquinas virtuais, como apresentado na Figura 4.1, os recursos computacionais (e.g. CPU, memória, disco

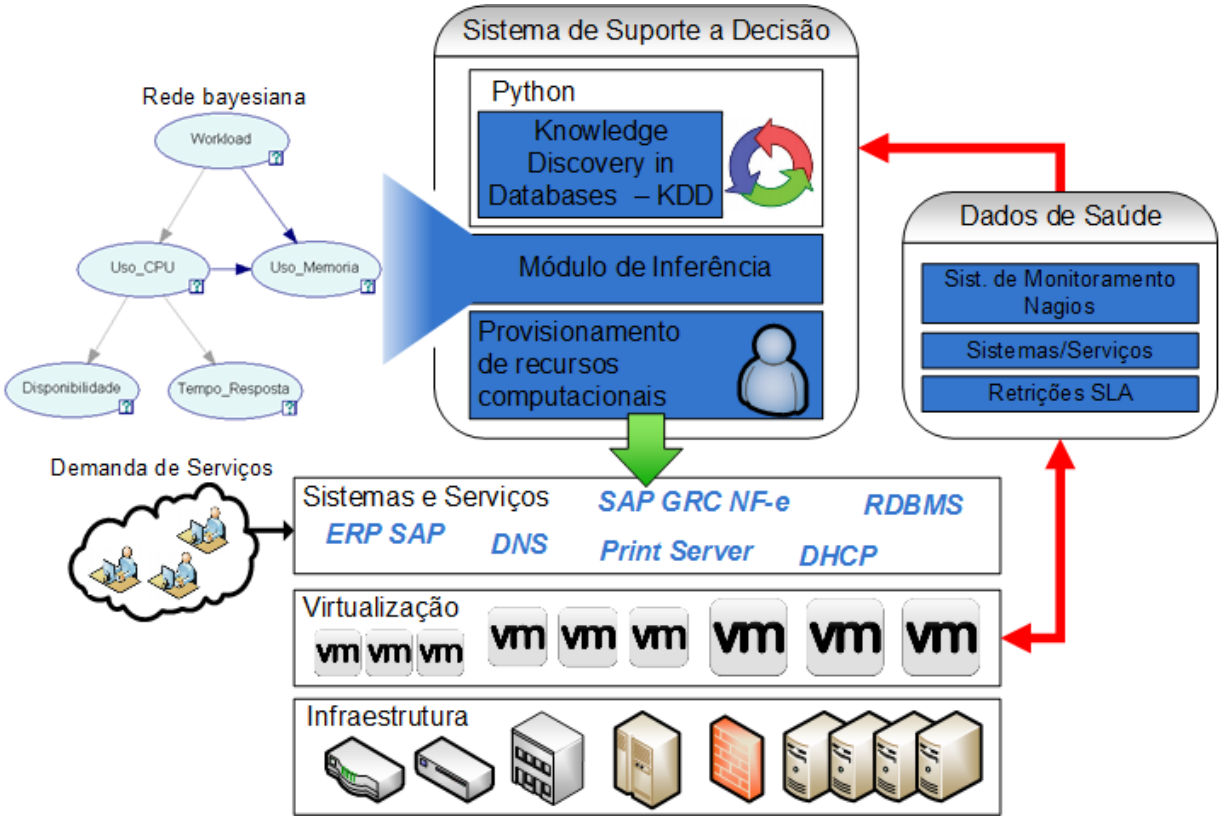


Figura 4.1: Proposta conceitual de SSD baseado em Rede Bayesiana para nuvem IaaS

e rede) precisam ser monitorados periodicamente, para garantir a disponibilidade e a qualidade, como previsto no SLA dos serviços e sistemas demandados.

Para a coleta de dados da saúde de serviços e sistemas, é proposto para a arquitetura o sistema de monitoramento Nagios. Estatísticas de monitoramento geradas pelo Nagios serão recolhidas para alimentar o SSD (Sistema de Suporte a Decisão), onde está incluso um módulo de inferência que interage com a rede Bayesiana, a fim de realizar a predição de falhas ou aumento/diminuição na demanda dos serviços e sistemas.

O sistema de suporte a decisão proposto, será composto de três módulos, responsáveis por realizar o processamento, a inferência e o provisionamento, com base nos dados de monitoramento da saúde dos sistemas e serviços, sendo que o processamento corresponde à etapa de preparação dos dados (e.g. discretização dos dados e seleção de atributos) e extração do conhecimento dos dados de monitoramento. O módulo de inferência corresponde ao desenvolvimento e modelagem de uma rede Bayesiana, capaz de inferir o real estado dos serviços e do sistema na nuvem, para que, no módulo de provisionamento, seja tomada uma decisão correta na gestão dos serviços e dos sistemas, e também no provisionamento adequado de recursos compu-

tacionais.

4.2 Módulos da arquitetura

A modularização da arquitetura é dividida em partes distintas, compondo o ferramental necessário para ter um panorama mais estruturado do projeto. Com as técnicas de programação, a modularização apresentada nessa arquitetura visa a integrar o ferramental necessário para a elaboração do SSD, visando principalmente, ao desenvolvimento com aspectos de confiabilidade, legibilidade, manutenção e flexibilidade. A arquitetura é dividida em quatro módulos: *módulo de monitoramento*, *módulo KDD*, *módulo de inferência*, e *módulo de provisionamento*, e a proposta resumida dos respectivos módulos será apresentada nesta seção.

4.2.1 Módulo de monitoramento

Esse módulo, além de definir restrições do contrato SLA, também define o monitoramento da saúde dos serviços e dos sistemas na nuvem. Nagios será o sistema de monitoramento utilizado. Propõe-se também recuperar do log de eventos do Nagios todo o histórico de monitoramento, para posterior processamento no módulo *KDD*.

O panorama de serviços e sistemas na nuvem é extenso; portanto, para a proposta deste trabalho, decidiu-se limitar a pesquisa para um grupo específico de servidores, serviços e sistemas, com o objetivo de analisar os estados nominais de cada item e identificar a frequência em que ocorrem os eventos que podem degradar a saúde dos itens analisados.

Uma importante atividade nesse módulo será realizar a configuração adequada do Nagios, dos hosts e serviços a serem monitorados, bem como identificar nos arquivos de configuração do Nagios onde estão sendo gravados os arquivos do log de eventos.

4.2.2 Módulo KDD

O módulo KDD (Descoberta de conhecimento em Banco de Dados) é proposto para essa arquitetura, a fim de descobrir padrões válidos e potencialmente úteis à aplicação e à modelagem da Rede Bayesiana, que é o principal artefato do SSD (Sistema de Suporte a Decisão). O modelo KDD genérico (KURGAN; MUSILEK, 2006) é indicado para

este trabalho, dada uma série de observações nos modelos citados e analisados.

Neste módulo, as atividades propostas são divididas em etapas, que compreendem atividades, tais como: a compreensão do domínio, ou seja, compreender as funcionalidades do Nagios que possam ser relevantes ao assunto tratado no processo de extração do conhecimento a seleção do conjunto de dados, mais especificamente um conjunto de arquivos de logs relevante ao processo.

Outras atividades, como as de pré-processamento e transformação, são mais custosas porque demandam mais esforço e tempo para realizá-las. Tendo em vista as dificuldades nas realizações destas atividades, é proposto para este trabalho o desenvolvimento de dois algoritmos: um para pré-processamento e outro para transformação do conjunto de dados selecionado para o processamento. Esses algoritmos serão desenvolvidos com a linguagem *Python* e irão desempenhar operações que consistem na eliminação de dados redundantes, inconsistentes e discrepantes do conjunto de dados-alvo, realizando a organização dos dados, dadas as particularidades do processamento seguinte a que os dados irão servir de insumo. A próxima etapa de processamento ficará sob a tutela do módulo de inferência.

4.2.3 Módulo de inferência

A ideia proposta para este módulo está ligada principalmente à construção de um modelo gráfico Bayesiano para realizar conclusões sobre a saúde dos sistemas monitorados na nuvem. A ideia está dependentemente ligada ao processamento realizado no módulo KDD, pois os dados processados servirão de insumo ao módulo de inferência. O fluxograma proposto na Figura 4.2 indica, sequencialmente, as etapas que serão realizadas para a construção e a validação do modelo probabilístico de inferência; para a nossa proposta, uma rede Bayesiana que permitirá responder a uma série de "consultas" sobre a saúde de sistemas na nuvem.

Na etapa de aprendizagem estrutural, propõe-se identificar a estrutura da rede Bayesiana, aplicando no conjunto de dados do módulo KDD o algoritmo PC, que é baseado em testes de independência das variáveis, a fim de construir os DAGs (Grafos acíclicos dirigidos). Propõe-se esse algoritmo, levando em conta que, possivelmente, as variáveis estarão desordenas; assim, a melhor forma de construir os DAGs é considerando a independência condicional entre elas.

Com a estrutura da rede Bayesiana conhecida, a etapa que realiza a aprendiza-

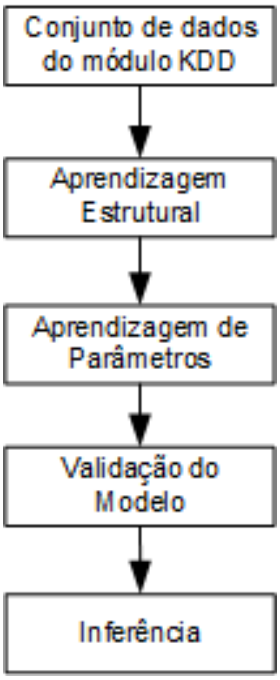


Figura 4.2: Proposta para a construção do Módulo de Inferência

gem de parâmetros atribui uma distribuição de probabilidades condicionais ou não condicionais a cada variável aleatória. Propõe-se então, para esta etapa, o uso do algoritmo EM (*Expectation Maximization*) como sendo o mais adequado para estimar a distribuição de parâmetros, pois as redes Bayesianas propostas não são consideras redes complexas. Uma rede Bayesiana é considerada complexa se contiver mais de 20 variáveis aleatórias (KORB; NICHOLSON, 2010). Outras razões levaram à escolha do algoritmo EM para aprendizagem de parâmetros, tais como: adequação às distribuições discretas e bom desempenho para observações parciais.

Definido o algoritmo e estimados os parâmetros da RB, o próximo passo é validar o modelo, para verificar se o modelo é preciso, a fim de realizar a inferência apropriada. Para a validação do modelo, é proposto o método conhecido por validação cruzada, denominado *k-fold cross-validation* (REFAEILZADEH; TANG; LIU, 2009). Este método consiste em dividir o conjunto total de dados em k subconjuntos mutuamente exclusivos do mesmo tamanho e, a partir disto, utilizar um subconjunto para teste e k-1 subconjuntos para treinamento e cálculo da acurácia do modelo. Depois da validação do modelo, a predição poderá ser realizada e utilizada pelo módulo de inferência.

4.2.4 Módulo de provisionamento

Dado uma nova evidência na rede Bayesiana no *módulo de inferência*, uma proposição é feita a ela, por meio da qual são feitos diagnóstico de falhas. Os diagnósticos levam então a são passados para o *módulo de provisionamento*. Este módulo tem a função de realizar o provisionamento dinâmico de recursos, atuando como controle de escalabilidade, com duas ações *scale down* e *scale up*, garantindo a escalabilidade automática, *scale up*, em caso de altas demandas de recursos ou falha de serviços e sistemas ou *scale down*, a redução do consumo de recursos computacionais em casos de baixa demanda.

Capítulo 5

EXPERIMENTOS E ANÁLISES

O Experimento descrito nesta seção envolve o serviço de nuvem privada no modelo IaaS de uma das maiores empresas sucroenergéticas do Brasil, o Grupo São Martinho S/A, com o propósito de modelar um Sistema de Suporte à Decisão, baseado em rede Bayesiana, a fim de gerenciar a saúde dos sistemas e serviços demandados pelos clientes internos.

A Figura 5.1 ilustra a arquitetura do serviço IaaS do Grupo São Martinho S/A, bem como os sistemas e serviços que esta infraestrutura suporta. Nessa arquitetura de nuvem privada o *hypervisor* utilizado é o vSphere da VMware, que virtualiza os recursos físicos dos servidores *bare-metal*, tais como memória, disco, processadores e interfaces de rede, e fornece-os como um pool de recursos para as máquinas virtuais (VMs), e as VMs entregam aos clientes internos através da infraestrutura de rede corporativa os sistemas e serviços demandados.

Podem-se observar, na Figura 5.1, alguns dos serviços e sistemas suportados pela nuvem privada IaaS do Grupo São Martinho S/A, dos quais se tem, um *cluster* SAP ERP composto pelos servidores *SAP App Server 1*, *SAP App Server 2* e *SAP HA Central Instance*, um serviço de nota fiscal eletrônica da SAP no servidor *SAP GRC NF-e*, um serviço de impressão no servidor *Print Server* e serviços de rede, tais como DNS e DHCP no servidor *Net Server*. Existe um número amplo de sistemas e serviços suportados pela infraestrutura do Grupo São Martinho, dadas suas amplas redes de cadeias produtivas em suas bases territoriais; porém, para alvo de nossos estudos e experimentos, limitar-se-á aos serviços e sistemas descritos neste parágrafo.

O grau de complexidade do ambiente e o grande número de sistemas e serviços levaram à adoção da ferramenta de monitoramento Nagios com o objetivo de gerenciar

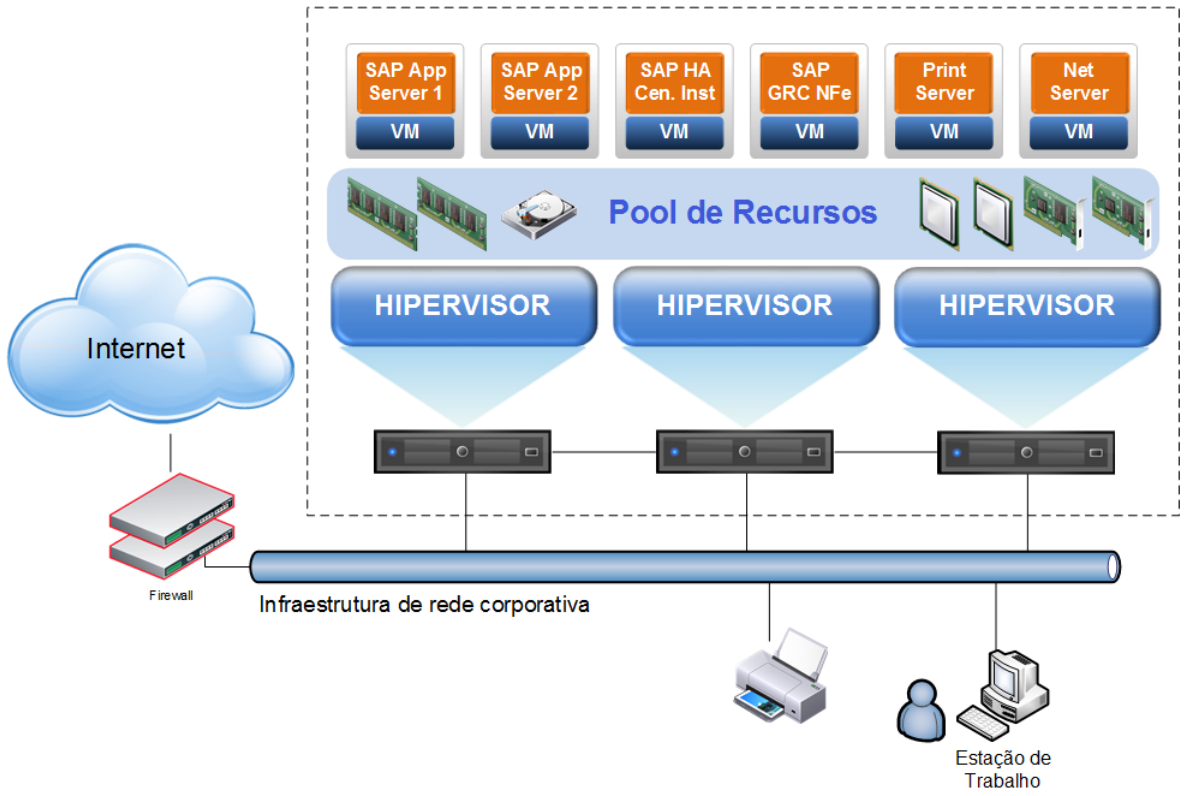


Figura 5.1: Arquitetura de nuvem privada IaaS do experimento

a qualidade da entrega dos serviços ofertados aos clientes, permitindo registrar e analisar informações de status dos serviços e sistemas monitorados. O Nagios vem sendo usando pelo Grupo São Martinho S/A há mais de cinco anos, e atualmente ele possui uma grande quantidade de hosts e serviços sendo monitorados, pontuando o tempo de indisponibilidade dos hosts e serviços de acordo com o SLA. O procedimento para novas instalações e configurações do Nagios é apresentado no Apêndice A, visto que, para o módulo de monitoramento da arquitetura proposta, é necessário identificar os arquivos de configuração, bem como realizar as configurações adequadas do Nagios.

5.1 Dados de saúde monitorados com o Nagios

O Nagios foi configurado para monitorar a saúde dos sistemas e serviços de três formas, como demonstrado na Figura 5.2. Para o monitoramento de recursos privados, como memória, disco e CPU dos hosts Linux/Unix, foi usado o plugin *check_by_ssh*, e para monitoramento de recursos privados dos hosts Windows, foi usado o plugin *check_nrpe*. Entretanto, para monitoramento de serviços públicos, tais como os serviços DHCP, DNS e FTP, dentre outros serviços que são disponibilizados através de protocolos TCP e UDP, foram utilizados outros plugins Nagios (e.g. *check_dhcp*, *check_dns*,

check_ftp, *check_tcp* e *check_udp*), para verificar se os serviços estão disponíveis ou não para os usuários. Os hosts destacados em laranja, na Figura 5.1, foram monitorados com o uso dos respectivos plugins citados neste parágrafo.

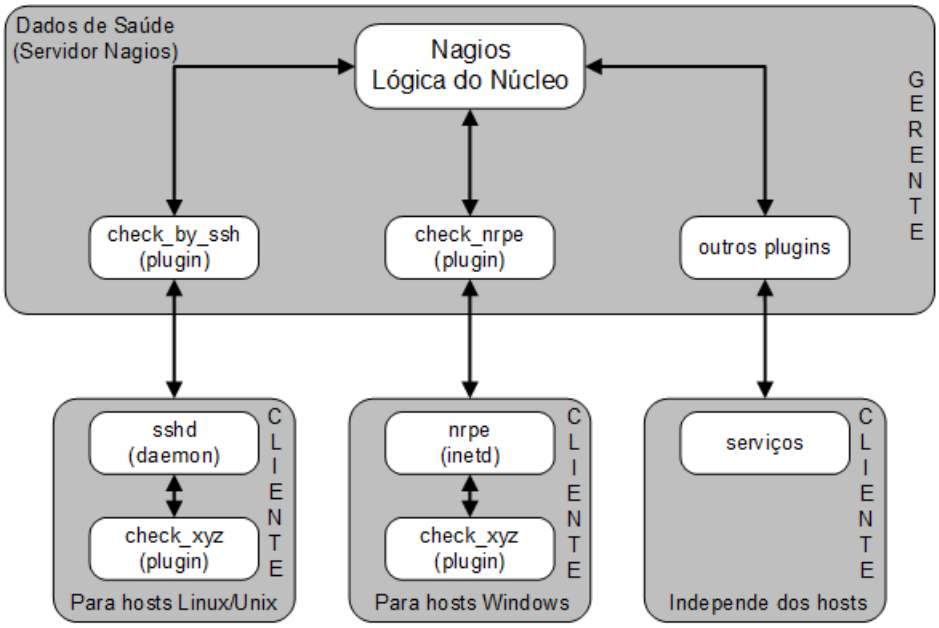


Figura 5.2: Tipos de monitoramento usados para verificar a saúde dos sistemas e serviços.

Dentre os hosts e serviços monitorados pelo Nagios, que vão além dos destacados na Figura 5.1, um histórico com mais de onze milhões de eventos foi gravado pelo Nagios em arquivos de logs, em um período de seis anos de monitoramento realizado no Grupo São Martinho S/A.

Realizou-se a análise de como esses arquivos de logs foram gerados, e identificou-se que esses logs foram gerados com base nas configurações do arquivo *nagios.cfg*, sendo habilitada a rotação do principal arquivo de log do Nagios (*nagios.log*). Este é o arquivo onde os eventos de serviços e hosts são registrados para propósito de histórico. A diretiva *log_rotation_method* define qual método de rotação de log deve ser usado pelo Nagios. Há quatro formas de rotacionamento (e.g. de hora em hora, diariamente, semanalmente e mensalmente). Dessa forma, identificou-se pelo arquivo de configuração *nagios.cfg* e pelos arquivos de logs gerados, e durante os seis anos de monitoramento, o arquivo *nagios.log* foi rotacionado diariamente. O diagrama de estado da Figura 5.3 foi construído após a análise do método de rotação usado pelo Nagios.

Como pode ser visto na Figura 5.3, no arquivo *nagios.log*, são primeiro registrados os estados iniciais de todos os sistemas, serviços e hosts monitorados. Também são registrados todas as alterações de estados. O Nagios recebe, no final do dia, às 23h59m um evento de rotação, nesse momento é tratado o rotacionamento diário do arquivo

nagios.log; em seguida, outros dois eventos são disparados: um para arquivar o log do dia anterior e outro para gerar um novo arquivo de log. Os arquivos de logs rotacionados são arquivados no diretório especificado na diretiva *log_archive_path* do arquivo *nagios.cfg*. Depois do Nagios ter gerado um novo arquivo de log, o processo de rotação dos arquivos é reiniciado.

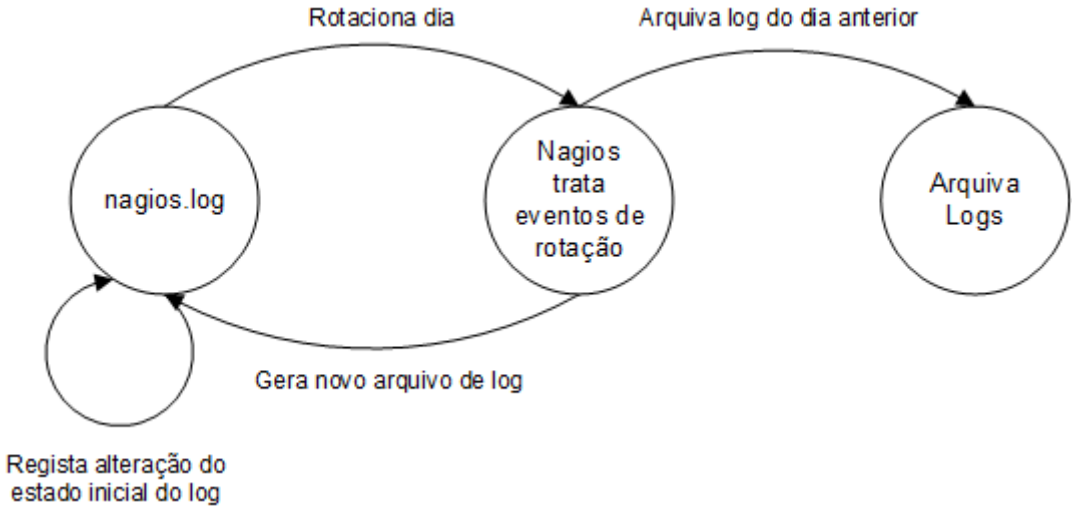


Figura 5.3: Diagrama de estado para o processo de rotação do *nagios.log*

Durante seis anos de monitoramento, obteve-se o número de 2.190 arquivos rotacionados, e cada arquivo com seu respectivo nome identificado pela data do monitoramento. O arquivo *nagios-12-29-2013-00.log* foi selecionado para exemplificar. Como pode ser visto, tem-se separado por hífen a palavra "nagios", o mês, o dia e o ano em que os eventos de monitoramento foram gerados no arquivo.

Arquivo de Log 5.1: Alguns eventos do arquivo de log *nagios-12-29-2013-00.log*

```
1 [1388196000] CURRENT HOST STATE: HOST01;UP;HARD;1;PING
     OK - Packet loss = 0%, RTA = 1.10 ms
2 [1388196000] CURRENT HOST STATE: HOST02;UP;HARD;1;PING
     OK - Packet loss = 0%, RTA = 1.39 ms
3 [1388196000] CURRENT HOST STATE: HOST03;DOWN;HARD;1;
     CRITICAL - Host Unreachable (172.16.5.150)
4 [1388196000] CURRENT SERVICE STATE: HOST04;CPU Load;OK;
     HARD;1;OK - CPU 'Load Percentage' = 6%
5 [1388196000] CURRENT SERVICE STATE: HOST05;Memory Usage;
     OK;HARD;1;OK - Memory 'Utilization' = 34%:
6 [1388196000] CURRENT SERVICE STATE: HOST06;Swap Usage;OK
     ;HARD;1;OK - Swap 'Pages/sec' = 41
7 ...
8 [1388210751] SERVICE FLAPPING ALERT: HOST14;Active
     Directory;STOPPED; Service appears to have stopped
     flapping (4.9% change < 5.0% threshold)
```

```
9  [1388233518] EXTERNAL COMMAND: SCHEDULE_HOST_DOWNTIME;
     HOST08;1388233507;1391178307;1;0;7200;renato.santos;
     Periodo de Entressafra
10 [1388243491] SERVICE ALERT: HOST10;Swap Usage;WARNING;
     SOFT;1;Warning - Swap 'Pages/sec' = 4096
11 [1388229011] SERVICE ALERT: HOST12;CPU Load;CRITICAL;
     SOFT;1;Critical - CPU0 'Load Percentage' = 91:
12 [1388226521] SERVICE ALERT: HOST11;Swap Usage;CRITICAL;
     SOFT;1;Critical - Swap 'Pages/sec' = 5400
13 [1388210911] SERVICE NOTIFICATION: IT-Monitor;HOST13;
     SMTP_8025;OK;notify-service-by-email;SMTP OK - 6.982
     sec. response time
14 [1388232941] HOST ALERT: HOST09;DOWN;SOFT;1;CRITICAL -
     Host Unreachable (172.16.5.46)
15 [1388233971] HOST NOTIFICATION: IT-Monitor;HOST07;UP;
     notify-host-by-email;PING OK - Packet loss = 0%, RTA
     = 0.62 ms
16 ...
```

A cada mudança de status verificada nas checagem realizadas por plugins, uma série de eventos é disparada pelos Nagios, como pode ser visto no Arquivo de Log 5.1. Alguns desses eventos são: CURRENT HOST STATE, CURRENT SERVICE STATE, SERVICE FLAPPING ALERT, EXTERNAL COMMAND, SERVICE ALERT, SERVICE NOTIFICATION, HOST ALERT e HOST NOTIFICATION. Cada evento vem acompanhado da data e da hora no formato *Unix*, seguido pelo nome do evento, o nome do host, o nome do serviço, nos casos de eventos de serviços, o estado do evento, dentre outros dados de desempenho.

Analisaram-se os comportamentos da ferramenta de monitoramento Nagios, bem como são registrados os históricos de saúde dos hosts e serviços que foram monitorados. Com isso, decisões consistentes foram tomadas no que diz respeito à seleção dos dados que foram processados pelo módulo KKD da arquitetura proposta.

5.2 Arquitetura do módulo KDD

A primeira etapa do processo KDD é a seleção de dados. As análises realizadas na seção anterior forneceram os parâmetros adequados para realizar a seleção dos dados no módulo KDD. Com os dados relevantes em mãos, as próximas etapas do processo de extração do conhecimento são as de pré-processamento e transformação dos dados. Essas etapas serão implementadas segundo o algoritmo de mineração proposto, com o objetivo de utilizar algoritmos probabilísticos Bayesianos. Para o módulo

de inferência, foi necessário adaptar os dados, o que levou a restrições impostas pelo módulo de inferência, tais como: se o conjunto de dados possuir atributos numéricos, eles devem ser discretizados; com isso, focou-se a pesquisa em variáveis categóricas que são adequadas ao processamento dos algoritmos Bayesianos.

Selecionado o conjunto de logs do Nagios relevantes ao processo de extração dos dados, iniciou-se a proposição de um algoritmo para pré-processamento, com o objetivo de criar um *dataset* (Conjunto de dados) para cada evento registrado pelo Nagios, eliminando inconsistências, reduzindo a dimensão e a complexidade do conjunto de dados. Foi proposto um algoritmo que denominou-se MAPLOG (Matching and Preprocessing Logs), e sua implementação pode ser vista no Apêndice B.

O algoritmo MAPLOG foi proposto com o objetivo de realizar o pré-processamento de dados históricos do Nagios que corresponde ao monitoramento de hosts e serviços, porém seu uso pode ser aplicado a outros conjuntos de dados. O MAPLOG tem como principal característica identificar padrões de texto em arquivos de log e realizar o devido processamento. O Algoritmo 5.3 do MAPLOG está dividido em três partes: declaração e inicialização das variáveis, iteração principal, onde ocorre o processamento, e a saída, onde o pós-processamento é realizado. Implementou-se o algoritmo utilizando a linguagem *Python*.

A fim de buscar padrões de caracteres nos arquivos de logs, expressões regulares é a melhor ferramenta para esta tarefa. Sabendo disso, para cada evento Nagios que se deseja selecionar, foi criada uma expressão regular, como apresentado no Código 5.2.

Código 5.2: Expressão regular para identificar o evento "HOST ALERT"

```
^.(\w+).\s([\w\s]+)\W\s([\w\s\_\.\-]+)\W(\w+)\W(\w+)\W([1-5])\W(.*)$
```

O pseudocódigo do Algoritmo 5.3 tem seu código implementado no apêndice B, onde, no Código B.1, ocorre a declaração e a inicialização das variáveis. Utiliza-se o módulo de função "re" que possui várias funções para manipulação de expressões regulares, atribuindo-se a expressão regular com o uso do método *compile()*, dessa forma instancia-se cada variável com um objeto que contenha sua respectiva expressão regular.

Na segunda parte do algoritmo MAPLOG, onde ocorre a iteração principal, é realizado o processamento dos arquivos de logs, e dado um diretório que contenha os arquivos de logs, a cada arquivo lido iterativamente, e em cada linha do respectivo arquivo lido, as expressões regulares são casadas com um eventos que a corresponde,

retornando dessa forma uma tupla, que é processada pela função desenvolvida denominada *tupla_to_list*. O Código B.2 pode ser visto no Apêndice B. A função processa os dados e transforma o campo *timestamp* do Unix para um formato com data e hora legíveis, e em seguida retornam todos os elementos da tupla como uma *string* para posterior inclusão em uma lista correspondente ao evento. No final do processo iterativo, obtêm-se dez listas, cada uma contendo os respectivos eventos de monitoramento do Nagios. A Figura 5.4 ilustra as entradas e saídas do algoritmo MAPLOG, e o MAPLOG faz a leitura de arquivos de log em um determinado diretório, processa esses arquivos e, como saída, gera novos arquivos agrupados por eventos de monitoramento do Nagios.

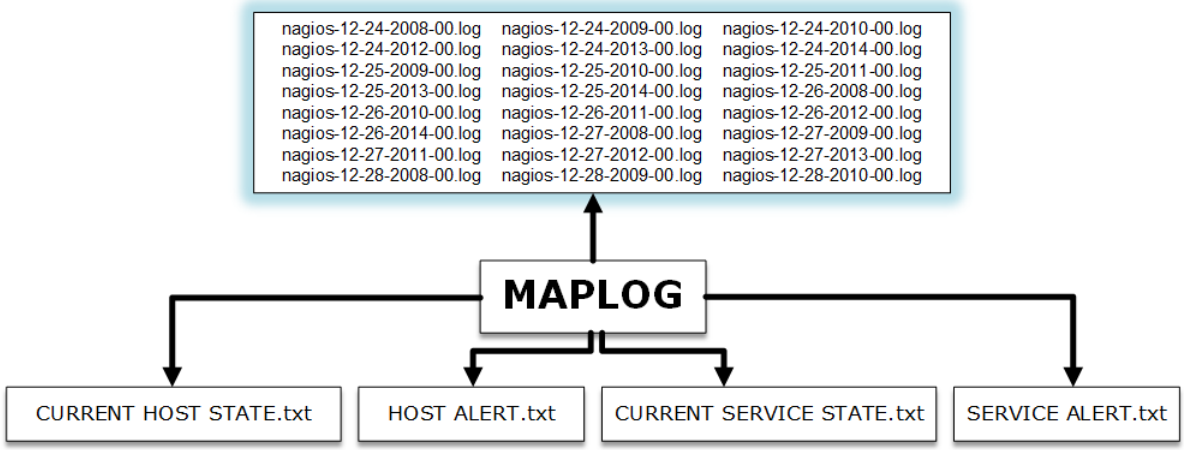


Figura 5.4: Diagrama para ilustração de entradas e saídas do algoritmo MAPLOG.

No código B.3 do Apêndice B, é apresentada a implementação do MAPLOG, algoritmo 5.3, onde cada vetor contém seus respectivos eventos de monitoramento do Nagios. Para cada vetor, são realizadas iterações, e em cada índice do vetor o respectivo evento é escrito no arquivo, ou seja, para um vetor "host alert", a cada índice do vetor um evento "HOST ALERT" é escrito no arquivo "HOST ALERT.txt". Isso ocorre até à ultima iteração do respectivo vetor. Ao terminar o processamento, o algoritmo MAPLOG gerou como saída os seguintes arquivos: CURRENT HOST STATE, CURRENT SERVICE STATE, HOST ALERT, HOST DOWNTIME ALERT, HOST FLAPPING ALERT, HOST NOTIFICATION, SERVICE ALERT, SERVICE NOTIFICATION, SERVICE DOWNTIME ALERT, SERVICE FLAPPING ALERT.

Algoritmo 5.3: Pseudocódigo do Algoritmo MAPLOG

```
INICIALIZAÇÃO:
1. Inicializa vetores com expressões regulares

ITERAÇÃO PRINCIPAL:
```

```
2. Para cada arquivo do diretório faça
  2.1 Para cada linha do arquivo faça
    2.2.1 Agrupa eventos utilizando o procedimento ATRIBUI;

ATRIBUI:
  Se vetor.regex(linha) diferente de [] então
    Se o evento for identificado então
      Formata os dados e os atribua a um vetor do respectivo evento;

SAÍDA:
3. Para cada índice no vetor de um respectivo evento faça
  3.1 Escreve linha no arquivo do respectivo evento;
```

Os arquivos gerados pelo MAPLOG foi o produto da execução direta ou indireta de atividades como limpeza, agrupamento e redução dos dados, reduzindo assim a complexidade computacional associada ao problema, ou seja, permitindo realizar uma manipulação mais eficiente dos dados. Após o pré-processamento, os dados ainda não estão apropriados para servirem de insumo aos algoritmos Bayesianos do módulo de inferência. Para adequar os dados aos algoritmos que permitam a construção de um modelo de inferência Bayesiano, foi proposto o algoritmo TOTRANS.

O algoritmo TOTRANS (Transpose To Transform) tem como objetivo estruturar os dados para a obtenção de uma tabela única, onde os campos representam os atributos e os dados o *dataset*. O *dataset* é um conjunto de todos os dados que serão utilizados para a construção da rede Bayesiana. Para esse processo de transformação, foram selecionados os servidores Net Server (Serviços de rede), Print Server (Serviço de impressão), SAP GRC NF-e (Sistema de nota fiscal eletrônica), SAP HA Cen. Inst. (Instância central do sistema ERP), SAP App Server 1 (Servidor de aplicação do sistema ERP), SAP App Server 2 (Servidor de aplicação do sistema ERP), em que, para cada servidor após a execução do TOTRANS, foi obtido um *dataset* com seus respectivos atributos.

Os arquivos gerados pelo MAPLOG são manipulados a fim de obter três novos arquivos para cada um dos servidores citados no parágrafo anterior. Os arquivos foram gerados no formato *.csv, sendo que, para o arquivo HCA (Host Current Alert), selecionaram-se três variáveis (Data, Hora e Estado do host), agrupando assim os dados dos arquivos “CURRENT HOST STATE.txt” e “HOST ALERT.txt”. Para o arquivo SCA (Service Current Alert), selecionaram-se apenas quatro variáveis (Data, Hora, Serviço e Estado do serviço), agrupando os dados dos arquivos “CURRENT SERVICE STATE.txt” e “SERVICE ALERT.txt”; e para o arquivo HSTE (Host Service Time Event), somente duas variáveis foram selecionadas (Data e Hora). Neste caso, os dados deste

arquivo contêm a data e o horário de todos os eventos dos arquivos HCA e SCA. Para uma execução adequada do algoritmo TOTRANS, é importante que o arquivo HSTE não tenha dados duplicados e que todos os arquivos estejam ordenados por data e hora.

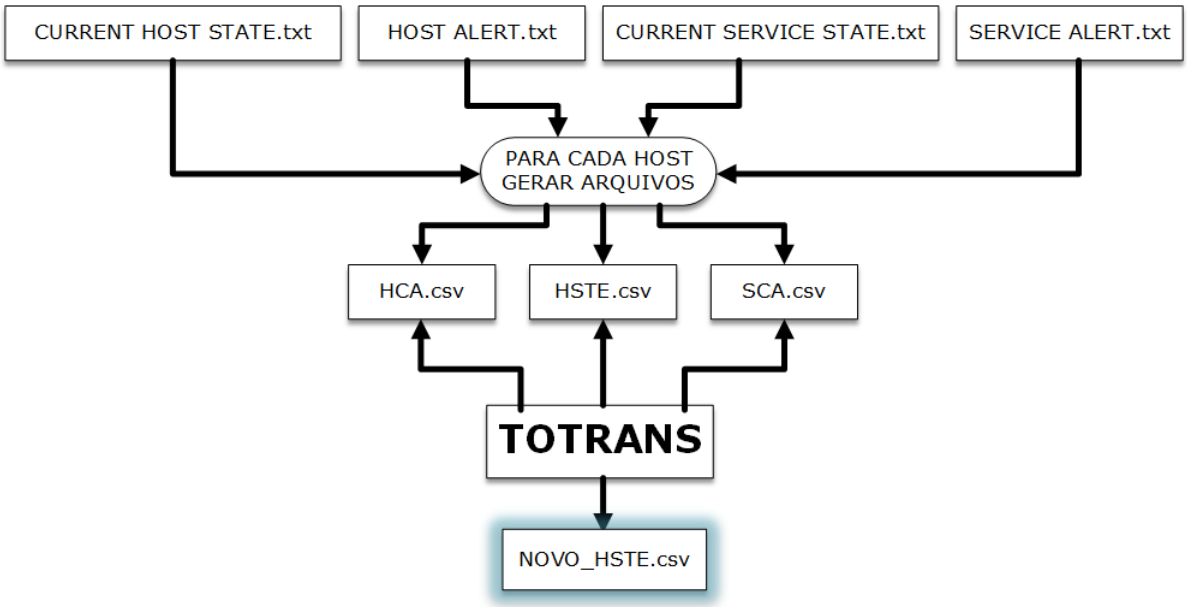


Figura 5.5: Diagrama para ilustração de entradas e saídas do algoritmo TOTRANS.

No código C.2 do Apêndice C, é apresentada a implementação do TOTRANS, algoritmo 5.4, onde o TOTRANS lê os arquivos HCA, SCA e HSTE e realiza a transposição matricial dos dados para o arquivo HSTE, de modo a realizar uma discretização contínua dos eventos de monitoramento dada uma data e hora do arquivo HSTE. Na iteração principal do algoritmo, para cada linha lida no arquivo HSTE, com variáveis data e hora, pode-se ter o um evento de alertar para host, e um ou mais eventos de alerta para os serviços. Esses eventos são transpostos para a respectiva linha do arquivo HSTE, através do procedimento ATRIBUI-1 e ATRIBUI-2, para iterações realizadas no arquivo HCA e SCA, como pode ser visto, respectivamente, no item 2.2 e 2.4 do Algoritmo 5.4.

Algoritmo 5.4: Pseudocódigo do Algoritmo TOTRANS

```
INICIALIZAÇÃO:
1. Carrega os arquivos para memória;

ITERAÇÃO PRINCIPAL:
2. Para cada linha do arquivo HSTE faça
  2.1 Recomeça o laço no arquivo HCA
  2.2 Para cada linha em HCA faça
    2.2.1 Transpõe estados dos eventos com o procedimento ATRIBUI-1;
```

```
  2.3 Recomeça o laço no arquivo SCA
  2.4 Para cada linha em SCA faça
    2.4.1 Transpõe estados dos eventos com o procedimento ATRIBUI-2;

ATRIBUI-1:
Se data e hora da linha do arquivo HCA éigual a data e hora da linha
   do arquivo HSTE então
  Atribui estado do host na célula C_ij, do arquivo HSTE;

ATRIBUI-2:
Se data, hora e serviço da linha do arquivo SCA éigual data, hora do
   arquivo HSTE, mais o serviço que se deseja transpor o estado faça
  Atribui estado do serviço na célula C_ij, do arquivo HSTE;

SAÍDA:
3. Para cada linha no arquivo HSTE faça
3.1 Escreve linha no arquivo NOVO_HSTE;
```

Como resultado da execução do algoritmo TOTRANS, é gerado o *dataset* denominado NOVO_HSTE.csv. Na Tabela 5.1, é possível observar os atributos e uma pequena amostra de dados gerados para o *Print Server* (Servidor de impressão), após a execução do algoritmo TOTRANS.

O TOTRANS permitiu, com a transposição dos dados, agrupar os estados do host e serviços em uma única linha, dada uma determinada data e hora de ocorrência dos eventos de monitoramento. Com isso, consegue-se observar a relação da mudança de estados dos hosts e serviços ao longo do tempo. Isso fica claro quando se olha para a Tabela 5.1 e observa-se que no dia 18-07-2014, às 17h46m09s, o *Print Server* mudou seu estado para *DOWN*, ou seja, o servidor ficou indisponível. Isso implica que todos os demais serviços e recursos monitorados também estarão indisponíveis com o estado *CRITICAL*.

Tabela 5.1: *Dataset* "NOVO_HSTE.csv" gerado pelo algoritmo TOTRANS.

Data	Hora	Estado do Host	Carga da CPU	Uso de RAM	Uso de Disco	Uso de Swap	Serviço de Impressão
18/07/2014	17:46:09	DOWN	CRITICAL	CRITICAL	CRITICAL	CRITICAL	CRITICAL
18/07/2014	18:23:29	UP	CRITICAL	CRITICAL	CRITICAL	CRITICAL	CRITICAL
18/07/2014	18:23:39	DOWN	CRITICAL	CRITICAL	CRITICAL	CRITICAL	CRITICAL
18/07/2014	18:23:49	UP	CRITICAL	CRITICAL	CRITICAL	CRITICAL	OK
18/07/2014	18:24:29	UP	CRITICAL	CRITICAL	OK	CRITICAL	OK
18/07/2014	18:24:49	UP	CRITICAL	CRITICAL	OK	OK	OK
18/07/2014	18:25:39	UP	OK	CRITICAL	OK	OK	OK
18/07/2014	18:26:09	UP	OK	OK	OK	OK	OK

Os *datasets* gerados após o processamento do algoritmo TOTRANS serão o insumo para a construção da rede Bayesiana do módulo de inferência, pois agora os dados estão apresentando dependências funcionais transitivas, não há variáveis omitidas e não há entradas em branco no *dataset*, tornando-o adequado para a utilização dos algoritmos Bayesianos.

5.3 Arquitetura do módulo de inferência

O módulo de inferência resume-se basicamente em um DAG (Grafo Acíclico Dirigido), em que, para cada variável A que possui pais B_1, ..., B_n, existe uma tabela de probabilidade condicional $P(A|B_1, ..., B_n)$, mas para a construção desse modelo, foi proposta uma arquitetura, sendo as etapas sequenciais realizadas para a construção e a validação do modelo gráfico probabilístico (Rede Bayesiana). O objeto principal da arquitetura deste módulo é gerar, para cada servidor que hospeda determinado sistema e serviço, uma rede Bayesiana para realização de inferência na saúde dos sistemas.

Primeiramente, para cada servidor, foi gerado um *dataset* no módulo KDD, e os conjuntos de dados gerados serão o insumo para os algoritmos de aprendizagem estrutural e paramétrica da rede Bayesiana. Dois algoritmos são propostos: para aprendizagem estrutural, será utilizado o algoritmo PC, e para aprendizagem paramétrica, o algoritmo EM.

Na etapa de aprendizagem estrutural, a construção da rede Bayesiana é semi-automática, conhecida também como "baseada em dados", e busca identificar a estrutura do modelo através da aplicação de algoritmos. O objetivo da construção baseada em dados é identificar um modelo que fique adequado à distribuição de probabilidade dos dados. Essa técnica surgiu pela necessidade de incluir uma grande quantidade de variáveis nos modelos, dificultando assim a construção manual, pois mesmo especialistas de determinada área têm dificuldades de expressar a relação das variáveis de forma satisfatória. Por isso, foi escolhido o algoritmo PC, porque, além da construção do modelo ser baseada em dados, é possível também direcionar conexões do DAG com base em conhecimentos causais de um especialista.

Como não foi implementada nenhuma mudança no algoritmo PC, um diagrama conceitual é apresentado na Figura 5.6, a fim de demonstrar o funcionamento do algoritmo PC, supondo-se que o módulo KDD forneça um *dataset* com cinco atributos. A entrada para o algoritmo PC é um grafo completo não direcionado contendo to-

das as variáveis de V (Conjunto de variáveis dado um *dataset*), que é representado pelo primeiro grafo da Figura 5.6. Após testes incondicionais que testam as relações de independência condicional variáveis, o segundo grafo da Figura 5.6 é gerado. Em seguida, o terceiro e o quarto grafos são gerados após testes condicionados a uma e duas variáveis, respectivamente. No quinto, um PDAG (Acyclic Partically Directed Graphs), ou seja, um grafo acíclico, parcialmente dirigido, é construído, para finalmente, no sexto grafo, com base ou não em conhecimentos causais de um especialista, o possível DAG é gerado.

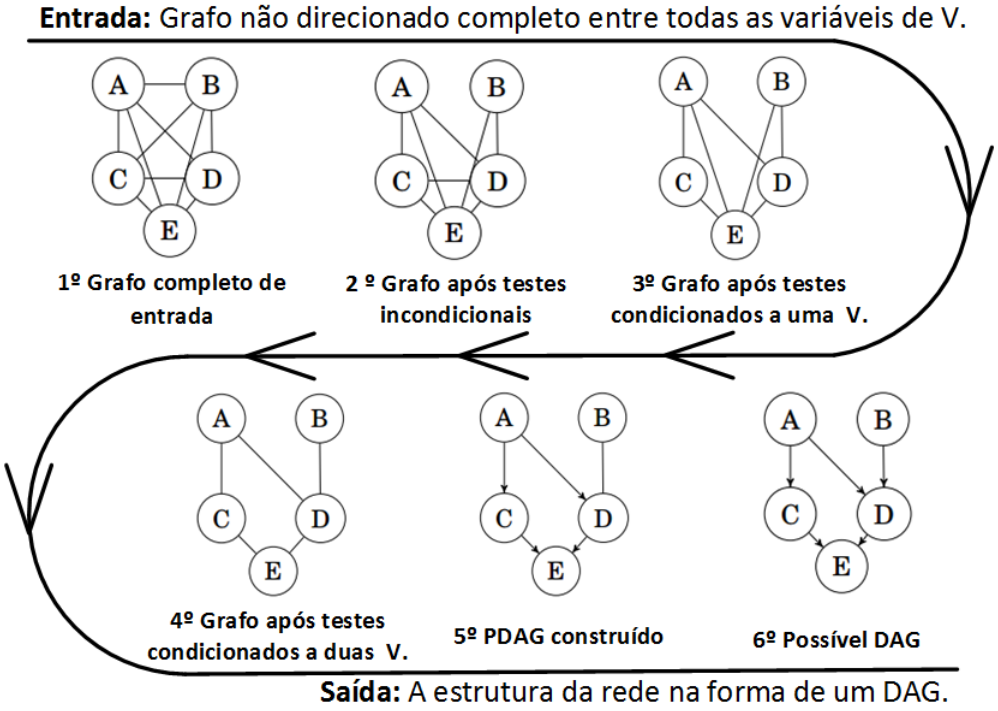


Figura 5.6: Ilustração conceitual para Algoritmo PC

Para gerar as redes Bayesianas do módulo de inferência, utiliza-se o algoritmo PC implementado na ferramenta GeNIe, desenvolvida pelo laboratório de Sistemas de Decisão da Universidade de Pittsburg. GeNie é uma interface gráfica de usuário para o SMILE (Structural Modeling, Inference, and Learning Engine), um motor de raciocínio para modelos gráficos, tais como redes Bayesianas, diagramas de influência e modelos de equação estrutural.

Para cada servidor após a execução do módulo KDD, foi obtido um *dataset*, e para demonstrar a etapa de aprendizagem estrutural da rede Bayesiana utilizando o algoritmo PC da ferramenta GeNie, selecionou-se então o *dataset* do *Print Server*. Primeiramente, ao carregar o *dataset* na ferramenta e selecionar o algoritmo PC para execução, foi retirada a seleção dos atributos data e hora antes da execução do algoritmo, dei-

xando apenas os que correspondam aos estados dos serviços e do host. Desse modo, o grafo de entrada para o algoritmo PC é representado pelo primeiro grafo da Figura 5.7. Após a execução do algoritmo um segundo grafo é obtido, mais especificamente um PDAG, como demonstrado na Figura 5.7. Em seguida, é então criado um possível DAG FINAL, assumindo como padrão o PDAG obtido para o *dataset* selecionado.

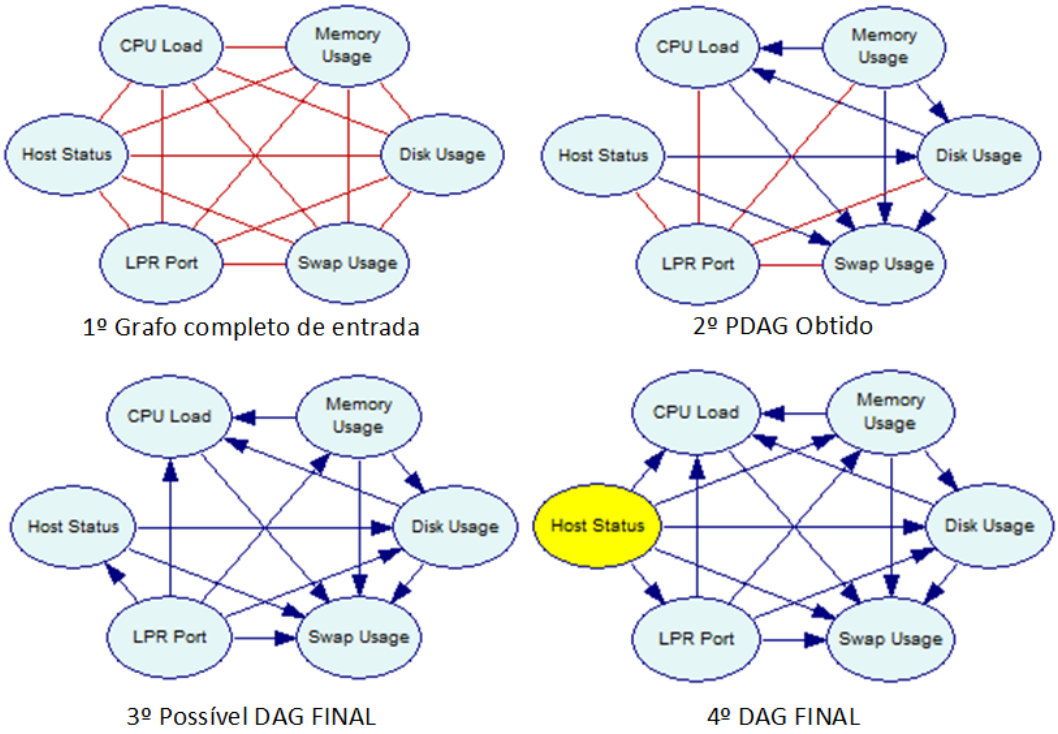


Figura 5.7: Algoritmo PC para o *dataset* do servidor *Print Server*.

A fim de validar o possível DAG FINAL, foi apresentado a um especialista em computação em nuvem e sistemas distribuídos o terceiro grafo da Figura 5.7. O objetivo foi validar, através de conhecimentos causais, as relações de dependência e independência condicional das variáveis. Com o apoio do especialista, identificou-se que a variável *host status* que expressa se o *host* está disponível ou indisponível torna-o uma variável condicionalmente independente de todas as outras; porém, todas as variáveis são condicionalmente dependentes da variável *Host Status*, pois dado um estado de indisponibilidade para o *host* em questão, todos os seus recursos também ficaram indisponíveis. Através das constatações do especialista foram feitos ajustes que resultaram no quarto e último grafo da Figura 5.7. Esse é então o DAG FINAL para o servidor *Print Server* proposto ao módulo de inferência.

A etapa final da modelagem das rede Bayesianas é a que estima os parâmetros das variáveis selecionadas, gerando assim as TPCs (Tabelas de Probabilidades Condicio-

nais), que quantificam os efeitos que nós pais têm sobre os nós filhos. Definir cada uma das probabilidades condicionais das TPCs, dado o número de variáveis, pode tornar-se uma tarefa complicada se apenas for usada a experiência de um especialista. Particularmente, opta-se em utilizar o algoritmo EM (Expectation Maximization) implementado na ferramenta GeNIe, a fim de se gerar as TCPs para os nós dos DAGs das redes Bayesianas propostas para o módulo de inferência. Como não foi implementada nenhuma mudança no algoritmo EM, apenas é descrito o funcionamento do algoritmo.

O algoritmo EM é particularmente útil nessa arquitetura, dado que já se tem a estrutura da rede Bayesiana e apenas se quer apreender os parâmetros, as distribuições de probabilidade condicional para a estrutura definida. O EM é iterativo, e a cada iteração são realizados dois passos, conhecidos como E (Expectation) e M (Maximization). Assim, se a rede Bayesiana possui n nós, o vetor de parâmetro da rede θ pode ser decomposto em $\theta = (\theta_1, ..., \theta_n)$, em que θ_i representa o conjunto de parâmetros da distribuição de probabilidades condicionais associadas a cada nó da rede. Deste modo, a definição genérica do EM pode ser dada como se segue:

1. No passo E, são estimadas as probabilidades a *posteriori* de cada nó da rede, dado por $P(z|x, \theta^{(t)})$.

2. No passo M o parâmetro θ é utilizado e calculado através da maximização da função Q de log-verossimilhança definida em 5.1, utilizando-se as estimativas da etapa anterior:

$$Q(\theta, \theta^{(t)}) = E[logP(Z|\theta)|x, \theta^{(t)}] \tag{5.1}$$

Em que X é o conjunto de dados observados; Z é o conjuntos de dados completos; bem como x e z são, respectivamente, observações desses conjuntos, e $\theta^{(t)}$ é o vetor de parâmetros estimados na iteração t. A cada iteração, ocorre um aumento do logaritmo de verossimilhança que, no final, convergirá para um máximo local da função.

Como os dados dos *datasets* gerados para cada servidor são completos e não contêm variáveis ocultas, nesse caso, a distribuição de probabilidades das variáveis é obtida pelo EM calculando-se as estatísticas suficientes para os dados. Após a execução do EM, tabelas de probabilidades condicionais foram geradas para os nós da rede Bayesiana. A Figura 5.8 demonstra as TCPs geradas para os nós *Host Status*, *Memory Usage* e *LPR Port* do servidor *Print Server* que disponibiliza serviço de impressão para os usuários. Como pode ser visto na figura, o nó *Host Status* não é condicionalmente dependente de nenhum outro nó, implicando que a TPC terá apenas as probabilidades

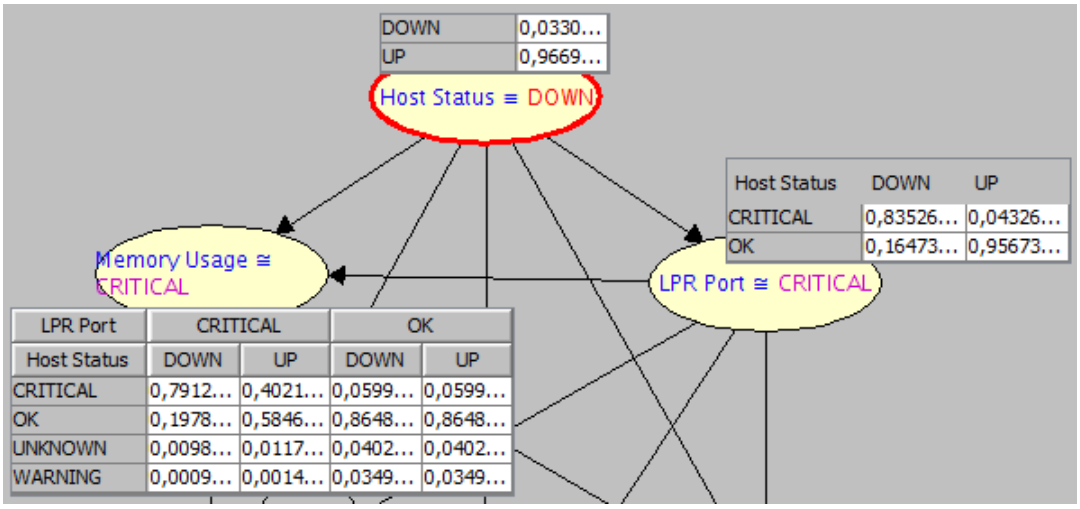


Figura 5.8: TCPs dos nós da rede Bayesiana *Print Server* geradas com o Algoritmo EM.

a priori dos estados deste nó, mas o nó *LPR Port* é condicionalmente dependente do nó *Host Status*, e o nó *Memory Usage* é condicionalmente dependente dos nós *Host Status* e *LPR Port*. Isso implica que a distribuição de probabilidades condicionais para as TPCs destes nós é ponderada, dada a relação causal entre as variáveis aleatórias, e os parâmetros (distribuição de probabilidades) para TCPs destes nós foram estimados com o algoritmo EM, utilizando-se como base do *dataset* gerado pelo módulo KDD.

Após a realização das etapas de aprendizagem estrutural e aprendizagem paramétrica, faz-se necessário validar as redes Bayesianas. Usou-se o método de validação cruzada, denominado *k-fold*, para evitar superajustes (*overfitting*) decorrentes do processo de aprendizagem, bem como a inconsistência na validação das redes Bayesianas geradas. Utilizou-se da validação cruzada com 4 subconjuntos (*folds*), e para cada subconjunto a avaliação é realizada com um número de métodos complementares.

Para cada subconjunto, são calculados: a *precisão total* da rede Bayesiana, que é calculada em função do número de predições corretas da variável observada; o *coeficiente de correlação de Pearson* (R), em que é medido o grau da correlação (e a direção dessa correlação - se positiva ou negativa) entre as variáveis.

Para o *coeficiente de correlação de Pearson*, os resultados podem ser interpretados na escala abaixo:

- 0.90 a 1.00 positivo ou negativo indica uma correlação muito forte.
- 0.70 a 0.90 positivo ou negativo indica uma correlação forte.
- 0.50 a 0.70 positivo ou negativo indica uma correlação moderada.

- 0.30 a 0.50 positivo ou negativo indica uma correlação fraca.
- 0.00 a 0.30 positivo ou negativo indica uma correlação desprezível.

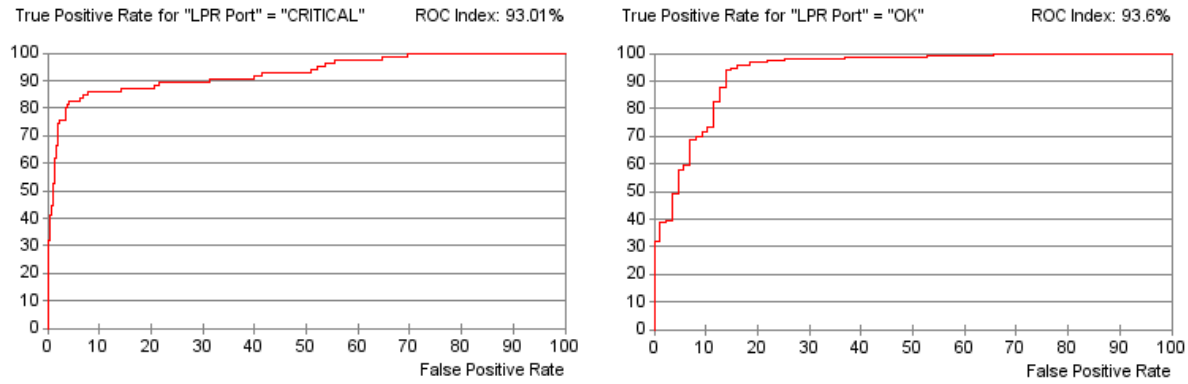
São calculados também, para cada subconjunto: o coeficiente de determinação (R^2) é uma medida de ajustamento de um modelo estatístico linear generalizado, que varia entre 0 e 1, indicando, em percentagem, o quanto o modelo consegue explicar os valores observados, e sendo mais explicativo, melhor ele se ajusta à amostra do subconjunto de dados; a *média relativa do índice de Gini* é usada como medida de dispersão das variáveis dos subconjuntos de amostra, em que um coeficiente de Gini zero expressa igualdade perfeita, e um coeficiente de Gini 1 (ou 100%) expressa a desigualdade máxima entre os valores; *média relativa do índice lift* reflete a confiança do resultado de classificação; e a *média do índice ROC* é calculada com base nos índices ROC (Receiver Operating Characteristic) de cada classe do nó observado. Essa média expressa a precisão, ou seja, a probabilidade em que o classificador irá classificar para o nó observado, um valor positivo escolhido aleatoriamente, maior do que um valor negativo escolhido aleatoriamente.

Para classificar a precisão de um teste de diagnóstico, a escala tradicional de pontuação para a *média do índice ROC*:

- 0.90 a 1.00 = Excelente
- 0.80 a 0.90 = Bom
- 0.70 a 0.80 = Justo
- 0.60 a 0.70 = Pobre
- 0.50 a 0.60 = Falho

Os resultados para índice ROC de um *fold*, na validação cruzada, são demonstrados nos gráficos da Figura 5.9. As curvas ROC descrevem a capacidade discriminativa de um teste diagnóstico. Isto permite pôr em evidência os valores para os quais existe maior otimização da sensibilidade em função da especificidade. O ponto, numa curva ROC, onde isto acontece, é aquele que se encontra mais próximo do canto superior esquerdo do gráfico.

O índice ROC é calculado de acordo com a curva que é apresentada na parte superior do gráfico. Ele representa a superfície abaixo da curva ROC, dividida pela metade



(a) Curva ROC para classe "CRITICAL" **(b) Curva ROC para classe "OK"**

Figura 5.9: Curvas ROC para o nó observador "LPR Port" da rede Bayesiana *Print Server*.

da superfície total. Com base nos índices ROC das classes dos nós observados, é obtida a *média do índice ROC*. Para o *fold* em que os gráficos da Figura 5.9 foram gerados, tem-se para o gráfico da Figura 5.9a, um ROC índex de 93,01%, e para o gráfico da Figura 5.9b, um ROC index de 93,6%. Dessa forma, a *média do índice ROC* para este *fold* é: 93,01 + 93,6 / 2 = 93,3%. O resultado para a *média do índice ROC* de 93,3% indica uma precisão de diagnóstico *Excelente* para o nó LPR Port da rede Bayesiana Print Server.

Para cada *fold*, são apresentados resultados a todos os métodos de avaliação citados, e esses resultados são sintetizados para cada rede Bayesiana. Os resultados para as avaliações das redes Bayesianas são apresentados no próximo capítulo.

Após a validação, as redes Bayesianas são incluídas no sistema de suporte à decisão para realização de inferências e apoio nas decisões de provisionamento dinâmico de recursos computacionais, e em caso de falhas ou altas demandas de recursos computacionais, a fim de garantir a estabilidade dos sistemas e serviços na nuvem.

Capítulo 6

RESULTADOS EXPERIMENTAIS

Nesse capítulo, abordaremos os resultados do processamento dos dados, bem como a criação do módulo de inferência e o resultado principal da criação da rede bayesiana que representa a saúde do sistema. Inicia-se o processo, pela obtenção dos dados e aplicação da metodologia para descoberta de conhecimento, descrita anteriormente. Essa metodologia propicia selecionar os principais atributos, eliminando os registros inúteis ou com problemas nos dados. Para as atividade de descoberta do conhecimento, usando arquivos de *log* gerados através da ferramenta de monitoramento Nagios, os algoritmos MAPLOG E TOTRANS foram implementados na linguagem de programação *python*, como pode ser visto nos apêndices B e C, respectivamente.

O MAPLOG processou 12.412.259 linhas de eventos de monitoramento do Nagios e eliminou 1.080.339 linhas de registros inúteis ou com problemas nos dados. Além disso, efetuou o pré-processamento dos eventos de monitoramento do Nagios que foram relevante ao domínio de nossa aplicação. Na etapa de redução e de projeção dos dados, o TOTRANS processou 11.331.920 linhas de eventos de monitoramento do Nagios. E através de métodos de transformação para reduzir o número efetivo de atributos dos *datasets*, reduziu para 79.669 o número de linhas de dados e consolidou os 6 *datasets* gerados. A Tabela 6.1 apresenta uma visão dos tamanhos dos *dataset*.

Tabela 6.1: Datasets gerados após processamento TOTRANS.

Dataset	Total de linhas
Print Server	6.282
Net Server	5.410
SAPPRD AS S1	3.971
SAPPRD AS S2	3.886
SAPPRD HA CI	58.508
SAPPRD GRC NF-e	1.642

Para validar e avaliar a frequência dos sintomas de falhas nos serviços e sistemas presentes nos *datasets*, é preciso conhecer como esses dados estão organizados e quais são as relações entre os diferentes atributos. Portanto, para a análise dos *datasets*, utilizou-se da linguagem estatística R (CHANG, 2012), a fim de realizar uma análise exploratória dos dados. Esta análise, condensada em 5 histogramas, foi repassada ao especialista e administrador de sistema que avaliar se os padrões gerados, após a transformação dos dados, eram representativos da frequência da ocorrência de alertas dos serviços e sistemas, e se são justificadas as suas causas.

Para o gráfico da Figura 6.1, nota-se a proeminencia da classe *down* nos servidores *Print Server* e *SAPPRD HA CI* do *dataset*. Estes possuem uma frequência maior do alerta, sendo que a classe *down* indica indisponibilidade. Segundo a análise do especialista, problemas no dimensionamento correto dos recursos computacionais, bem como configuração inadequada são causas de paradas frequentes de manutenção corretiva. Além disso, tratam-se de servidores que possuem sistemas e serviços com maior demanda dos usuários da empresa. Portanto, falhas acarretam a reinicialização dos servidores e posterior realização de manutenções.

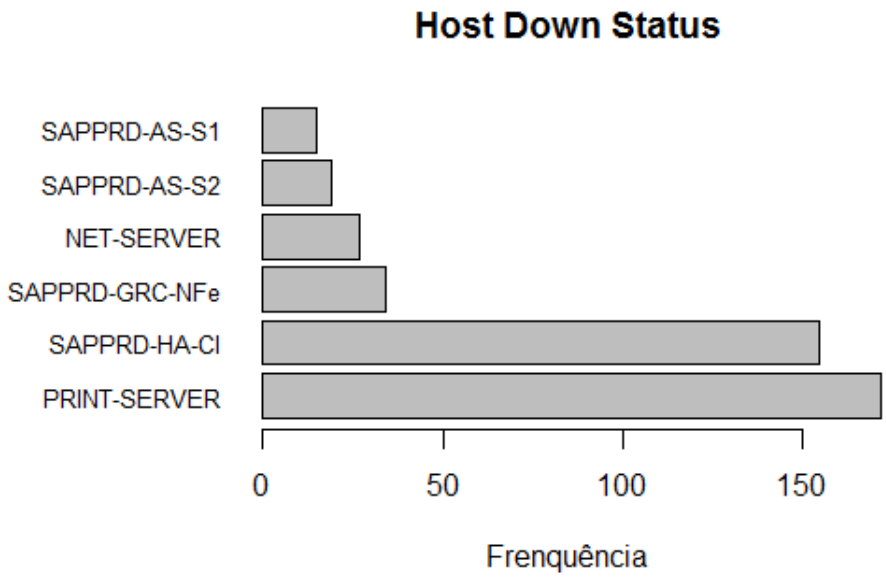


Figura 6.1: Frequência da classe *DOWN* para o atributo estado do host.

Observando-se a sazonalidade (em termos de dias da semana) da ocorrência dos eventos de alertas para os servidores, é possível identificar que em determinados períodos, há baixa demanda dos serviços, e consequentemente baixa quantidade de falhas. Em outros dias, existe um aumento na demanda dos serviços, levando assim a um con-

sumo maior dos recursos computacionais, tais como memória, CPU e disco. Desse modo, fica evidente o problema de dimensionamento dos recursos computacionais para os servidores. O gráfico da Figura 6.2 compara os alertas de recursos computacionais de dois servidores NetServer e Print Server para cada dia da semana, no período de seis anos de monitoramento do Nagios.

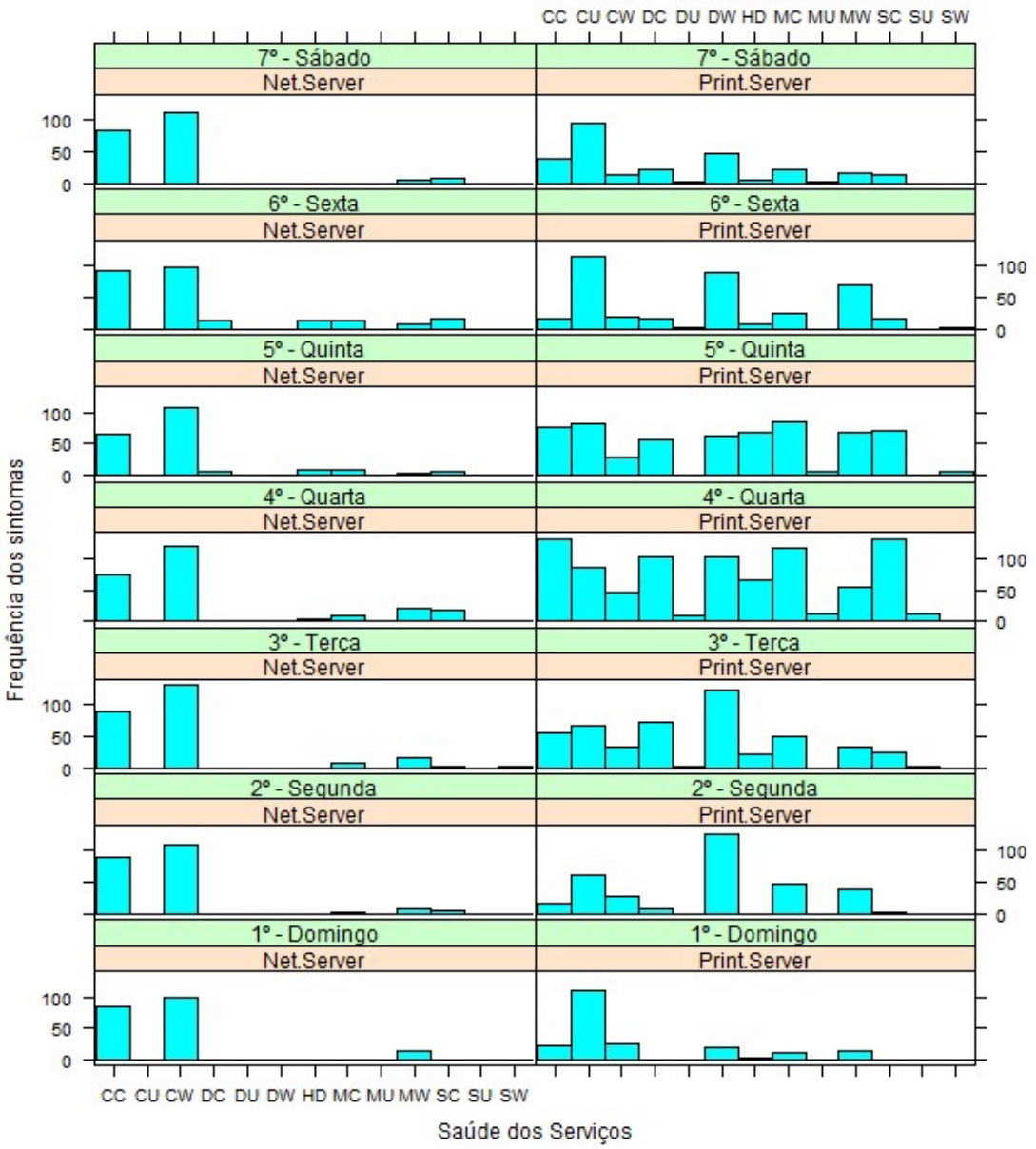


Figura 6.2: Saúde dos serviços Net Server e Print Server dado um dia da semana.

Os códigos de eventos de alertas do Nagios dispostos no eixo X do gráfico na Figura 6.2, estão descritos na Tabela 6.2.

Pode-se observar que há uma ocorrência maior de alertas para o servidor Print Server comparado ao servidor Net Server. Em ambos, os dias da semana com menor demanda são sábado e domingo, o que faz com que o número de alertas para falhas

Tabela 6.2: Eventos de alerta do Nagios e seus códigos.

Código	Nome	Código	Nome	Código	Nome
CC	CPU Critical	MC	Memory Critical	AC	Archive Critical
CU	CPU Unknown	MU	Memory Unknown	AU	Archive Unknown
CW	CPU Warning	MW	Memory Warning	AW	Archive Warning
DC	Disk Critical	SC	Swap Critical	DC	DW Process Critical
DU	Disk Unknown	SU	Swap Unknown	FC	File Server Critical
DW	Disk Warning	SW	Swap Warning	FU	File Server Unknown
HD	Host Down	OC	Oracle Critical	FW	File Server Warning
HU	Host Unknown	OU	Oracle Unknown	MC	Message Server Critical
TC	TNS Critical	GC	SAP Gateway Critical		

também diminua. Esse comportamento também acontece na comparação com outros conjuntos de servidores, na Figura 6.3.

Porém, nos demais dias da semana, a frequência de "sintomas" aumenta para uns e permanece baixo para outros. Por exemplo, o servidor Net Server da Figura 6.2, e o servidor SAPPRD GRC Nfe da Figura 6.3 tem baixa incidência de falhas, devido a dois fatores: os recursos são pouco demandados pelos clientes, ou os recursos para esses servidores foram corretamente dimensionados.

Analisando-se detalhadamente o servidor Net Server é possível verificar que o mesmo não possui alta variação nas demandas dos serviços e sistemas. Ele possui recursos computacionais de acordo com as especificações, embora no gráfico da Figura 6.2 o recurso "CPU" tenha mais de 100 eventos de alerta WARNING e mais de 50 eventos de alerta CRITICAL, indicando a necessidade de ajuste no poder de processamento do servidor. No caso do servidor SAPPRD GRC Nfe, da Figura 6.3, os recursos computacionais foram corretamente dimensionados, de acordo com as especificações, porém variações na demanda dos serviços causam pequena taxa de alertas a este servidor.

Por outro lado, para o servidor Print Server, a frequência de sintomas é elevada para recursos computacionais, tais como CPU, Disco, Memória e Swap. Isso indica que os recursos computacionais para este servidor não foram corretamente dimensionados e ainda sofrem alta demanda dos usuários. Embora este servidor tenha sido corretamente dimensionado, segundo especialista do sistema. Entretanto, com a instalação de novos sistemas e serviços, e com o aumento do número de impressoras na empresa, houve também aumento na demanda dos recursos computacionais. Desse modo, para reduzir o número de sintomas, faz-se necessário realizar um novo provisionamento de recursos computacionais para o servidor Print Server.

O SAPPRD HA CI é o servidor do sistema SAP ERP (Sistema integrado de gestão

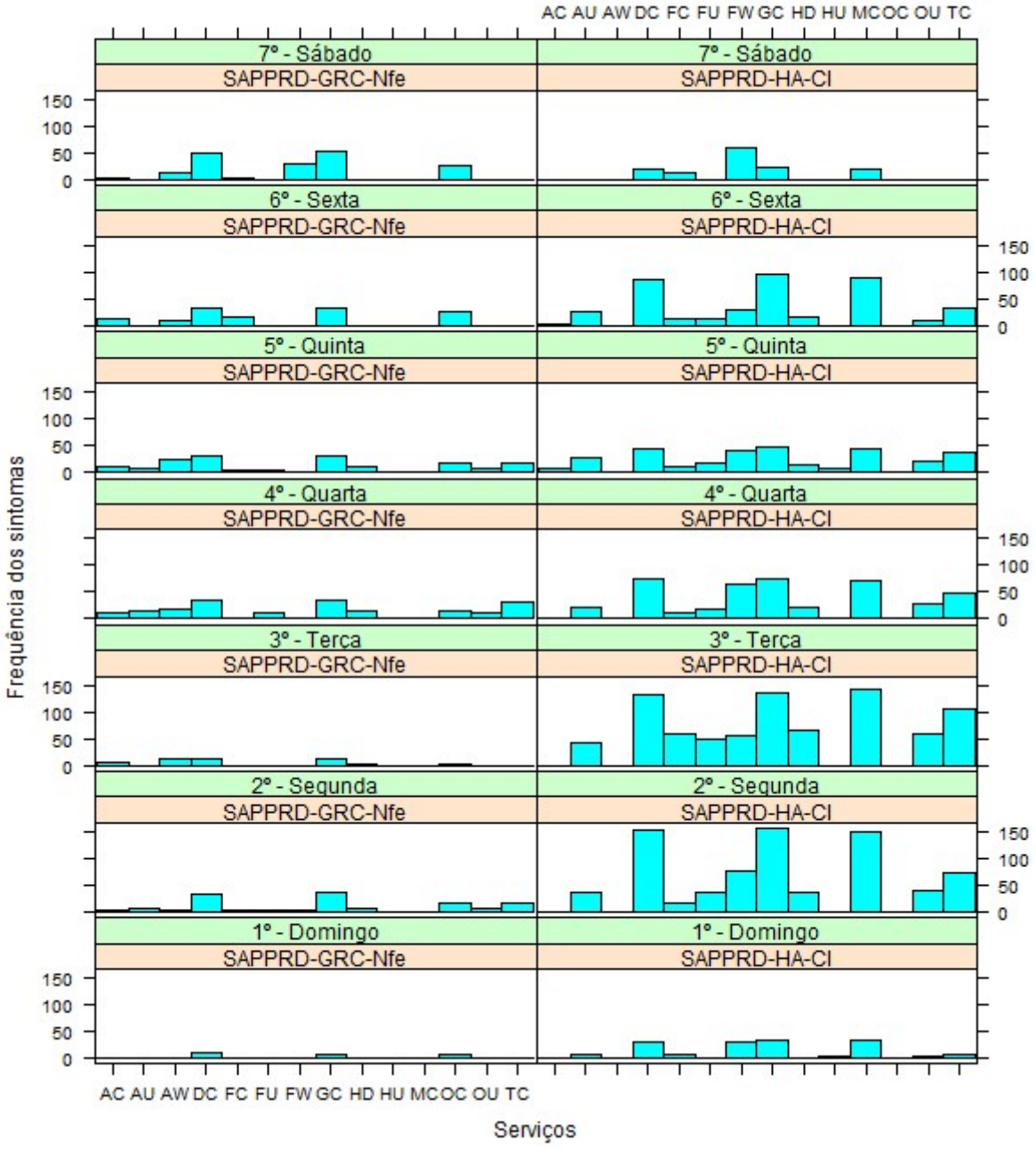


Figura 6.3: Saúde dos sistemas SAPPRD, GRC NFe e HA CI dado um dia da semana.

empresarial), e neste servidor temos a *central instance*, que é o conceito único da SAP. A *central instance* é a combinação de *hardware* e *software*, contendo o servidor físico (servidor de aplicação) e uma variedade de componentes de *software*, incluindo um *message server*, um *database gateway* (conexão preestabelecida entre o SAP ERP e o banco de dados Oracle), dentre outros componentes de *software*. Além do SAP ERP, este servidor conta com uma instância do banco de dados Oracle. Visto que a arquitetura do SAP ERP é complexa, associar as falhas ao sistema apenas a um mau dimensionamento de recursos computacionais seria negligenciar o grande número de variáveis e configurações que um ERP possui. Entretanto, se não houver espaço em disco para o sistema de arquivo, serviços como o do banco de dados Oracle poderá ser interrompido e, como consequência, o SAP ERP também. Portanto, a alta demanda pelos serviços do servidor SAPPRD HA CI é uma das causas que levam a uma frequência maior dos sintomas de falhas, comparado ao servidor SAPPRD GRC Nfe, como demonstrado na Figura 6.3.

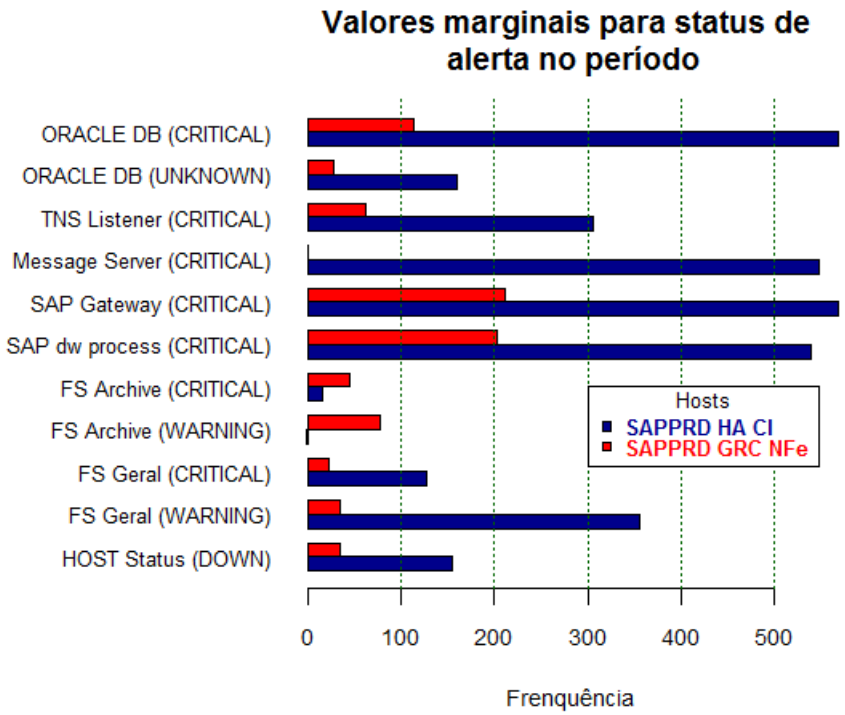


Figura 6.4: Alertas para o estado da saúde dos sistemas SAPPRD.

Pode-se observar na Figura 6.4 que os serviços monitorados pelo servidor SAPPRD HA CI, tais como *SAP dw process*, *SAP Gateway*, *Message Server* e *ORACLE DB*, possuem a maior frequência de alertas para falhas, indicando que a falha direta ou indireta de algum desses serviços, pode influenciar na falha dos demais. Os atributos *HOST Status (DOWN)*, *FS Geral WARNING* e *FS Archive (WARNING)*, desse gráfico (Figura 6.4,

influenciam diretamente na frequência de alerta dos demais atributos, indicando uma provável relação de causa-efeito, ou seja, quanto maior a frequência de falhas para estes atributos, também será maior a frequência de falhas nos demais atributos.

Para fazer o balanceamento de carga das demandas dos usuários para acesso ao sistema, o servidor SAPPRD HA CI possui no *cluster* mais dois servidores de aplicativos: SAPPRD AS S1 e SAPPRD AS S2. Um servidor de aplicativo é um conjunto de executáveis que interpreta coletivamente os programas do ERP e gerencia as entradas e saídas para eles. As solicitações dos servidores de aplicação para seleção de dados no banco de dados são dirigidas ao servidor SAPPRD HA CI, em que o banco de dados está instanciado.

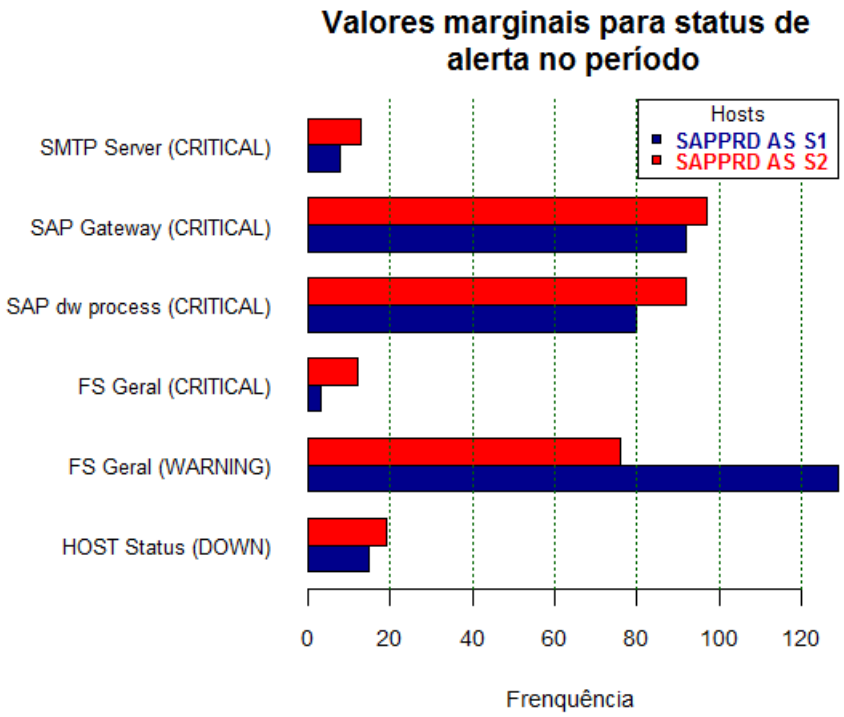


Figura 6.5: Alertas para o estado da saúde no cluster do servidor de aplicações SAPPRD.

Os servidores de aplicações da Figura 6.5 apresentam dimensionamento e configuração identicas, o que reflete um número muito próximo de ocorrência de alertas entre os servidores. Segundo o funcionamento do sistema SAP, os serviços destes servidores são mais sensíveis à indisponibilidade dos recursos físicos, mas outros fatores podem levar à indisponibilidade dos serviços, como manutenções preventivas e corretivas.

Baseado em entrevistas com especialistas da empresa, os *datasets* gerados são consistentes e apresentam o real estado das falhas ocorridas nos servidores monitorados ao longo de seis anos. Também é nítido que o mau dimensionamento dos recursos

computacionais, bem como o aumento da demanda dos serviços e sistemas podem causar aumento da frequência de sintomas prejudiciais à saúde dos sistemas e serviços na nuvem. Entretanto, apesar dos esforços de análise, é difícil aferir a relação da causa-efeito de falhas nos serviços e sistemas, dado o grande número de variáveis. Com o objetivo de identificar as relações entre as variáveis nos *datasets* usando probabilidades condicionais foi desenvolvido o módulo de inferência da arquitetura proposta. Esse módulo após o pré-processamento, obteve seis redes Bayesianas, cada uma para o respectivo *dataset* da Tabela 6.1. As redes Bayesianas geradas para cada *dataset* da Tabela 6.1 estão identificadas e podem ser vistas na Figura 6.6.

A partir de novas evidências dos estados de cada sistema e serviço aplicadas às respectivas rede Bayesianas, as probabilidades dos nós das redes são atualizadas, porém nem todos os nós são conclusivos na indicação de falhas dos sistemas e serviços. A fim de definir os nós de diagnósticos das redes, contou-se com a ajuda de um especialista que indicou para os sistemas SAP, os nós: SAP Gateway, Message Server, TNS Listener e JAVA WEB; para os serviços do servidor NET SERVER, os nós: DNS e DHCP; e para o servidor de impressão, o nó LPR Port.

Após a definição dos nós de diagnósticos das redes Bayesianas, aplicou-se o método de validação cruzada denominado *k-fold*, a fim de validar a precisão e a capacidade de generalização das redes Bayesianas para novas evidências de falhas nos sistemas e serviços. A Tabela 6.3 apresenta os resultados resumidos pela média aritmética dos resultados sintetizados dos quatro *folds* para todos os métodos de validação da tabela. Na Tabela 6.3, pode-se observar que a *precisão total* (TP) dos nós observados para redes Bayesianas apresentou excelentes resultados para predição e diagnóstico de falhas; para o coeficiente de *Pearson* (R), apenas o nó observado DHCP da rede Bayesiana Print Server apresentou a correção moderada de 0.5493 entre as variáveis da rede Bayesiana. Todos os outros nós observados das redes apresentaram correlação muito forte entre as variáveis, como coeficiente de *Pearson* acima de 0.90. Para a medida de ajustamento dos dados às redes Bayesianas, o coeficiente de determinação, conhecido como quadrado de *Pearson* (R2), acompanha as classificações para as demais avaliações, em que apenas o nó observado DHCP da rede Bayesiana Print Server apresenta uma piora na avaliação. A média relativa do índice de *Gini* (RGIM) expressa a dispersão das variáveis dos subconjuntos de dados das amostras de testes, e observando na Tabela 6.3, o nó observado que apresentou menor dispersão foi o DHCP, o que pode indicar uma ocorrência menor dos estados de falhas para o nó observado.

Tabela 6.3: Resultados para o método de validação cruzada *k-fold*.

Rede Bayesiana Print Server						
Nó observado	TP$^{(1)}$	R$^{(2)}$	R2$^{(3)}$	RGIM$^{(4)}$	RLIM$^{(5)}$	RIM$^{(6)}$
LPR Port	96,02%	0,9462	0,8952	85,90%	93,41%	92,95%
Rede Bayesiana Net Server						
Nó observado	TP$^{(1)}$	R$^{(2)}$	R2$^{(3)}$	RGIM$^{(4)}$	RLIM$^{(5)}$	RIM$^{(6)}$
DNS	99,76%	0,9017	0,8131	99,90%	99,05%	99,96%
DHCP	88,75%	0,5493	0,3017	35,63%	71,19%	67,82%
Rede Bayesiana SAP GRC NFe						
Nó observado	TP$^{(1)}$	R$^{(2)}$	R2$^{(3)}$	RGIM$^{(4)}$	RLIM$^{(5)}$	RIM$^{(6)}$
SAP Gateway	96,01%	0,9781	0,9567	93,07%	97,13%	96,54%
TNS Listener	98,86%	0,9637	0,9288	96,03%	98,40%	98,03%
JAVA WEB	95,04%	0,9881	0,9763	84,02%	95,07%	92,01%
Rede Bayesiana SAPPRD HA CI						
Nó observado	TP$^{(1)}$	R$^{(2)}$	R2$^{(3)}$	RGIM$^{(4)}$	RLIM$^{(5)}$	RIM$^{(6)}$
Message Server	95,23%	0,9772	0,9549	96,88%	98,96%	98,44%
SAP Gateway	95,02%	0,9844	0,9691	92,99%	97,15%	95,50%
TNS Listener	94%	0,9519	0,9061	94,86%	97,82%	97,43%
Rede Bayesiana SAPPRD AS S1						
Nó observado	TP$^{(1)}$	R$^{(2)}$	R2$^{(3)}$	RGIM$^{(4)}$	RLIM$^{(5)}$	RIM$^{(6)}$
SAP Gateway AS S1	96,54%	0,9953	0,9907	79,06%	92,43%	89,54%
Rede Bayesiana SAPPRD AS S2						
Nó observado	TP$^{(1)}$	R$^{(2)}$	R2$^{(3)}$	RGIM$^{(4)}$	RLIM$^{(5)}$	RIM$^{(6)}$
SAP Gateway AS S2	96,50%	0,9946	0,9892	82,12%	93,90%	91,07%

(1) Total de Precisão. (2) Coeficiente de *Pearson*. (3) Quadrado de Pearson. (4) Média relativa do índice de *Gini*. (5) Média relativa do índice *lift*. (6) Média do índice ROC.

Outra avaliação realizada reflete a confiança nos resultados de classificação, e a média relativa do índice *lift* (RLIM) refletiu nos resultados de classificação uma boa confiança para as redes Bayesianas avaliadas. Os resultados para a média do índice ROC (RIM) apresentaram uma *excelente* precisão no teste de diagnóstico, considerando a aproximação de 100% para todos os nós, menos para o nó DHCP, que apresenta precisão *pobre*, de acordo com a escala para a média do índice ROC. Esses resultados obtidos e demonstrados na Tabela 6.3 reapresentam para as redes Bayesianas um desempenho excelente para o diagnóstico de falhas na saúde dos sistemas e serviços na nuvem. Uma forma de melhorar o desempenho para alguns nós, como é o caso do nó DHCP da rede Bayesiana, é considerar uma *dataset* mais representativo para a variável.

Depois de validados os resultados com a ajuda de um especialista da área e pelo método de validação cruzada *k-fold*, os nós foram definidos como conclusivos para o

diagnóstico de falhas nos sistemas. Entretanto, embora eles sejam conclusivos para diagnóstico de suas respectivas redes Bayesianas, porém para alguns serviços existe também uma dependência entre serviços e sistemas que são diferentes ou apenas distribuídos em servidores distintos.

A relação de interdependência pode ser observada nos serviços de emissão de NF-e (Nota Fiscal Eletrônica), representada pela Figura 6.6. Neste cenário, os clientes necessitam de serviços de rede (NET SERVER) para se conectarem à infraestrutura dos sistemas e serviços, a fim de realizar a emissão de notas fiscais. Para emissão de notas fiscais, os clientes precisam dos dados fiscais e contábeis da empresa que estão no ERP (SAPPRD HA CI, SAPPRD AS S1 e SAPPRD AS S2), bem como ter um sistema (SAP GRC NFe) que realize a mensageira desses dados junto à Receita Federal para a validação e a emissão da nota fiscal. Alguns tipos de notas fiscais precisam ser impressos (PRINT SERVER), como no caso de emissão de nota fiscal de simples remessa, ou seja, notas fiscais de mercadorias que são frutos de venda ou de transporte.

Com o objetivo de avaliar a saúde de todos os sistemas e serviços envolvido no processo de emissão de NF-e, utilizou-se de nós determinísticos, a fim de criar as relações de dependência para os servidores dos sistemas, representados na Figura 6.6 pelas redes Bayesianas. Os nós determinísticos são representados na Figura 6.6 pelas tabelas com barras horizontais. Nós determinísticos correspondem a uma função booleana e representam valores que são algebricamente determinados a partir dos estados de seus pais. Não há nenhuma incerteza associada aos resultados de um nó determinístico, uma vez que todos os seus pais são conhecidos. A única diferença é que suas TCPs contêm apenas zeros e uns.

Para o nó determinístico SAPPRD CLUSTER AS da Figura 6.6, tem-se a TPC da Tabela 6.4, em que o número de possibilidade de configurações para esta tabela é $2^2 * 3 = 12$ possíveis configurações para os estados de alerta do nó determinístico, em que a combinação dos estados dos nós pais, podem ser 1 para verdadeiro e 0 para falso, elevado ao número de nós pais, vezes o número de estados da TCP do nó determinístico.

Tabela 6.4: TPC do nó determinístico SAPPRD CLUSTER AS.

SAP Gateway AS S1	CRITICAL		OK	
SAP Gateway AS S2	CRITICAL	OK	CRITICAL	OK
Estado Crítico	1	0	0	0
Estado de Alerta	0	1	1	0
Estado Normal	0	0	0	1

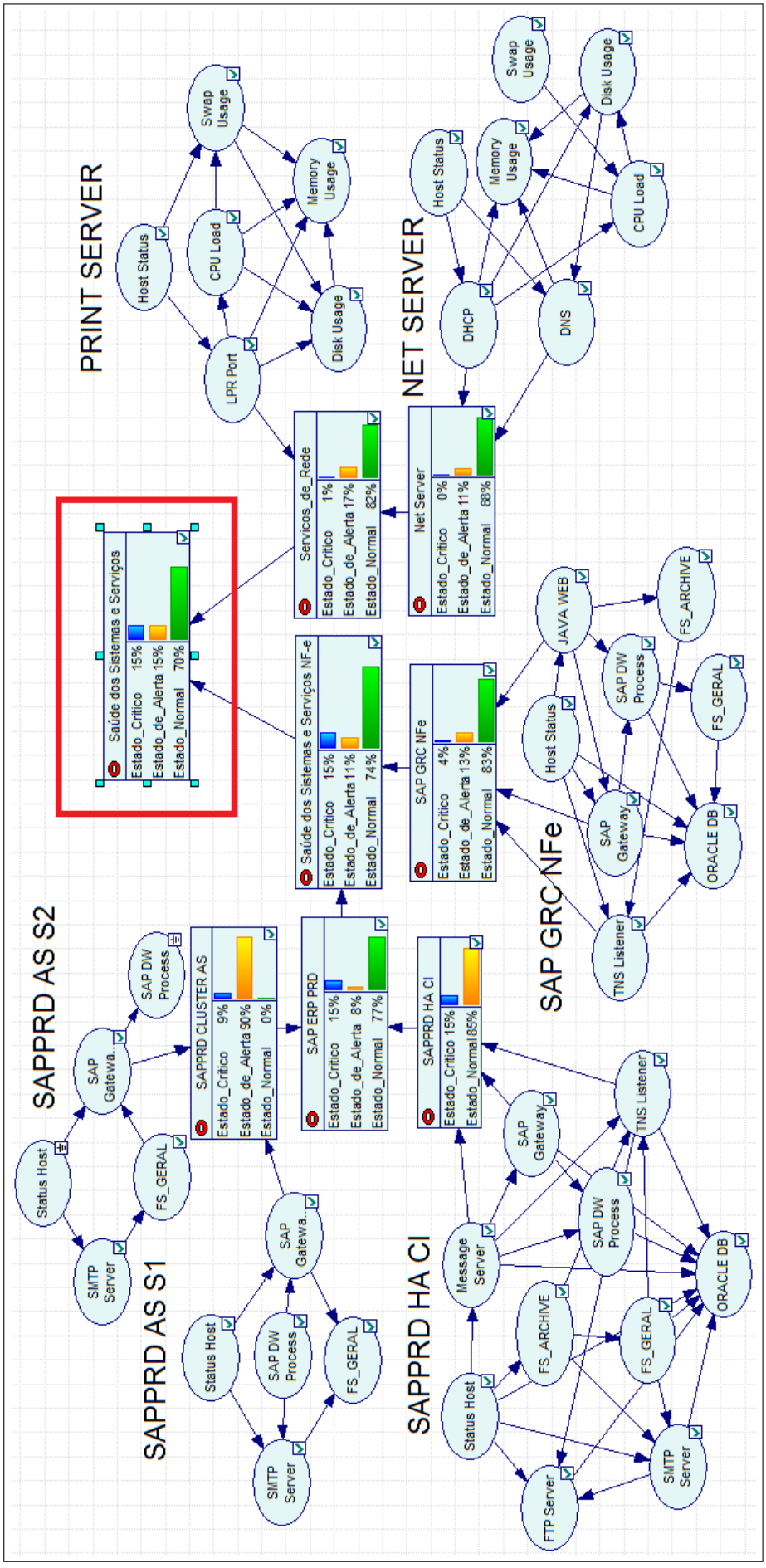


Figura 6.6: Redes Bayesianas empregadas no gerenciamento da saúde dos sistemas na nuvem.

Especificadas as relações dos nós das redes Bayesianas, para cada nó determinístico. As probabilidades do estado da saúde dos sistemas e serviços na nuvem são indicadas pelo nó determinístico "Saúde dos Sistemas e Serviços".

As configuração das TPCs para os nós determinísticos são feitas com base no conhecimento especializado das dependencias do sistema como um todo. Portanto, com ajuda de um especialista, definimos que, se os nós de diagnóstico SAP Gateway S1 da rede Bayesiana SAPPRD AS S2 e SAP Gateway S1 da rede Bayesiana SAPPRD AS S2 estivem em estado crítico, o nó determinístico da Tabela 6.4 também deve apresentar seu estado como crítico, mas se apenas um dos nós de observação falhar, a configuração da TCP deve indicar estado de alerta, como pode ser visto na Tabela 6.4. No caso em que os dois nós de observação estão no estado OK, o especialista indica que o estado do cluster deve ser normal.

Os valores representados pelas barras horizontais nos estados dos nós determinísticos da Figura 6.6 expressam as probabilidades do nós determinístico obtido algebricamente pelos cálculos das probabilidades marginais de seus respectivos pais, considerando uma rede Bayesiana sem evidências. A fim de demonstrar os cálculos, considere o nó determinístico SAPPRD CLUSTER AS (Estado de Alerta = Verdadeiro), em que a probabilidade marginal dos nós pais para a configuração da TCP na Tabela 6.4 é: P(SAP Gateway AS S1 = CRITICAL) = 0,0925, P(SAP Gateway AS S2 = OK) = 0,9, P(SAP Gateway AS S1 = CRITICAL) = 0,9074 e P(SAP Gateway AS S2 = OK) = 0,0996. A distribuição de probabilidade conjunta para as variáveis independentes na configuração apresentada na TCP da Tabela 6.4 é: Ps1s2 = P(SAP Gateway AS S1 = CRITICAL e SAP Gateway AS S2 = OK) = 0,0925 x 0,9 = 0,08325 e Ps2s1 = P(SAP Gateway AS S2 = OK e SAP Gateway AS S1 = CRITICAL) = 0,0996 x 0,9074 = 0,0903. Com isso, a probabilidade do nó determinístico SAPPRD CLUSTER AS para o estado de alerta é P(Estado de Alerta = Verdadeiro) = Ps1s2 + Ps2s1 = 0,17355 $\approx$ 17%. Dessa forma, são feitos os cálculos para as probabilidades dos nós determinísticos.

Os nós determinísticos foram usados de forma a parametrizar os impactos e os riscos à saúde dos sistemas na nuvem, sendo o nó determinístico *Saúde dos Sistemas e Serviços* da Figura 6.6, o nó que representa o diagnóstico de falhas e saúde dos serviços e sistemas usados para emissão de NF-e.

Capítulo 7

CONCLUSÃO

Este trabalho teve como foco propor uma arquitetura de sistema com suporte à decisão para o gerenciamento da saúde dos sistemas e serviços na computação em nuvem. A arquitetura proposta integra a ferramenta de monitoramento Nagios com o objetivo de obter informações sobre a saúde dos sistema e serviços dispostos no ambiente computacional. Com uma abordagem modular a complexidade da arquitetura foi reduzida, permitindo realizar a pesquisa, a análise e a configuração, bem como o desenvolvimento de novas soluções.

Dessa forma, o Nagios foi integrado ao módulo KKD, tendo seus registros de monitoramento processados, onde os arquivos de logs são minerados em busca de padrões que representem a saúde dos sistemas e serviços na nuvem. Para isso, este módulo dispõe de dois algoritmos: MAPLOG, responsável por realizar o pré-processamento dos dados, e o TOTRANS, responsável por realizar a transformação dos dados pré--processados. O módulo KDD processou 12.412.259 linhas de eventos de monitoramento, gerando seis conjuntos de dados contendo atributos que determinam a saúde dos sistemas que tem seus serviços relacionados à emissão de notas fiscais eletrônicas (NF-e). Após análise dos dados nos *datasets*, o especialista [1] validou o processamento realizado pelo módulo KDD. De acordo com o especialista, os gráficos gerados, tendo como base os *datasets*, representam fielmente o comportamento de falhas ocorridas nos sistemas. Portanto, demonstrando que, quando os recursos computacionais estão corretamente dimensionados, os serviços são menos sensíveis a falhas quando há aumento na demanda dos serviços e sistemas.

Os algoritmos foram propostos com o objetivo de gerar dados estruturados para o módulo desenvolvido de inferência, que é responsável pela construção das redes

[1] Especialista de infra-estrutura da empresa

Bayesianas apropriadas modelando o processo de decisão envolvendo incertezas e observações parciais, características presentes na computação em nuvem. Tais redes Bayesianas foram construídas com o uso do algoritmo PC (usando métodos baseado em restrições, e testes de independencia entre variáveis) para aprendizagem estrutural. Em seguida, com essa rede inferida, foi feito um trabalho junto a um especialista em computação em nuvem, para validar e auxiliar, em caso de ajustes. Finalmente, os parâmetros da rede bayesiana foram estimados com o uso do algoritmo de aprendizagem paramétrica EM (expectation–maximization). Os algoritmos de aprendizagem supervisionada foram orientados pelos *datasets* gerados pelo módulo KDD. Dessa forma, foram geradas seis redes que formam a rede bayesiana completa, uma para cada *dataset*.

As redes Bayesianas geradas foram avaliadas através do conhecimento de um especialista e também por medidas de desempenho, utilizando a validação cruzada *k-fold*. O especialista validou e apoio a construção estrutural das redes Bayesianas, por ter domínio na relação causa e efeito das falhas nos sistemas. Para a validação paramétrica, foram utilizadas medidas de desempenho. Como resultado, as redes Bayesianas, de acordo com os resultados inferidos, apresentaram um excelente desempenho e precisão para o diagnóstico de falhas na saúde dos sistemas e serviços na nuvem.

Na fase de validação, para cada rede Bayesiana, alguns nós de cada serviço ou sistema foram considerados como conclusivos para o diagnóstico de falhas. Como o objetivo da arquitetura é avaliar a saúde de todos os sistemas e serviços na nuvem, levando em consideração o sistema de emissão de nota fiscal eletrônica (NF-e), utilizou-se de nós determinísticos, a fim de reproduzir a correta relação de dependência entre os nós observados das redes Bayesianas, validado pelo apoio de um especialista nos sistemas, e portanto, obtendo ao final do processo um único nó que representa a saúde global do sistema. Cada nó determinístico dá suporte ao provisionamento de recursos em casos de falhas ou alta demanda nos sistemas e serviços. Dessa forma, é possível oferecer certa confiabilidade e elasticidade para os recursos computacionais dos sistemas.

É possível concluir para este trabalho que, para a modelagem das redes Bayesianas do sistema de suporte à decisão, é imprescindível a participação de um especialista da área. Com o apoio do especialista, a arquitetura proposta apresentou resultados promissores, tendo resultado e interpretação coerente com os resultados e medidas de validação das redes Bayesianas.

Para a realização de trabalhos futuros, propõe-se o desenvolvimento do software

para a arquitetura proposta, sendo disponibilizado, em tempo real, diretamente na nuvem com uma interface gráfica que possibilite uma utilização mais intuitiva e com melhor apresentação visual das redes Bayesianas. Além disso, uma outra possibilidade é que o sistema tome medidas pró-ativas, baseado nas inferencias bayesianas, para melhoria da saúde dos sistemas de nuvem.

REFERÊNCIAS

BARTH, W. *Nagios: System and Network Monitoring*. Open Source Press, 2006. (No Starch Press Series). ISBN 9781593270704. Disponível em: <https://books.google.com.br/books?id=F3ealpHDklUC>.

BAYES Nets. Agost 2014. Disponível em: <http://www.bayesnets.com/>.

BERMUDEZ, I. et al. Exploring the cloud from passive measurements: The amazon aws case. In: *INFOCOM, 2013 Proceedings IEEE*. [S.l.: s.n.], 2013. p. 230–234. ISSN 0743-166X.

BERMUDEZ, I. N. et al. Dns to the rescue: Discerning content and services in a tangled web. In: *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*. New York, NY, USA: ACM, 2012. (IMC '12), p. 413–426. ISBN 978-1-4503-1705-4. Disponível em: <http://doi.acm.org/10.1145/2398776.2398819>.

BOUCKAERT, R. R. *Probabilistic Network Construction Using the Minimum Description Length Principle*. 1994.

BUNTINE, W. Theory refinement on bayesian networks. In: . [S.l.]: Morgan Kaufmann, 1991. p. 52–60.

BUYYA, R.; VECCHIOLA, C.; SELVI, S. T. *Mastering cloud computing: foundations and applications programming*. [S.l.]: Newnes, 2013.

CABENA, P. et al. *Discovering Data Mining: From Concept to Implementation*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1998. ISBN 0-13-743980-6.

CAMPOS, L. M. de; HUETE, J. F. A new approach for learning belief networks using independence criteria. *International Journal of Approximate Reasoning*, v. 24, n. 1, p. 11 – 37, 2000. ISSN 0888-613X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0888613X99000420>.

CHANG, W. *R graphics cookbook*. [S.l.]: "O'Reilly Media, Inc.", 2012.

CHENG, J. et al. Learning bayesian networks from data: An information-theory based approach. *Artif. Intell.*, Elsevier Science Publishers Ltd., Essex, UK, v. 137, n. 1-2, p. 43–90, maio 2002. ISSN 0004-3702. Disponível em: <http://dx.doi.org/10.1016/S0004-3702(02)00191-1>.

CHOW, C.; LIU, C. Approximating discrete probability distributions with dependence trees. *IEEE Trans. Inf. Theor.*, IEEE Press, Piscataway, NJ, USA, v. 14, n. 3, p. 462–467, set. 2006. ISSN 0018-9448. Disponível em: <http://dx.doi.org/10.1109/TIT.1968.1054142>.

CIOS, K. J. et al. A knowledge discovery approach to diagnosing myocardial perfusion. *IEEE Engineering in Medicine and Biology Magazine*, v. 19, n. 4, p. 17–25, July 2000. ISSN 0739-5175.

DODGSON, J. et al. *Multi-criteria analysis: a manual*. [S.l.]: Department for Communities and Local Government: London, 2009.

DUMOULIN, B.; GUIMARÃES, D.; NEVES, G. *O método AHP como ferramenta de focalização do processo de gerenciamento de projetos-Caso: APEX-Brasil*. [S.l.]: Macroplan. Publicação Eletrônica, 2006.

FAYYAD, U. *Advances in Knowledge Discovery and Data Mining*. AAAI Press, 1996. (AAAI Press Series). ISBN 9780262560979. Disponível em: <https://books.google.com.br/books?id=XqVQAAAAMAAJ>.

FERNÁNDEZ, V. et al. Arduino and nagios integration for monitoring. In: IOP PUBLISHING. *Journal of Physics: Conference Series*. [S.l.], 2014. v. 513, n. 6, p. 062015.

FERREIRA, M. M.; MARTINS, J. M.; SALGADO, E. G. Aplicação do método analytic hierarchy process para a adoção de computação em nuvem em empresas juniores. *Revista Eletrônica de Sistemas de Informação e de Gestão Tecnológica*, v. 5, n. 1, 2015.

FOX, A. et al. Above the clouds: A berkeley view of cloud computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, v. 28, n. 13, p. 2009, 2009.

GALSTAD, E. Nagios core version 3. x documentation. *Nagios Community*, 2009.

HARRIS, M.; GENG, H. Data center infrastructure management. *Data Center Handbook*, Wiley Online Library, p. 601–618, 2015.

HECKERMAN, D. Learning in graphical models. In: JORDAN, M. I. (Ed.). Cambridge, MA, USA: MIT Press, 1999. cap. A Tutorial on Learning with Bayesian Networks, p. 301–354. ISBN 0-262-60032-3. Disponível em: <http://dl.acm.org/citation.cfm?id=308574.308676>.

HOFFMANN, J. 16. probabilistic reasoning, part ii: Bayesian networks putting the machinery to practical use. Citeseer, 2014.

IaaS - Provedores de Serviços. 2016. Acessado em 3 de abril, de 2016. Disponível em: <https://aws.amazon.com/; https://www.rackspace.com/; https://www.cloudsigma.com/>.

JÚNIOR, B. Conceito nacional de empresa júnior. *Acessado em*, v. 12, n. 10, 2012.

JÚNIOR, E. R. H. *Imputação bayesiana no contexto da mineração de dados*. Tese (Doutorado) — PhD thesis, Universidade Federal do Rio de Janeiro, 2003.

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

KORB, K.; NICHOLSON, A. *Bayesian Artificial Intelligence, Second Edition*. CRC Press, 2010. (Chapman & Hall/CRC Computer Science & Data Analysis). ISBN 9781439815922. Disponível em: <https://books.google.com.br/books?id=LxXOBQAAQBAJ>.

KUNDU, D.; LAVLU, S. *Cacti 0.8 Network Monitoring*. Packt Publishing, 2009. (From Technologies to Solutions). ISBN 9781847195975. Disponível em: <https://books.google.com.br/books?id=BtyfzrPeAX0C>.

KURGAN, L. A.; MUSILEK, P. A survey of knowledge discovery and data mining process models. *Knowl. Eng. Rev.*, Cambridge University Press, New York, NY, USA, v. 21, n. 1, p. 1–24, mar. 2006. ISSN 0269-8889. Disponível em: <http://dx.doi.org/10.1017/S0269888906000737>.

LEMES, J. É. A. Monitoramento em redes ipv6 com zabbix e raspberry pi. Curitiba, 2014.

LI, Y.; ZHAO, C.; YIN, Y. A method of building the fault propogation model of distributed application systems based on bayesian network. In: *Computer Science and Engineering, 2009. WCSE '09. Second International Workshop on*. [S.l.: s.n.], 2009. v. 1, p. 20–24.

MEISTER, D.; SULLIVAN, D. *Evaluation of User Reactions to a Prototype On-line Information Retrieval System*. Clearinghouse for Federal Scientific and Technical Information, 1967. (Evaluation of User Reactions to a Prototype On-line Information Retrieval System, v. 918). Disponível em: <https://books.google.com.br/books?id=O5UaAAAAMAAJ>.

MELL, P.; GRANCE, T. *The NIST Definition of Cloud Computing*. Gaithersburg, MD, September 2011. Disponível em: <http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf>.

MI, H. et al. Performance problems online detection in cloud computing systems via analyzing request execution paths. In: *Dependable Systems and Networks Workshops (DSN-W), 2011 IEEE/IFIP 41st International Conference on*. [S.l.: s.n.], 2011. p. 135–139.

NATARAJAN, R. *Votação para sistemas de monitoramento favoritos*. O artigo foi endereçado ao blog Linux and Open Source Technologies: [s.n.], 2009. Disponível em: <http://www.thegeekstuff.com/2009/09/top-5-best-network-monitoring-tools/>.

PAAS - Provedores de Serviços. 2016. Acessado em 3 de abril, de 2016. Disponível em: <https://azure.microsoft.com/; https://cloud.google.com/appengine/; https://www.salesforce.com/>.

PENA, S. D. Thomas bayes: o 'cara'! In: . [S.l.]: Revista Ciência Hoje, 2006. v. 38, p. 22–29.

REFAEILZADEH, P.; TANG, L.; LIU, H. Cross-validation. In: *Encyclopedia of database systems*. [S.l.]: Springer, 2009. p. 532–538.

RUIZ, E. E. S. Mineração de texto em saúde. *Journal of Health Informatics*, v. 8, n. 1, 2016.

RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2010. (Prentice Hall series in artificial intelligence). ISBN 9780136042594. Disponível em: <http://books.google.com.br/books?id=8jZBksh-bUMC>.

SAATY, T. *Mathematical Principles of Decision Making (Principia Mathematica Decernendi)*. RWS Publications, 2010. ISBN 9781888603149. Disponível em: <https://books.google.com.br/books?id=EURSngEACAAJ>.

SAHASRABUDHE, S. S.; SONAWANI, S. S. Comparing openstack and vmware. In: *Advances in Electronics, Computers and Communications (ICAECC), 2014 International Conference on*. [S.l.: s.n.], 2014. p. 1–4.

SCHUBERT, M. et al. *Nagios 3 Enterprise Network Monitoring: Including Plug-Ins and Hardware Devices*. Elsevier Science, 2008. ISBN 9780080560182. Disponível em: <https://books.google.com.br/books?id=d4U7fLQuRaQC>.

SEMPOLINSKI, P.; THAIN, D. A comparison and critique of eucalyptus, opennebula and nimbus. In: IEEE. *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*. [S.l.], 2010. p. 417–426.

SHEARER, C. The crisp-dm model: The new blueprint for data mining. *Journal of Data Warehousing*, v. 5, n. 4, 2000.

SMITH, J.; NAIR, R. *Virtual machines: versatile platforms for systems and processes*. [S.l.]: Elsevier, 2005.

SMITH, J. E.; NAIR, R. The architecture of virtual machines. *Computer*, IEEE, v. 38, n. 5, p. 32–38, 2005.

SOTOMAYOR, B. et al. Virtual infrastructure management in private and hybrid clouds. *Internet computing, IEEE*, IEEE, v. 13, n. 5, p. 14–22, 2009.

TSURUOKA, Y. Cloud computing-current status and future directions. *Journal of Information Processing*, v. 24, n. 2, p. 183–194, 2016.

VOGEL, A. J.; GRIEBLER, D.; ROVEDA, D. Understanding, discussing and analysing the opennebula's and openstack's iaas management layers. *Revista Eletrônica Argentina-Brasil de Tecnologias da Informação e da Comunicação*, v. 1, n. 3, 2015.

YISHAN, G.; CHEN, D.; JIA, H. The bayesian network about computer network failure diagnosis under osi. In: *Intelligent Human-Machine Systems and Cybernetics, 2009. IHMSC '09. International Conference on*. [S.l.: s.n.], 2009. v. 1, p. 44–46.

GLOSSÁRIO

ABI – *Application Binary Interface*

AHP – *Analytic Hierarchy Process*

API – *Application Programming Interface*

AWS – *Amazon Web Services*

BIC – *Bayesian Information Criterion*

CAPEX – *Capital Expenditure*

CDN – *Content Delivery Network*

CPT – *Condicional Probability Table*

CPU – *Central Processing Unit*

DAG – *Directed Acyclic Graphs*

DCIM – *Data Center Infrastructure Management*

DHCP – *Dynamic Host Configuration Protocol*

DNS – *Domain Naming Services*

EC2 – *Elastic Compute Cloud*

EM – *Expectation Maximization*

ERP – *Enterprise Resource Planning*

FS – *File System*

FTP – *File Transfer Protocol*

GB – *Gigabyte*

GS – *Greedly Search*

GUI – *Graphical User Interface*

HTTP – *Hypertext Transfer Protocol*

ICMP – *Internet Control Message Protocol*

IP – *Internet Protocol*

IaaS – *Infrastructure as a Service*

KDD – *Knowledge Discovery Databases*

MAPLOG – *Matching and Pre-processing Logs*

MB – *Megabyte*

ML – *Maximum Likelihood*

MWST – *Maximum Weight Spanning Tree Algorithm*

NRPE – *Nagios Remote Plugin Executor*

OPEX – *Operational Expenditure*

OSI – *Open Systems Interconnection*

PASI – *Performance Agent and Server Interface*

PCA – *Principal Component Analysis*

POP3 – *Post Office Protocol*

PaaS – *Platform as a Service*

RB – *Rede Bayesiana*

RTT – *Round Trip Time*

S3 – *Simple Storage Service*

SLA – *Service Level Agreement*

SMTP – *Simple Mail Transfer Protocol*

SNMP – *Simple Network Management Protocol*

SO – *Sistema Operacional*

SSD – *Sistema de Suporte a Decisão*

SSH – *Secure Shell*

SSL – *Secure Socket Layer*

SaaS – *Software as a Service*

TB – *Terabyte*

TCP – *Transmission Control Protocol*

TOTRANS – *Transpose To Transform*

TPC – *Tabela de Probabilidade Condicional*

UDP – *User Datagram Protocol*

Apendice A

INSTALAÇÃO DO NAGIOS

O processo de compilação e instalação de programas pelo código fonte é mais trabalhoso, para conduzir a instalação de uma forma mais prática é importante que sejam instaladas todas a dependências de pacotes necessários, para correta compilação e instalação do programa. A instalação dos pacotes necessários na compilação do Nagios pode ser realizada utilizando o gerenciador de pacotes *yum*, como é apresentado no Comando A.1.

Comando A.1: Instalação das dependências com "yum"

```
1 # yum install gd gd-devel httpd php gcc glibc glibc-common -y
```

O Nagios permite controle de acesso ao ambiente e para uma compilação básica é necessário a criação de usuário e grupos com acesso total a ferramenta. Neste caso para uma correta compilação do Nagios no Comando A.2 foram criados os grupos *"nagios"* e *"nagcmd"*, e o usuário *"nagios"*. É importante também a inclusão do usuário *"apache"* no grupo *"nagcmd"*, como apresentado na 5ª linha do Comando A.2.

Comando A.2: Criação e configuração de usuários e grupos para o Nagios

```
1 # groupadd nagios
2 # useradd -g nagios -s /bin/false -mk /home/nagios nagios
3 # passwd nagios
4 # groupadd nagcmd
5 # usermod -G nagcmd apache
6 # usermod -G nagcmd nagios
```

Depois de realizar a instalação das dependência e criar o usuário e grupos ne-

cessário para instalação, os próximos passos são: configurar, compilar e instalar o Nagios. Nesse processo a primeira atividade é obter o código fonte, acessando o seguinte site *http://www.nagios.org/donwload/* e realizando o download da ultima versão estável. Para esse trabalho foi realizada a instalação da versão 3.3.1 do Nagios. Realizado o download e a descompactação dos arquivos fontes do Nagios e plugins como apresentado no Comando A.3, o código fonte do Nagios é submetido ao processo de instalação com o uso do Comando A.4, em que na 1ª linha é criado o diretório de instalação, da 2ª à 7ª é definido os parâmetros de instalação, na 8ª o comando criar os binários, os demais comandos *"make install"* na ordem em que seguem, realizam a instalação dos binários, a instalação de script para inicialização e por fim a instalação dos exemplos de configuração no diretório *"/usr/local/nagios/etc/"*.

Comando A.3: Descompactação dos arquivos fontes

```
1 # tar xzvpf nagios-3.3.1.tar.gz
2 # tar xzvpf nagios-plugins-1.4.15.tar.gz
```

Depois de descompactar os arquivos, para efetuar o procedimento de instalação é necessário que seja realizado o acesso aos diretórios Nagios e Plugins, para que a respectiva instalação seja realizada.

Comando A.4: Configuração e instalação do Nagios

```
1  # mkdir -p /usr/local/nagios
2  # ./configure -prefix=/usr/local/nagios \
3  --with-cgiurl=/nagios/cgi-bin \
4  --with-htmlurl=/nagios \
5  --with-nagios-user=nagios \
6  --with-nagios-group=nagios \
7  --with-command-group=nagcmd
8  # make all -s
9  # make install -s
10 # make install-init -s
11 # make install-commandmode -s
12 # make install-config -s
```

Após a realização da instalação do Nagios foi realizada a instalação dos plugins. Para realizar o procedimento de instalação dos *plugins* foi necessário acessar a pasta dos arquivos que foram descompactados e depois executar o Comando A.5.

Comando A.5: Instalação dos plugins

```
# ./configure --prefix=/usr/local/nagios \
--with-nagios-user=nagios \
--with-nagios-grp=nagios
# make all -s
# make install -s
```

Após a compilação e instalação, o Nagios e seus plugins ficaram instalados no diretório */usr/local/nagios/* com a seguinte estrutura:

- ../bin (Arquivos binários do Nagios);
- ../etc (Arquivos de configuração);
- ../include (Não possui arquivos);
- ../libexec (Arquivos de Plugins do Nagios);
- ../sbin (Cgi's);
- ../share (Interface Web do nagios);
- ../var (Diretório de Logs).

Para acessar o Nagios pela interface Web e ainda adicionar controle de autenticação, é necessário realizar uma configuração no servidor Web. No Comando A.6 o arquivo de configuração é editado para que as linhas do Comando A.7 sejam adicionadas. Após a configuração o serviço *"http"* dever ser reinicializado.

Comando A.6: Configurar autenticação para o Nagios

```
# vi /etc/httpd/conf/httpd.conf
```

Comando A.7: Configurar autenticação para o Nagios

```
ScriptAlias /nagios/cgi-bin "/usr/local/nagios/sbin"

<Directory "/usr/local/nagios/sbin">
        AllowOverride None
        Options ExecCGI
        Order allow,deny
        Allow from all
        AuthName "Nagios Access"
        AuthType Basic
        AuthUserFile /usr/local/nagios/etc/htpasswd.users
        Require valid-user
</Directory>
```

```
13
14 Alias /nagios "/usr/local/nagios/share"
15 <Directory "/usr/local/nagios/share">
16         AllowOverride None
17         Options None
18         Order allow,deny
19         Allow from all
20         AuthName "Nagios Access"
21         AuthType Basic
22         AuthUserFile /usr/local/nagios/etc/htpasswd.users
23         Require valid-user
24 </Directory>
```

Depois de realizar a configuração acima utilize o Comando A.8 para criar o arquivo com usuário e senha utilizado na autenticação.

Comando A.8: Arquivo com usuário e senha

```
1 # htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
```

No arquivo *"cgi.conf"* fica as configurações dos arquivos *cgi* do Nagios. Neste arquivo deve ser configurado os parâmetros de autorização para os usuário da interface Web. As linhas do Comando A.9 foram alterados para o nome do usuário cadastrado no arquivo /usr/local/nagios/etc/htpasswd.users, para que assim o usuário tenha acesso ao sistema de monitoramento.

Comando A.9: Parâmetros de autorização do arquivo cgi.conf

```
1 authorized_for_system_information=nagiosadmin
2 authorized_for_configuration_information=nagiosadmin
3 authorized_for_system_commands=nagiosadmin
4 authorized_for_all_services=nagiosadmin
5 authorized_for_all_hosts=nagiosadmin
6 authorized_for_all_service_commands=nagiosadmin
7 authorized_for_all_host_commands=nagiosadmin
```

Dentro do diretório /usr/local/nagios/ digite o Comando A.10 para validar a configuração realizada.

Comando A.10: Validação da configuração do Nagios

```
1 # bin/nagios -v etc/nagios.cfg
```

Para evitar problemas no experimento é importante que o *firewall* do Sistema Operacional esteja desabilitado. Foi realizada a configuração para desabilitar os serviço como apresentado no Comando A.11

Comando A.11: Desabilitar a inicialização automática

```
1 # chkconfig iptables --level 2345 off
2 # chkconfig ip6tables --level 2345 off
3 # chkconfig httpd --level 2345 on
4 # chkconfig nagios --level 2345 on
```

Como apresentado no Comando A.12, habilite a inicialização automática dos serviços *"httpd"* e *"nagios"*.

Comando A.12: Habilitar a inicialização automática

```
1 # chkconfig httpd --level 2345 on
2 # chkconfig nagios --level 2345 on
```

Apendice B

ALGORITMO DE PRÉ-PROCESSAMENTO MAPLOG

O algoritmo MAPLOG (Matching and Pre-processing Logs) foi implementado como um algoritmo para Mineração de Dados, mais especificamente para realizar o pré-processamento de logs, nesse caso para processar logs da ferramenta de monitoramento Nagios. O algoritmo foi desenvolvido em *Python*, a ideia foi utilizar uma linguagem produtiva.

B.1 Declaração e inicialização das variáveis

Programa B.1: *Implementação do algoritmo MAPLOG (Declaração das variáveis)*

```
import re

# Expressão regular para selecionar o tipo do evento
regex_tag = re.compile('^.\w+.\s(\w+\s?\w+\s\w+).*$')

  ######################################################################
  ###       Expressões regulares para selecionar eventos             ###
  ######################################################################

# HOST NOTIFICATION/SERVICE NOTIFICATION
regex_hn = re.compile('^.(\w+).\s([\w\s]+)\W\s(\w+\W?\w+)\W([\w\s\_
   \.\-]+)\W(\w+)\W(.+?[l])\W(.*)$')
regex_sn = re.compile('^.(\w+).\s([\w\s]+)\W\s(\w+\W?\w+)\W(\w+)\W(\w
   +\s?\w+)\W(\w+)\W(.+?[l])\W(.*)$')

# HOST FLAPPING ALERT/SERVICE FLAPPING ALERT
regex_hfa = re.compile('^.(\w+).\s([\w\s]+)\W\s(\w+)\W(\w+)\W(.*)$')
```

```
regex_sfa = re.compile('^.(\w+).\s([\w\s]+)\W\s(\w+)\W(\w+\s?\w+)\W(\
    w+)\W(.*)$')

# HOST DOWNTIME ALERT/SERVICE DOWNTIME ALERT
regex_hda = re.compile('^.(\w+).\s([\w\s]+)\W\s(\w+)\W(\w+)\W(.*)$')
regex_sda = re.compile('^.(\w+).\s([\w\s]+)\W\s(\w+)\W(\w+\s?\w+)\W(\
    w+)\W(.*)$')

# HOST ALERT/SERVICE ALERT
regex_ha = re.compile('^.(\w+).\s([\w\s]+)\W\s([\w\s\_\.\-]+)\W(\w+)\
    W(\w+)\W([1-5])\W(.*)$')
regex_sa = re.compile('^.(\w+).\s([\w\s]+)\W\s([\w\s\_\.\-]+)\W([\w\s
    \-\.\/]+)\W(\w+)\W(\w+)\W([1-5])\W(.*)$')

# CURRENT SERVICE STATE/CURRENT HOST STATE
regex_chs = re.compile('^.(\w+).\s([\w\s]+)\W\s([\w\s\_\.\-]+)\W(\w+\
    s?\w+)\W(\w+)\W([1-5])\W(.*)$')
regex_css = re.compile('^.(\w+).\s([\w\s]+)\W\s([\w\s\_\.\-]+)\W([\w\
    s\-\/]+)\W(\w+)\W(\w+)\W([1-5])\W(.*)$')

# Definição e inicialização das listas
match_current,cs_state,ch_state,halert,salert,hf_alert,sf_alert,
    hd_alert,sd_alert,hnotif,snotif = [[] for _ in range(11)]
```

B.2 Detalhes da implementação do MAPLOG

Programa B.2: *Implementação do algoritmo MAPLOG (Lógica Principal)*

```
import glob, os, datetime, sys

def tupla_to_list(lista):
  global string_erro
  try:
    tupla = lista.pop()
    unix_data_hora = tupla[0]
  except IndexError:
    print('Erro na lista %s' %file)
  try:
    string = '$ '.join(tupla)  # converte tupla em string
    string_part2 = string.strip(unix_data_hora)
    data_hora = datetime.datetime.fromtimestamp(int(unix_data_hora)).
       strftime('%Y%m%d %H%M%S')
    data_hora_split = data_hora.split()
    string_part1 = '$ '.join(data_hora_split)
    string_full = string_part1+string_part2
    string_erro = '$ '.join(tupla) #usado no except
  except UnboundLocalError:
    print('Erro converte tupla em string %s' %string_erro)
    return ''
  except ValueError:
    print('Erro converte tupla em string %s' %string_erro)
    return ''
  return string_full

#Redireciona a saída padrão para um arquivo de texto
sys.stdout=open('log_erro.txt','w')

os.chdir('D:/log_para_processo')

for file in glob.glob('*.log'):
  arquivo = open(file, 'r')
  for linha in arquivo:
    match_tag = regex_tag.findall(linha)
    if match_tag != []:
      if match_tag[0] == 'CURRENT HOST STATE':
        ch_state.append(tupla_to_list(regex_chs.findall(linha)))
      elif match_tag[0] == 'CURRENT SERVICE STATE':
        cs_state.append(tupla_to_list(regex_css.findall(linha)))
      elif match_tag[0] == 'HOST ALERT':
        halert.append(tupla_to_list(regex_ha.findall(linha)))
      elif match_tag[0] == 'SERVICE ALERT':
        salert.append(tupla_to_list(regex_sa.findall(linha)))
      elif match_tag[0] == 'HOST FLAPPING ALERT':
        hf_alert.append(tupla_to_list(regex_hfa.findall(linha)))
      elif match_tag[0] == 'SERVICE FLAPPING ALERT':
```

```
        sf_alert.append(tupla_to_list(regex_sfa.findall(linha)))
      elif match_tag[0] == 'HOST DOWNTIME ALERT':
        hd_alert.append(tupla_to_list(regex_hda.findall(linha)))
      elif match_tag[0] == 'SERVICE DOWNTIME ALERT':
        sd_alert.append(tupla_to_list(regex_sda.findall(linha)))
      elif match_tag[0] == 'HOST NOTIFICATION':
        hnotif.append(tupla_to_list(regex_hn.findall(linha)))
      elif match_tag[0] == 'SERVICE NOTIFICATION':
        snotif.append(tupla_to_list(regex_sn.findall(linha)))

gera_arquivos()
```

B.3 Pós-processamento do algoritmo MAPLOG

Programa B.3: *Implementação do algoritmo MAPLOG (Pós-processamento)*

```
from Processa_Log.my_functions import *

def gera_arquivos():
  chs_file = open('D:/Logs_PrepProc/CURRENT HOST STATE.txt', 'w')
  css_file = open('D:/Logs_PrepProc/CURRENT SERVICE STATE.txt', 'w')
  ha_file = open('D:/Logs_PrepProc/HOST ALERT.txt', 'w')
  sa_file = open('D:/Logs_PrepProc/SERVICE ALERT.txt', 'w')
  hda_file = open('D:/Logs_PrepProc/HOST DOWNTIME ALERT.txt', 'w')
  sda_file = open('D:/Logs_PrepProc/SERVICE DOWNTIME ALERT.txt', 'w')
  hfa_file = open('D:/Logs_PrepProc/HOST FLAPPING ALERT.txt', 'w')
  sfa_file = open('D:/Logs_PrepProc/SERVICE FLAPPING ALERT.txt', 'w')
  hn_file = open('D:/Logs_PrepProc/HOST NOTIFICATION.txt', 'w')
  sn_file = open('D:/Logs_PrepProc/SERVICE NOTIFICATION.txt', 'w')
  for indice in ch_state:
    chs_file.write(indice + '\n')
  for indice in cs_state:
    css_file.write(indice + '\n')
  for indice in halert:
    ha_file.write(indice + '\n')
  for indice in salert:
    sa_file.write(indice + '\n')
  for indice in hd_alert:
    hda_file.write(indice + '\n')
  for indice in sd_alert:
    sda_file.write(indice + '\n')
  for indice in hf_alert:
    hfa_file.write(indice + '\n')
  for indice in sf_alert:
    sfa_file.write(indice + '\n')
  for indice in hnotif:
    hn_file.write(indice + '\n')
  for indice in snotif:
    sn_file.write(indice + '\n')
  chs_file.close()
  css_file.close()
  ha_file.close()
  sa_file.close()
  hda_file.close()
  sda_file.close()
  hfa_file.close()
  sfa_file.close()
  hn_file.close()
  sn_file.close()
```

Apendice C

ALGORITMO DE TRANSFORMAÇÃO TOTRANS

Para uma correta execução da implementação do algoritmo TOTRANS novos *datsets* devem ser gerados. Para cada host que disponibiliza sistemas e serviços, realizou-se os procedimentos no Comando C.1. O Comando C.1 gera os arquivos "HCA.csv", "SCA.csv" e "HSCA.csv" para o servidor de impressão *Print Server*.

Comandos C.1: ***Manipulação dos arquivos para processamento***

```
1  >awk /DC1_PS1/ "CURRENT HOST STATE.txt" > DC1_PS1_Host.
       txt
2  >awk /DC1_PS1/ "HOST ALERT.txt" >> DC1_PS1_Host.txt
3  >
4  >awk /DC1_PS1/ "CURRENT SERVICE STATE.txt" >
       DC1_PS1_Serv.txt
5  >awk /DC1_PS1/ "SERVICE ALERT.txt" >> DC1_PS1_Serv.txt
6  >
7  >gawk < DC1_PS1_Host.txt -F"$ " "{print $1 \"",\"" $2 \"
       ",\"" $5}" > HCA.csv
8  >gawk < DC1_PS1_Serv.txt -F"$ " "{print $1 \"",\"" $2 \"
       ",\"" $5,"",\"" $6}" > SCA.csv
9  >
10 >gawk < DC1_PS1_Host.txt -F"$ " "{print $1 \"",\"" $2}"
       > HSTE.csv
11 >gawk < DC1_PS1_Serv.txt -F"$ " "{print $1 \"",\"" $2}"
       >> HSTE.csv
```

É importe que após a execução dos comandos, seja feito para o arquivo "HSCA.csv", a remoção de linhas duplicadas e a ordenação dos dados por ordem crescente da data dos registros.

O algoritmo TOTRANS (Transpose To Transform) foi implementado como um algoritmo para Mineração de Dados, mais especificamente para realizar a transformação dos dados. O algoritmo foi desenvolvido em *Python*, a ideia foi utilizar uma linguagem produtiva.

Programa C.2: ***Implementação do algoritmo TOTRANS***

```
import csv

print('Informe o servidor para processamento:')
serv_input = input()

if serv_input != 'SAPPRD AS S1' and serv_input != 'SAPPRD AS S1' \
    and serv_input != 'SAPPRD HA CI' and serv_input != 'PRINT SERV' \
    and serv_input != 'SAPPRD GRC NFe' and serv_input != 'NET SERV':
  print('Informe o nome correto do servidor')
  exit(0)

new_bn = open('NOVO_HSTE.csv', 'w', newline='')
out_bn = csv.writer(new_bn)

host = open('HCA.csv')
csv_host = csv.reader(host)

servicos = open('SCA.csv')
csv_servicos = csv.reader(servicos)

bn = open('HSTE.csv')
csv_bn = csv.reader(bn)
next(csv_bn)    # Pula a primeira linha, começa o loop na segunda

for row in csv_bn:
  host.seek(0)                    # reinicia o csv_host iterator
  next(csv_host)                  # começa na segunda linha
  for linha_h in csv_host:
    if linha_h[0:3] == row[0:3]:
      row[3] = linha_h[3]
    if linha_h[0] > row[0]:
      break
  servicos.seek(0)                # reinicia o csv_servicos iterator
  next(csv_servicos)              # começa na segunda linha
  for linha_s in csv_servicos:
    if linha_s[0] > row[0]:
      break
    if serv_input == 'PRINT SERV':
      row_col4,row_col3,row_col2 = [row[0:3] for _ in range(3)]
      row_col6,row_col5 = [row[0:3] for _ in range(2)]
      row_col2.append('CPU Load')
      row_col3.append('Memory Usage')
      row_col4.append('Disk Usage')
```

```
      row_col5.append('Swap Usage')
      row_col6.append('LPR Port')
      if linha_s[0:4] == row_col2:
        row[4] = linha_s[4]
      if linha_s[0:4] == row_col3:
        row[5] = linha_s[4]
      if linha_s[0:4] == row_col4:
        row[6] = linha_s[4]
      if linha_s[0:4] == row_col5:
        row[7] = linha_s[4]
      if linha_s[0:4] == row_col6:
        row[8] = linha_s[4]
    if serv_input == 'NET SERV':
      row_col4,row_col3,row_col2 = [row[0:3] for _ in range(3)]
      row_col7,row_col6,row_col5 = [row[0:3] for _ in range(3)]
      row_col9,row_col8 = [row[0:3] for _ in range(2)]
      row_col2.append('CPU Load')
      row_col3.append('Memory Usage')
      row_col4.append('Disk Usage')
      row_col5.append('Swap Usage')
      row_col6.append('DHCP')
      row_col7.append('DNS')
      row_col8.append('LDAP')
      row_col9.append('Active Directory')
      if linha_s[0:4] == row_col2:
        row[4] = linha_s[4]
      if linha_s[0:4] == row_col3:
        row[5] = linha_s[4]
      if linha_s[0:4] == row_col4:
        row[6] = linha_s[4]
      if linha_s[0:4] == row_col5:
        row[7] = linha_s[4]
      if linha_s[0:4] == row_col6:
        row[8] = linha_s[4]
      if linha_s[0:4] == row_col7:
        row[9] = linha_s[4]
      if linha_s[0:4] == row_col8:
        row[10] = linha_s[4]
      if linha_s[0:4] == row_col9:
        row[11] = linha_s[4]
    if serv_input == 'SAPPRD AS S1'
      row_col3,row_col2 = [row[0:3] for _ in range(2)]
      row_col5,row_col4 = [row[0:3] for _ in range(2)]
      row_col2.append('FS_GERAL')
      row_col3.append('SAP DW Process')
      row_col4.append('SAP Gateway')
      row_col5.append('SMTP Server')
      if linha_s[0:4] == row_col2:
        row[4] = linha_s[4]
      if linha_s[0:4] == row_col3:
        row[5] = linha_s[4]
```

```
      if linha_s[0:4] == row_col4:
        row[6] = linha_s[4]
      if linha_s[0:4] == row_col5:
        row[7] = linha_s[4]
    if serv_input == 'SAPPRD AS S2'
      row_col3,row_col2 = [row[0:3] for _ in range(2)]
      row_col5,row_col4 = [row[0:3] for _ in range(2)]
      row_col2.append('FS_GERAL')
      row_col3.append('SAP DW Process')
      row_col4.append('SAP Gateway')
      row_col5.append('SMTP Server')
      if linha_s[0:4] == row_col2:
        row[4] = linha_s[4]
      if linha_s[0:4] == row_col3:
        row[5] = linha_s[4]
      if linha_s[0:4] == row_col4:
        row[6] = linha_s[4]
      if linha_s[0:4] == row_col5:
        row[7] = linha_s[4]
    if serv_input == 'SAPPRD HA CI':
      row_col4,row_col3,row_col2 = [row[0:3] for _ in range(3)]
      row_col7,row_col6,row_col5 = [row[0:3] for _ in range(3)]
      row_col10,row_col9,row_col8 = [row[0:3] for _ in range(3)]
      row_col2.append('FS_GERAL')
      row_col3.append('SAP DW Process')
      row_col4.append('SAP Gateway')
      row_col5.append('SMTP Server')
      row_col6.append('FTP Server')
      row_col7.append('Message Server')
      row_col8.append('TNS Listener')
      row_col9.append('FS_ARCHIVE')
      row_col10.append('ORACLE DB')
      if linha_s[0:4] == row_col2:
        row[4] = linha_s[4]
      if linha_s[0:4] == row_col3:
        row[5] = linha_s[4]
      if linha_s[0:4] == row_col4:
        row[6] = linha_s[4]
      if linha_s[0:4] == row_col5:
        row[7] = linha_s[4]
      if linha_s[0:4] == row_col6:
        row[8] = linha_s[4]
      if linha_s[0:4] == row_col7:
        row[9] = linha_s[4]
      if linha_s[0:4] == row_col8:
        row[10] = linha_s[4]
      if linha_s[0:4] == row_col9:
        row[11] = linha_s[4]
      if linha_s[0:4] == row_col10:
        row[12] = linha_s[4]
    if serv_input == 'SAPPRD GRC NFe':
```

```
      row_col4,row_col3,row_col2 = [row[0:3] for _ in range(3)]
      row_col7,row_col6,row_col5 = [row[0:3] for _ in range(3)]
      row_col9,row_col8 = [row[0:3] for _ in range(2)]
      row_col2.append('FS_GERAL')
      row_col3.append('SAP DW Process')
      row_col4.append('SAP Gateway')
      row_col5.append('Message Server')
      row_col6.append('TNS Listener')
      row_col7.append('FS_ARCHIVE')
      row_col8.append('ORACLE DB')
      row_col9.append('JAVA WEB')
      if linha_s[0:4] == row_col2:
        row[4] = linha_s[4]
      if linha_s[0:4] == row_col3:
        row[5] = linha_s[4]
      if linha_s[0:4] == row_col4:
        row[6] = linha_s[4]
      if linha_s[0:4] == row_col5:
        row[7] = linha_s[4]
      if linha_s[0:4] == row_col6:
        row[8] = linha_s[4]
      if linha_s[0:4] == row_col7:
        row[9] = linha_s[4]
      if linha_s[0:4] == row_col8:
        row[10] = linha_s[4]
      if linha_s[0:4] == row_col9:
        row[11] = linha_s[4]
  print(row[0:14])
  out_bn.writerow(row)

host.close()
bn.close()
new_bn.close()
```