UNISINOS

Programa Interdisciplinar de Pós-Graduação em

Computação Aplicada

Mestrado Acadêmico

Josimar Viana Silva

SLAd@CLOUD: Um Sistema de Acordo de Nível de Serviço para Computação em Nuvem

São Leopoldo, 2013

Josimar Viana Silva

SLAD@CLOUD: UM SISTEMA DE ACORDO DE NÍVEL DE SERVIÇO PARA COMPUTAÇÃO EM NUVEM

Dissertação apresentada como requisito parcial para a obtenção do título de Mestre pelo Programa de Pós-Graduação em Computação Aplicada da Universidade do Vale do Rio dos Sinos — UNISINOS

Orientador:
Prof. Dr. Cristiano André da Costa

Co-orientador:
Prof. Dr. Rodrigo da Rosa Righi

São Leopoldo
2013

S586s Silva, Josimar Viana

SLAd@Cloud: um sistema de acordo de nível de serviço para computação em nuvem / por Josimar Viana Silva. -- São Leopoldo, 2013.

93 f. : il. color. ; 30 cm.

Dissertação (mestrado) – Universidade do Vale do Rio dos Sinos, Programa de Pós-Graduação em Computação Aplicada, São Leopoldo, RS, 2013.

Orientação: Prof. Dr. Cristiano André da Costa ; Coorientação: Prof. Dr. Rodrigo da Rosa Righi, Escola Politécnica.

1.Computação em nuvem. 2.Sistemas operacionais distribuídos (Computadores). 3.Rede de computadores – Acesso remoto. 4.Ambientes virtuais compartilhados. 5.Elasticidade. 6.Monitoramento. I.Costa, Cristiano André da. II.Righi, Rodrigo da Rosa. III.Título.

CDU 004.7
004.75:004.451
004.771

Catalogação na publicação:
Bibliotecária Carla Maria Goulart de Moraes – CRB 10/1252

UNISINOS

UNIVERSIDADE DO VALE DO RIO DOS SINOS
Ciências Exatas e Tecnológicas
Programa Interdisciplinar de Pós-Graduação em Computação Aplicada – PIPCA

ATA DE BANCA EXAMINADORA DE DISSERTAÇÃO DE MESTRADO Nº 19/2013

Aluno: **Josimar Viana Silva**

Título da Dissertação: **"ACORDO DE NÍVEL DE SERVIÇO PARA COMPUTAÇÃO EM NUVEM".**

Banca:
Prof. Dr. Cristiano André da Costa
Presidente da Banca e Orientador - UNISINOS
Prof. Dr. Rodrigo da Rosa Righi
Membro da Banca e Coorientador – UNISINOS
Prof. Dr. Jorge Luis Victória Barbosa
Membro da Banca – UNISINOS
Prof. Dr. Philippe Olivier Alexandre Navaux
Membro da Banca - UFRGS

Aos sete dias do mês de agosto do ano de 2013, às 14h reuniu-se na sala 6B423, a Comissão Examinadora de Defesa de Dissertação composta pelos professores: Cristiano André da Costa, Orientador e Presidente, UNISINOS; Rodrigo da Rosa Righi, Coorientador, UNISINOS; Jorge Luis Victória Barbosa, UNISINOS; Philippe Olivier Alexandre Navaux, UFRGS para analisar e avaliar a Dissertação apresentada pelo aluno Josimar Viana Silva
APÓS A APRESENTAÇÃO DO ALUNO, A BANCA DECIDIRAM E DESTACAM QUE O TRABALHO CONSTITUI UMA DISSERTAÇÃO DE MESTRADO. FORAM FEITAS ALGUMAS SUGESTÕES QUE DEVEM SER INCLUÍDAS NO VERSÃO FINAL.

A Banca Examinadora, em cumprimento ao requisito exigido para a obtenção do Título de Mestre em Computação Aplicada, julga esta dissertação:

(X) APROVADA
( ) REPROVADA

São Leopoldo, 07 de agosto de 2013.

Prof. Dr. Cristiano André da Costa

Prof. Dr. Rodrigo da Rosa Righi

Prof. Dr. Jorge Luis Victória Barbosa

Prof. Dr. Philippe Olivier Alexandre Navaux

RESUMO

Computação em nuvem tem se tornado mais popular. Porém, apesar das facilidades e vantagens oferecidas pela computação em nuvem, ainda há obstáculos à sua adoção por parte dos usuários e limitações na prestação dos serviços por parte dos provedores. Para que as nuvens computacionais possam ser utilizadas e ser cruciais para as operações de negócios dos usuários é essencial que estes recebam garantias dos fornecedores na entrega dos serviços. Normalmente, estas garantias são fornecidos através de SLAs (Service Level Agreements ou Acordos de Níveis de Serviço) entre os provedores e consumidores. Os atuais acordos de nível de serviço (SLAs) oferecidos por provedores de computação em nuvem são simples, estáticos e pré-definidos pelos provedores. Estes SLAs não apresentam dinamicidade na negociação e portanto não acompanham o comportamento elástico da nuvem. Serviços em nuvem estão sujeitos à flutuações de carga e violações de SLA são mais propensos a acontecer durante estas flutuações. A natureza destas flutuações são imprevisíveis e, portanto, um SLA estático para suportar essas condições não será eficiente. Neste sentido, o presente trabalho tem como objetivo apresentar um sistema de SLA para computação em nuvem, denominado SLAd@Cloud. Este sistema trata as complexidades inerentes as características da nuvem permitindo a negociação e renegociação dinâmica do SLA baseada nos requisitos de qualidade da aplicação e o monitoramento de métricas específicas e de forma ativa e integrada ao SLA para garantir a qualidade dos serviços prestados nas nuvens computacionais. O sistema trabalha com múltiplas métricas e utilizando um sistema de qualificação baseado na metodologia de comparação par a par de Saaty realiza a priorização de ações quando ocorrem violações de métricas específicas em função dos pesos destas métricas. A avaliação se deu através de implementação de um protótipo em Java que interagiu com o framework Cloudsim, provendo interface de definição e contratação do SLA e o gerenciamento do seu ciclo de vida, mesmo após sua contratação em diferentes modelos de nuvem. Os resultados demonstraram ganhos no tempo de execução, nos custos financeiros e na taxa de sucesso de cumprimento das tarefas, superando o modelo tradicional ou que não trabalha com múltiplas métricas.

Palavras-chave: SLA. Nuvens Computacionais. Monitoramento. Elasticidade. Sistemas Distribuídos.

ABSTRACT

Cloud computing has become more popular. However, though the facilities and advantages offered by cloud computing, there are still obstacles to its adoption by users and limitations in the provision of services by providers. For the cloud computing can be used and be crucial to business operations of the users it is essential that they receive guarantees from suppliers in the delivery of services. Usually, these guarantees are provided through SLAs (Service Level Agreements) between providers and consumers. The current service level agreements (SLAs) offered by cloud computing providers are simple, static and predefined by providers. These SLAs have no dynamic negotiation and does not follow the elastic behavior of the cloud. Cloud services are subject to fluctuations in load and SLA violations are more likely to happen during these fluctuations. The nature of these variations are unpredictable and therefore a static SLA to withstand these conditions will not be efficient. In this sense, this work aims to present a model SLA for cloud computing called SLAd@Cloud. This model treats the complexities inherent characteristics of the cloud allowing the dynamic negotiation and renegotiation of the SLA based on the quality requirements of the application and monitoring of specific metrics and an active and integrated into the SLA to ensure the quality of services provided in the cloud computing. The model works with multiple metrics and using a system of qualification based on the methodology of pairwise comparison of Saaty, it can perform the prioritization of actions to prevent violations of specific metrics depending on the weights of these metrics. The evaluation was made through implementation of a Java prototype that interacted with the framework Cloudsim, providing interface definition and SLA hiring and managing its life cycle, even after their employment in different cloud models. The results showed gains in runtime, the financial cost and success rate of completion of the tasks, overcoming the traditional model or the model does not work with multiple metrics.

Keywords: SLA. Cloud Computing. Monitoring. Elasticity. Distributed Systems.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

API	Application Programming Interface, Interface de Programação de Aplicação
CPU	Central Processing Unit, Processador
HPC	High-performance computing, Computaçao de Alto Desempenho
IaaS	Infraestrutura como Serviço
PaaS	Plataforma como Serviço
QoS	Quality of Service, Qualidade do Serviço
SaaS	Software como Serviço
SLA	Service Level Agreements, Acordos de Nível de Serviço
SLO	Service Level Objectives, Objetivos de Nível de Serviço
SNMP	Simple Network Management Protocol, Protocolo Simples de Gerência de Rede
VM	Virtual Machine, Máquina Virtual
WS	Web Services
XML	eXtensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

A evolução da Web tem tornado fácil criar e fornecer conteúdo dos mais diversos tipos. Construir aplicações web para oferecer produtos e torná-las disponíveis para o público tem se tornado cada vez mais popular. Porém, oferecer produtos ou prestar serviços na Web em larga escala exige muito poder computacional e consequentemente, o custo para operar com alta disponibilidade e baixa latência é muito alto (BRANTNER et al., 2008). Para superar esses problemas, a computação em nuvem tem sido proposta como uma nova forma de explorar os serviços na Web (ROSS; WESTERMAN, 2004). A ideia essencial da computação em nuvem é permitir a transição da computação tradicional para um modelo onde o consumo de recursos computacionais, como armazenamento, processamento, banda de entrada e saída de dados, é realizado através de serviços (KARIN; ANDRé, 2010). Na computação em nuvem esses serviços são ofertados por empresas especializadas e ficam disponíveis em grande escala, a baixo custo sem necessidade de grandes investimentos. Isso permite que pequenas empresas e usuários também possam construir e executar suas aplicações e oferecer produtos ou serviços na Web em larga escala.

Apesar das facilidades e vantagens oferecidas pela computação em nuvem, ainda há obstáculos à sua adoção por parte dos usuários e limitações na prestação dos serviços por parte dos provedores (COSTA et al., 2010). Isto ficou bem demonstrado numa pesquisa conduzida pela Compuware (PREEZ, 2010), que concluiu que 57% das empresas européias pararam seus investimentos em computação em nuvem até que os provedores pudessem dar garantias de melhor desempenho. A pesquisa reportou que 72% das empresas disseram que a sua plataforma de nuvem estava prejudicando a sua capacidade de manter os níveis de serviço estabelecidos com seus clientes, algo que poderia ter um efeito debilitante sobre a receita de suas empresas. Richard Stone, gerente de soluções de computação em nuvem da Compuware, disse que as empresas deveriam implementar garantias rigorosas nos serviços para manter o controle sobre o desempenho (PREEZ, 2010). Ainda segundo a pesquisa, 84% dos entrevistados disseram que essas garantias devem ir além métricas de disponibilidade simples e devem levar em conta a experiência do usuário final.

Assim, é fato que, para que as nuvens computacionais possam ser utilizadas e ser cruciais para as operações de negócios dos usuários, é essencial que estas recebam garantias dos fornecedores na entrega dos serviços. Normalmente, estas garantias são fornecidos através de SLAs (Service Level Agreements ou Acordos de Níveis de Serviço) entre os provedores e consumidores. No entanto, este desafio não é trivial. Uma das características intrínsecas as nuvens é a elasticidade (MELL; GRANCE, 2011). A elasticidade é capacidade de adquirir e libertar recursos num curto período de tempo (ordem de minutos) (COMELLAS; PRESA; FERNANDEZ, 2010). Para que os SLAs se adaptem aos cenários da computação em nuvem, eles devem acompanhar os comportamentos elásticos dos serviços. Para isto, o mecanismo de gestão do SLA deve evitar que ocorram violações, ou se estas ocorrerem, ele deve ativar a renegociação

automática e dinâmica do SLA. Além disso, o mecanismo precisa tratar múltiplas métricas ou múltiplos requisitos e priorizar serviços quando nem todas as métricas puderem ser atendidas. A característica on-demand da computação em nuvem é outro aspecto que aumenta a complexidade da gestão de SLA. A nuvem precisa se ajustar às demandas do usuário e às mudanças e às disponibilidades de recursos automática e dinamicamente(UNDHEIM; CHILWAN; HEEGAARD, 2011). Essas mudanças podem ser detectadas se o SLA estiver fortemente integrado a um monitoramento ativo. De fato, vários desafios para a gestão de SLA autônomo na computação em nuvem ainda permanecem (UNDHEIM; CHILWAN; HEEGAARD, 2011).

Neste sentido, o presente trabalho tem como objetivo apresentar um sistema de SLA para computação em nuvem, denominado SLAd@Cloud. Este sistema é específico para computação em nuvem e trata as complexidades inerentes as características da nuvem. O SLAd@Cloud permite a negociação e renegociação dinâmica do SLA baseada nos requisitos de qualidade da aplicação e o monitoramento de métricas específicas para computação em nuvem de forma ativa e integrada ao SLA para garantir a qualidade dos serviços prestados nas nuvens computacionais. O SLAd@Cloud também trabalha com múltiplas métricas e utilizando um sistema de qualificação, pode realizar a priorização de ações quando ocorrerem violações de métricas específicas, em função dos pesos destas métricas. Esta é a principal contribuição do trabalho.

1.1 Motivação

Computação em nuvem tem se tornado mais popular. Empresas como Amazon, Google, IBM, Yahoo e Microsoft têm investido cada vez mais na oferta de serviços de nuvens computacionais (FOUQUET; NIEDERMAYER; CARLE, 2009). Alguns destes provedores incluem garantias em suas especificações de SLA. Porém o foco, na maioria dos casos, é somente a disponibilidade (UNDHEIM; CHILWAN; HEEGAARD, 2011). No EC2 da Amazon[1], se o serviço não estiver disponível num percentual de tempo de 99,95% calculado no período correspondente a um ano, créditos correspondentes à até 10% do valor pago são devolvidos aos usuários. No Windows Azure da Microsoft[2], o tempo de resposta é considerado, porém isso ocorre apenas em alguns dos serviços. Além de não cobrirem todas as métricas necessárias para serviços de nuvens computacionais, essas propostas de SLA são apenas documentos estáticos.

Os middlewares de computação em nuvem de código aberto, como o Eucalyptus (NURMI et al., 2008) e o OpenNebula (LLORENTE; VOZMEDIANO, 2009), proporcionam funcionalidades que realizam o escalonamento de novas instâncias caso ocorram violações de SLAs, porém esses SLAs são limitados a zonas de disponibilidade e não podem ser negociados dinamicamente.

Assim, fica evidente a necessidade de aumentar o esforço para garantir a qualidade do serviço prestado pelos provedores de nuvens computacionais aos seus usuários.

[1] http://aws.amazon.com/ec2-sla/

[2] http://www.windowsazure.com/en-us/support/legal/sla/

1.2 Objetivo Geral

O presente trabalho propõe um sistema de SLA para computação em nuvem. As características do sistema são: negociação e renegociação dinâmica do SLA baseada nos requisitos de qualidade da aplicação e o monitoramento de métricas específicos para computação em nuvem de forma ativa e integrada ao SLA para garantir a qualidade dos serviços prestados nas nuvens computacionais. A principal contribuição do sistema é que ele trabalha com múltiplas métricas e utiliza um sistema de priorização para manutenção dos SLAs.

1.3 Objetivos Específicos

Para a realização deste trabalho as seguintes tarefas foram necessárias:

1) Realizar o levantamento bibliográfico relacionando com computação em nuvem e SLAs;

2) Pesquisar os trabalhos relacionados;

3) Pesquisar os *frameworks* de código aberto para nuvens computacionais e suas ofertas de SLA;

4) Pesquisar as soluções de nuvens computacionais comerciais e suas ofertas de SLA;

5) Desenvolver de uma proposta de arquitetura de SLA utilizando componentes da computação em nuvem;

6) Comparar a arquitetura proposta com as arquiteturas relacionadas;

7) Realizar de um estudo de caso controlado utilizando um protótipo da arquitetura proposta;

8) Avaliação dos resultados obtidos.

1.4 Questão de pesquisa

A pergunta da pesquisa explorada nesse trabalho é:

"Como assegurar a qualidade dos serviços de computação em nuvem utilizando um acordo de nível de serviço (SLA) que possa ser contratado dinamicamente e adaptar-se automaticamente tanto as necessidades do consumidor quanto a capacidade dos provedores?".

1.5 Estrutura do Texto

Este documento está organizado da seguinte forma: após o capítulo da introdução, no Capítulo 2 são detalhados os conceitos de nuvens computacionais: fundamentos, definições, arquitetura e tecnologias associadas. No Capítulo 3 são apresentados detalhes sobre a definição de

SLA, suas características e sua aplicação em nuvens computacionais, tais como métricas e protocolos específicos para nuvens. Também são apresentados os trabalhos relacionados e como funcionam alguns sistemas de SLAs para computação em nuvem. O Capítulo 4 apresenta o SLAd@Cloud, o sistema de SLA para computação em nuvem. Neste capítulo são descritas as principais características e os detalhes da sua arquitetura.

A avaliação do sistema, por sua vez é apresentada no Capítulo 5, onde são detalhados o ambiente de experimentos, análises de custo de desempenho. Por fim, o Capítulo 6 apresenta a conclusão com contribuições realizadas e trabalhos futuros.

2 FUNDAMENTOS DE COMPUTAÇÃO EM NUVEM

Este capítulo é uma revisão bibliográfica e aborda os conceitos de computação em nuvem, sua definição, fundamentos, arquitetura e tecnologias associadas. Ele está dividido em seis partes e apresentará na Seção 2.1 a definição de Computação em Nuvem. Na Seção 2.2 serão apresentados os modelos de serviço para computação em nuvem. Na Seção 2.3 serão apresentados os modelos de implantação para computação em nuvem. Na Seção 2.4 serão apresentados os principais provedores de nuvens comerciais. Na Seção 2.5 serão apresentados os principais middlewares de código aberto para plataforma de nuvem. E, por fim, a Seção 2.6 resume tópicos descritos neste capítulo.

2.1 Definição

Vaquero descreve 'Nuvens' como grandes estruturas que possuem recursos (*hardware*, plataformas de desenvolvimento e/ou serviços) acessados virtualmente e de fácil utilização. Esses recursos podem ser configurados dinamicamente de modo a se ajustar a uma determinada escala, permitindo uso otimizado. Essas estruturas são exploradas através de um modelo 'pague pelo que se usa' com garantias fornecidas por meio de SLAs (VAQUERO et al., 2008). Recentemente o NIST - National Institute of Standards and Technology (Instituto Nacional de Padrões e Tecnologia) do Ministério do Comércio americano, publicou sua definição de computação em nuvem. Esta definição é uma referência por ter sido amplamente revisada pela comunidade durante anos. Segundo o NIST (MELL; GRANCE, 2011), a computação em nuvem é um modelo para habilitar o acesso por rede, conveniente e sob demanda a 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.

Também de acordo com NIST, cinco são as características essenciais de uma nuvem computacional (MELL; GRANCE, 2011):

- Auto-serviço sob demanda: o consumidor pode provisionar por conta própria recursos de computação, como tempo de servidor e armazenamento em rede, automaticamente e conforme necessário, sem necessitar intervenção humana dos provedores de serviços;

- Amplo acesso por rede: os recursos estão disponíveis através da rede e são acessados através de mecanismos padronizados que promovem o uso por dispositivos clientes leves ou com mais recursos disponíveis de diversas plataformas (como smartphones, tablets, laptops ou desktops);

- Agrupamento de recursos: os recursos de computação do provedor são agrupados para atender a múltiplos consumidores em modalidade multi-inquilinos, com recursos físicos e virtuais diferentes atribuídos dinamicamente e reatribuídos conforme a demanda dos

consumidores. Há independência de localização geográfica, uma vez que o consumidor não controla ou conhece a localização exata dos recursos fornecidos, mas pode ser capaz de especificar a localização em um nível de abstração mais alto (como país, estado ou datacenter);

- Elasticidade rápida: os recursos podem ser provisionados e liberados elasticamente, em alguns casos automaticamente, para rapidamente aumentar ou diminuir de acordo com a demanda. Para o consumidor, os recursos disponíveis para provisionamento muitas vezes parecem ser ilimitados e podem ser alocados em qualquer quantidade e a qualquer tempo;

- Serviço mensurado: os sistemas na nuvem automaticamente controlam e otimizam o uso dos recursos através de medições em um nível de abstração apropriado para o tipo de serviço. A utilização de recursos pode ser monitorada, controlada e informada, gerando transparência tanto para o fornecedor como para o consumidor do serviço utilizado.

Nota-se que apesar de não haver um denominador comum nas definições estudadas, muitas delas concordam que na computação em nuvem os recursos são alocados dinamicamente, de forma elástica, com pagamento somente pelo que se usa e precisam ser monitorados, mensurados e oferecer garantias.

2.2 Modelos de Serviço

Os modelos de serviço de Computação em Nuvem podem variar de acordo com a natureza da tecnologia oferecida pelos provedores. Os modelos arquiteturais são referenciados pelas siglas 'SaaS', 'PaaS' e 'IaaS' (ARMBRUST et al., 2009).

IaaS - Infraestrutura como Serviço é o modelo que consiste no fornecimento de processamento, armazenamento, rede e outros recursos computacionais fundamentais nos quais o usuário tem a capacidade de executar e implementar qualquer software (MELL; GRANCE, 2011). O usuário não gerencia ou controla a infraestrutura da nuvem onde o serviço é prestado, mas tem controle sobre o sistema operacional, aplicações instaladas e um controle limitado sobre os componentes de rede. Os recursos geralmente são disponibilizados através de máquinas virtuais (AKHANI; CHUADHARY; SOMANI, 2011), armazenamento virtual, infraestrutura virtual e outros ativos de hardware como recurso para o cliente, que pode requisitá-los conforme necessário (SOSINSKY, 2011). O provedor de serviço gerencia toda a infraestrutura, enquanto o cliente é responsável apenas pelos outros aspectos da implantação do sistema.

Esta infraestrutura pode ser escalada para cima ou para baixo de acordo com as necessidades do usuário e das aplicações que estão sendo executadas (VELTE; VELTE; ELSENPETER, 2009). Os usuários desses serviços são arquitetos ou analistas de rede ou especialistas em infraestrutura. Pioneira na oferta de serviços IaaS a Amazon AWS, oferta processamento, armazenamento e outros serviços de infraestrutura.

PaaS - Plataforma como Serviço representa um modelo de serviço da computação em nuvem que permite ao consumidor implementar e executar aplicações de infraestrutura na nuvem. Essas aplicações podem ser criadas utilizando linguagens, bibliotecas, serviços e ferramentas suportadas pelo provedor do serviço (MELL; GRANCE, 2011). O usuário não tem acesso ou controle a infraestrutura da nuvem, incluindo rede, servidores, sistema operacional ou armazenamento. Lenk (2009) categoriza a plataforma como serviço em dois grupos: ambientes de programação e ambientes de execução(LENK et al., 2009). Comparando com ambientes tradicionais de desenvolvimento de aplicações, a estratégia de utilização do PaaS pode resultar em um tempo de desenvolvimento reduzido e oferecer dezenas de ferramentas e serviços que permitam uma escalabilidade rápida da aplicação (RIMAL; CHOI; LUMB, 2010). Uma das maneiras mais comuns da utilização da modalidade plataforma como serviço é através de APIs disponibilizadas pelo fornecedor do serviço (FOUQUET; NIEDERMAYER; CARLE, 2009).

SaaS - Software como Serviço é um modelo que proporciona sistemas de software com propósitos específicos que estão disponíveis para os usuários a partir de vários dispositivos por meio de uma interface cliente. No SaaS, os usuários não administram ou controlam a infraestrutura, os usuários acessam direto a aplicação (LENK et al., 2009). Como o software está na Web, ele pode ser acessado pelos usuários de qualquer lugar e a qualquer momento. As únicas responsabilidades dos usuários são o envio e gestão dos dados que a aplicação irá processar e as etapas de interação com a aplicação. Assim, novos recursos podem ser incorporados automaticamente aos sistemas de software sem que os usuários percebam estas ações, tornando transparente a evolução e atualização dos sistemas. Todo o resto é responsabilidade da empresa que fornece o serviço (SOSINSKY, 2011). Para Sumter (2010) esse modelo de serviço pode ser resumido como um serviço que permite o aluguel do software ao usuário (SUMTER, 2010).

Como exemplos de SaaS podemos destacar os serviços de Customer Relationship Management (CRM) como o da Salesforce [3] e o Google Docs (CIURANA, 2009). Os usuários desses serviços são os usuários finais das aplicações, normalmente não especializados em tecnologia. O resumo dos modelos de serviço pode ser visto na Tabela 1.

2.3 Modelos de Implantação

Tratando-se do acesso e disponibilidade de ambientes de computação em nuvem, tem-se diferentes tipos de modelos de implantação. A restrição ou abertura de acesso depende do processo de negócio, do tipo de informação e do nível de visão (SOUSA; MOREIRA; MACHADO, 2010). Pode-se perceber que certas empresas não desejam que todos os usuários possam acessar e utilizar determinados recursos no seu ambientes de computação em nuvem. Neste sentido, surge a necessidade de ambientes mais restritos, onde somente alguns usuários devidamente autorizados possam utilizar os serviços providos. Os modelos de implantação da computação em nuvem podem ser divididos em nuvem pública, privada, comunitária e híbrida (MELL;

[3] http://www.salesforce.com/

Tabela 1: Resumo dos modelos de serviço para Nuvens

Categoria	Características	Tipo de Produto	Exemplos de Provedores	Vantagens	Desvantagens
IaaS	Consumidores tem acesso a hardware ou armazenamento virtualizado no topo do qual podem construir sua infraestrutura.	Máquinas virtuais, gerenciamento de infraestrutura e de armazenamento.	Amazon EC2 e S3, GoGrid, Nirvanix	Pode ser escalada para cima ou para baixo	Requer maior gerenciamento.
PaaS	Os consumidores recebem uma plataforma para o desenvolvimento de aplicações hospedadas na Nuvem.	APIs para programação e *frameworks*.	Google AppEngine, Microsoft Azure, Manjrasoft Aneka.	Tempo de desenvolvimento reduzido e escalabilidade rápida da aplicação.	Não é permitida portabilidade para outros provedores.
SaaS	Os consumidores são providos de aplicações que são acessíveis em qualquer horário e qualquer lugar.	Aplicações Web e servicos web	SalesForce.com (CRM), CA.com (Gerenciamento de Projetos), Google Docs, Google Mail	Novos recursos incorporados sem a intervenção do usuário.	Falta de aplicações específicas e não permite a portabilidade para outros provedores.

Fonte: Adaptado de (BUYYA et al., 2009).

GRANCE, 2011):

Nuvem Pública: A infraestrutura de nuvem é configurada para uso aberto pelo público em geral sob um contrato onde se paga pelo montante utilizado (MELL; GRANCE, 2011). Alguns exemplos são as plataformas da Amazon[4], Google App Engine e Microsoft Windows Azure;

Nuvem Privada: A nuvem privada se configura pela divisão, seguida de isolamento, de uma porção de um ambiente de computação da nuvem pública. Este ambiente fica isolado, e é de utilização dedicada a um único grupo ou entidade. Além disto, um ambiente de computação privada também pode ficar isolado da Internet, utilizando uma rede particular (rede privada virtual - virtual private network ou VPN), e uma rede LAN para o acesso aos serviços (MELL; GRANCE, 2011);

Nuvem Comunitária: A estrutura de nuvem é configurada para uso exclusivo de uma comunidade específica de consumidores de organizações que tenham objetivos em comum (MELL; GRANCE, 2011);

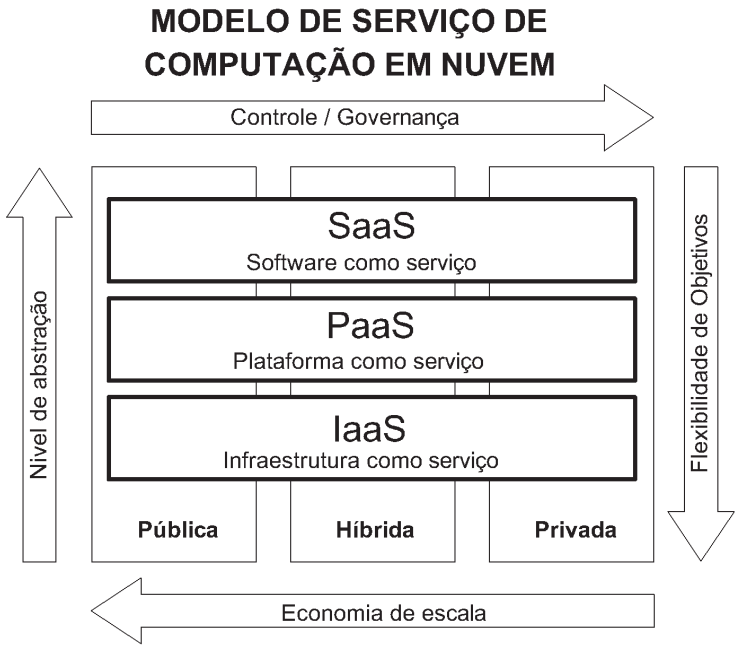
Nuvem Híbrida: O modelo de implantação de nuvem híbrida prevê uma utilização mista, porém integrada, dos três dos outros modelos de implantação de nuvens computacionais, combinando serviços de computação externos com serviços internos (MELL; GRANCE, 2011). Uma necessidade comum pode ser a extensão dinâmica de aplicativos de software internos que são capazes de recorrer a recursos da nuvem externos em momentos de pico, quando mais recursos são necessários para garantir níveis adequados de desempenho ou o cumprimento de SLAs.

A Figura 1 demonstra o relacionamento entre os modelos de serviços e os modelos de implementação na computação em nuvem. A seta 'nível de abstração' demonstra que, quanto mais alto o modelo de serviço, considerando a ordem IaaS, PaaS e SaaS, onde o SaaS é o mais alto, maior é o nível de abstração para o usuário. A seta 'controle/governança' demonstra que, quanto mais restrito o modelo de implantação, considerando a ordem pública, hibrída e pri-

[4]http://aws.amazon.com/

vada, onde privada é a mais restrita, maior o nível de governança ou controle sobre a nuvem. A seta ‘flexibilidade de objetivo’ demonstra que, quanto mais baixo o modelo de serviço, considerando a ordem IaaS, PaaS e SaaS, onde o IaaS é o mais baixo, maior é a flexibilidade de uso. E finalmente a seta ‘economia em escala’ demonstra que, quanto menos restrito o modelo de implementação, considerando a ordem privada, hibrída e pública, onde pública é a menos restrita, maior o nível de economia em escala.

Figura 1: Relacionamento entre os modelos de serviço e modelos de implementação da nuvem computacional.



Fonte: Adaptado de (OGUNMEFUN, 2011).

2.4 Nuvens Comerciais

Apesar de todas as arquiteturas em geral seguirem os modelos apresentados nas seções anteriores, pode-se dizer que cada provedor de serviços possui uma série de características próprias que variam desde a disponibilização ou não de todos os modelos, as linguagens suportadas e outras formas de serviços. Analisando os serviços oferecidos, pode-se apontar como os principais provedores de serviços de nuvens comerciais a Amazon (Amazon Web Services - AWS), a Google (Google App Engine - GAE) e a Microsoft (Microsoft Windows Azure). Essa afirmação é reforçada pelo nome e porte das organizações envolvidas e também pelo potencial investimento na corrida pela liderança no setor. Segundo o Forrester Research (STEFAN RIED, 2011), essas empresas investirão 241 bilhões de dólares em nuvens computacionais até 2020. A seguir, são comentadas as arquiteturas destes três provedores de serviços.

Amazon Web Services - AWS fornece o conjunto mais completo de serviços de apoio a

computação em nuvem (BREITMAN et al., 2010). Das plataformas comerciais é a mais antiga, foi lançada em 2002. O Amazon Web Services é composto por um conjunto de sistemas, dentre os quais pode-se destacar processamento, armazenamento e monitoramento. O Amazon AWS pode ser classificado como IaaS.

O serviço de processamento é o Elastic Compute Cloud (EC2). Ele é responsável pela provisão e gerenciamento de instâncias virtuais de servidores na infraestrutura da Amazon. O EC2 permite um controle completo das instâncias dos sistemas, sendo possível acessar e interagir com cada umas destas, de forma similar a máquinas convencionais. Também é possível escolher as características de cada instância, tais como sistema operacional, pacotes de software e as configurações das máquinas, como CPU, memória e armazenamento.

Microsoft Windows Azure é a proposta da Microsoft para serviços de computação na nuvem. O serviço consiste em uma plataforma (SaaS) para execução de aplicações. A plataforma é composta, essencialmente, por três grandes componentes, que formam o núcleo do serviço; são eles as unidades de computação, o espaço de armazenamento e o Fabric. Este último, é o software responsável por gerenciar e monitorar todos os recursos do datacenter, como servidores, balanceadores de carga, switches, roteadores. Este componente também é responsável pelo processo de gerência do ciclo de vida da aplicação e pela manutenção de níveis satisfatórios de serviços(REDKAR, 2010).

Google App Engine é a plataforma (PaaS) de desenvolvimento que permite que os aplicativos Web nele publicados sejam disponibilizados na infraestrutura do Google. Esta plataforma tem uma série de recursos aos desenvolvedores, como suporte a linguagens Java e Phyton, modelos de aplicação, APIs para integração aos recursos do Google e fornecimento de um ambiente com ajustes e balanceamentos de carga automáticos(SEVERANCE, 2009). A plataforma computacional do Google trabalha sobre um sistema de cotas por desenvolvedor para o qual é possível disponibilizar gratuitamente até dez aplicações com 500MB de limite de armazenamento e cinco milhões de acessos por mês. Estas cotas também possuem limitações, como um timeout de 30 segundos por requisição e um limite de mil linhas de retorno por consultas. Caso haja uma extrapolação dos limites no período de gratuidade, a aplicação poderá ser retirada do ar. O Google oferece também serviços que variam desde sistemas para envio de e-mails e edição de imagens até APIs para autenticação, utilizando uma conta integrada ao Google, agendamento de funções, cache e uma recuperação de URL. Também é possível disponibilizar através do domínio próprio do usuário os serviços do Google conhecidos como Apps, o Gmail ou o Google Docs. O Google Docs [5] é um conjunto de software de comunicação e colaboração para escritório (SaaS). Ele oferece suporte a edição de planilhas eletrônicas, documentos de texto e apresentações utilizando o navegador do usuário.

[5] https://docs.google.com/

2.5 Middlewares para Nuvens

Várias tecnologias de código aberto estão por trás dos serviços de nuvens computacionais comerciais. Logo, existem esforços da comunidade para criação de middlewares de código aberto para plataforma de nuvem. Estes esforços incluem os softwares Eucalyptus, OpenNebula e Nimbus.

Eucalyptus: foi desenvolvido pela Universidade da Califórnia em Santa Bárbara, o Eucalyptus é uma infraestrutura de código aberto que recria o EC2 da Amazon a partir das APIs disponibilizadas pelo serviço da Amazon (SEMPOLINSKI; THAIN, 2010). Desta forma, o usuário pode ter sua própria estrutura em Nuvem (Nuvem Privada) e pode escalonar para o Serviço da Amazon durante um 'pico de demanda'.

Uma Nuvem Privada com Eucalyptus consiste em pelo menos um sistema central e um ou mais clientes ou nós. A arquitetura do Eucalyptus é composta por quatro partes: Cloud Controller (CLC): nível superior que Controla a nuvem como um todo. Storage Controller (Walrus): nível superior que gerencia o tráfego de dados dentro e fora da nuvem. Cluster Controller (CC): nível intermediário que faz a ponte de comunicação entre CLC e NC. Node Controller (NC): nível inferior que controla instâncias das máquinas virtuais nos nós.

O Eucalyptus oferece uma interface baseada em navegador para ajudar a efetuar diversas tarefas de gerenciamento e iniciar máquinas virtuais na sua nuvem de forma facilitada.

OpenNebula é um conjunto de ferramentas de código aberto para criar nuvens privadas e híbridas (SEMPOLINSKI; THAIN, 2010). Assim como o Eucalyptus, ele suporta o Amazon EC2 e também oferece suporte a elasticidade. O OpenNebula também é modelado numa estrutura clássica de cluster. Um nó mestre faz o enfileiramento das tarefas, realiza o escalonamento e envias as tarefas às máquinas do cluster. Os nós de trabalho fornecem o poder computacional para processar as tarefas enviadas pelo nó mestre.

No OpenNebula também é possível adicionar novos nós de trabalho, bem como transferir as instâncias de máquinas virtuais entre os nós. A arquitetura pode ser dividida em três linhas: Ferramentas: ferramentas de gerenciamento desenvolvidas utilizando as interfaces fornecidas pelo Núcleo OpenNebula. Core: contém a máquina virtual principal, armazenamento, redes para VMs e componentes de gerenciamento de *hosts*. E Drivers: drivers para diferentes VMs, drivers de armazenamento e de monitoramento e serviços de nuvem para o núcleo. Além de permitir a criação de nuvens privadas e híbridas, o OpenNebula permite criar nuvens públicas, nas quais os usuários podem acessar a infraestrutura por meio de APIs públicas compatíveis com a AWS. O OpenNebula acrescenta ainda o suporte a uma recente API padronizada pelo Open Grid Forum.

Nimbus, assim como o OpenNebula, implementa as APIs do AWS. Ele tem como alvo especificamente a comunidade científica (SEMPOLINSKI; THAIN, 2010). O Nimbus oferece recursos relevantes à comunidade. Ele pode ser integrado ao Portable Batch System e ao Sun Grid Engine. Ambas ferramentas são amplamente usadas pela comunidade científica para pro-

cessar tarefas em grandes infraestruturas distribuídas. O Nimbus oferece recursos como a crição dinâmica de clusters de máquinas virtuais. Ele implementa um mecanismo de armazenamento compatível com Amazon S3 chamado Cumulus, projetado para ser usado primariamente como repositorio de máquinas virtuais; porém, também é possivel usá-lo de forma independente.

2.5.1 Comparação entre Middlewares

Tabela 2: Comparação dos middlewares de nuvens

Características	Eucalyptus	OpenNebula	Nimbus
Arquitetura	Hierárquica	Modular	Modular
Filosofia	Simular Amazon EC2	Alto nível de customização	Pesquisa científica
Customização	Alguma para administrador, baixa para usuário	Basicamente tudo	Muitas partes, exceto armazenamento imagens e credenciais
Cenário ideal	Grande número de nós para usuários semi-confiáveis	Pequeno número de nós para usuários muito confiáveis	Poucos usuários familiarizados com certificado x509
Rede	dhcpd no cluster controller	Administrador precisa setar manualmente mas tem muitas opções	dhcpd em todos os nós
Opções de Imagens de Disco	Configurado pelo administrador	Em nuvens privadas, as opções estão abertas	Depende da configuração
Armazenamento de Imagens de Disco	Walrus (semelhante ao S3)	Filesystem compartilhado (NFS ou SCP)	Cumulus (recentemente atualizado para GridFTP)
Hipervisores	Xen, KVM, VMWare	Xen, KVM, VMware	Xen, KVM
Características únicas	Interface web	Suporte migração de VM	Nimbus context broker

Fonte: Adaptado de (SEMPOLINSKI; THAIN, 2010).

As principais características dos middlewares de computação em nuvens podem ser vistos na Tabela 2. Pode-se observar a arquitetura do Eucalyptus é predominantemente hierárquica e possui três componentes de alto nível que se comunicam entre si através de redes de comunicação pública e privada. Em contrapartida, o OpenNebula e o Nimbus possuem arquitetura modular, ou seja, utilizam-se de recursos que estão disponiveis em módulos (Drivers, Core e Tools) que podem interoperar.

2.6 Considerações Finais

Esse capítulo apresentou as definições encontradas na literatura para computação em nuvem e os seus principais aspectos relacionados aos modelos de serviço e implentação. Especialmente, foi apresentado, como estes aspectos estão relacionados. Adicionamente foram apresentados os principais provedores de nuvens comerciais e suas características, bem como, os principais middlewares de código aberto que estão por trás dos serviços de nuvens computacionais comerciais. Uma comparação entre estes middlewares também foi apresentada.

3 ANALISANDO SLA PARA COMPUTAÇÃO EM NUVEM

Esse capítulo está dividido em três partes e apresentará na Seção 3.1 a definição de SLA e como os SLAs estão associados as nuvens comerciais e aos middlewares nas nuvens computacionais. Na Seção 3.2 serão apresentados os trabalhos relacionados de SLA para computação em nuvem. E, por fim, a Seção 3.3 mostra os principais tópicos descritos neste capítulo e uma tabela comparativa dos trabalhos estudados.

3.1 SLA

Na definição de computação em nuvem de Vaquero (VAQUERO et al., 2008), os autores chamam atenção para a necessidade de garantias na oferta dos serviços na nuvem com o uso de acordo de nível de serviço (SLA). A computação em nuvem dá aos usuários menos controle na prestação de serviços, assim eles precisam tomar precauções a fim de não sofrer baixo desempenho, longos períodos de inatividade ou perda de dados críticos. O SLA torna-se, portanto, uma parte importante do modelo de prestação de serviço de nuvem (UNDHEIM; CHILWAN; HEEGAARD, 2011).

3.1.1 Definição

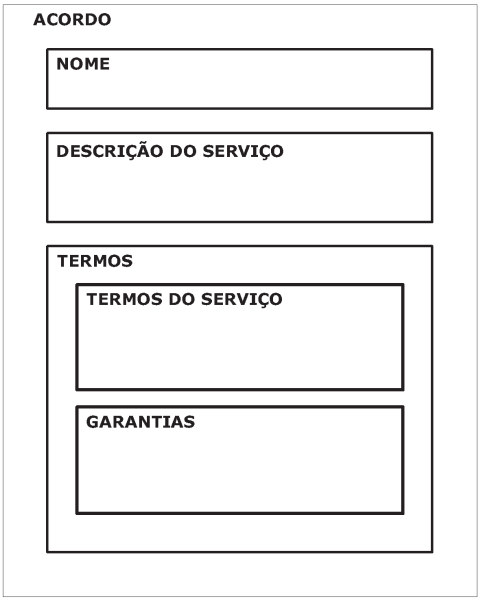
Um SLA é um documento formal, negociado entre as partes envolvidas, na contratação de um serviço (ARENAS; WILSON, 2008). Um SLA tem por objetivo especificar os requisitos mínimos aceitáveis para o serviço proposto e é essencial para o gerenciamento da qualidade dos serviços prestados. Um SLA é elaborado para cada serviço individual e é feito antes da contratação do serviço, antes de poder invocá-lo e utilizá-lo.

Um SLA inclui a decrição dos serviços, as garantias dadas pelos provedores que os serviços serão prestados, as ações e penalidades no caso de violação dessas garantias (ALHAMAD; DILLON; CHANG, 2010). A Figura 2 apresenta a estrutura de um SLA. Com o SLA é possível identificar e definir as necessidades dos usuários, descrever detalhadamente os serviços e a forma como serão entregues, monitorar a qualidade dos serviços, definir quais métricas serão analisadas e informar os resultados, prever como as violações serão tratadas, descrever as responsabilidades do usuários e do provedor e definir as penalidades em caso de descumprimento do SLA por ambas as partes.

Dentro dos SLAs há os SLOs (Service Level Objectives ou Objetivos de Nível de Serviço), que descrevem os tópicos efetivos a serem observados e medidos num SLA (KELLER; LUDWIG, 2003). Um SLO é um requisito que o prestador de serviços deve oferecer. As descrições dos SLOs dependerão muito da arquitetura do serviço a ser prestado (KELLER; LUDWIG, 2003).

O SLA não é estático (SMIT; STROULIA, 2011). A Figura 3 apresenta o ciclo de vida de

Figura 2: Estrutura de um SLA.



Fonte: Adaptado de (ANDRIEUX et al., 2005).

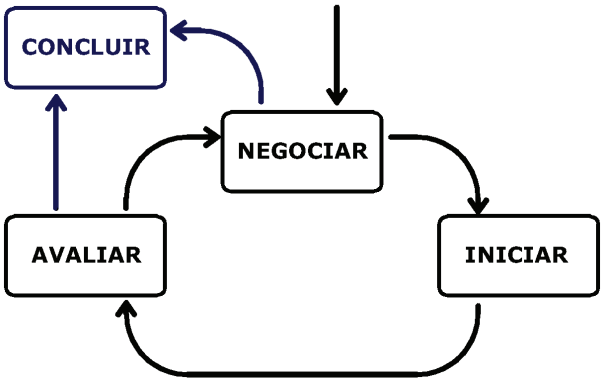
um SLA. As etapas são detalhadas a seguir.

Um SLA inicial é negociado para documentar os requisitos desejados da prestação de serviços. O cliente e o provedor de serviço, que são as partes envolvidas no processo, devem aceitar os termos do SLA que os vincula. Eles também devem detalhar as responsabilidades de cada parte bem como as consequências advindas da violação das normas. Esta negociação pode consumir grande quantidade de tempo representando um longo processo. Em um SLA automático, a disponibilização de uma interface onde seja possível debater pontos conflitantes do SLA pode gerar um rápido acordo, beneficiando mais rapidamente a todos.

O serviço é configurado e iniciado para atender o SLA. Quando um provedor aceita uma acordo, ele precisa colocá-lo em uma fila e utilizar uma política de agendamento para definir a ordem que vai atender os serviços. Além disto, o provedor também precisa considerar como otimizar a utilização dos recursos e como preservar os parâmetros de QoS que são garantidos pelo SLA. Neste cenário, é muito importante considerar a possibilidade da chegada de novas solicitações de serviço e suas respectivas prioridades, mesmo já estando executando outras tarefas, para atendê-las com os recursos que satisfaçam os requisitos da forma mais adequada.

As execuções dos serviços são monitorados e avaliados para assegurar que os termos do SLA serão cumpridos. Considerando o fato que um provedor iniciou a provisão de acesso aos recursos, ele deve monitorar a operacionalização destes recursos. As informações monitoradas podem ser utilizadas para verificar se os atributos de QoS definidos no SLA estão sendo respeitados. Os envolvidos, não devem se interessar somente em saber apenas se uma tarefa está sendo executada corretamente. Outras informações como violação de contrato ou estatísticas de utilização também são relevantes para a verificação do SLA. Uma avaliação consiste em um

Figura 3: Ciclo de vida de um SLA.



Fonte: Adaptado de (SMIT; STROULIA, 2011).

processo que analisa informações pré-determinadas que foram monitoradas e registradas em momentos anteriores. A identificação de informações relevantes é importante para retratar com fidelidade o processo. A qualquer momento, uma das partes do contrato pode querer alterar as políticas de uso dos recursos, normalmente para estar em conformidade com alguma exigência externa advinda de mudanças no contexto. Uma vez que a premissa de alterações sempre vai existir enquanto o sistema estiver executando, é importante considerar este aspecto, mas apesar do ambiente sofrer alterações o comportamento dos processos deve permanecer inalterado. Ou seja, é preciso garantir que após qualquer migração, adição ou remoção de recursos o sistema continuará funcionando corretamente. A utilização dos recursos obviamente deve gerar uma lista descrevendo quais foram utilizados, em que medida e por quanto tempo, bem como relacionando os valores acertados pelo uso de cada um deles de acordo com as definições estabelecidas no SLA. Isto não representa o faturamento propriamente dito, mas é a base para elaboração do prospecto financeiro, que inclusive pode ser desfavorável ao provedor em caso do não cumprimento dos requisitos de QoS.

3.1.2 Monitoramento e otimização de recursos

Para garantir o ciclo de vida do SLA ele precisa ser periodicamente verificado. Para isso ele deve ser monitorado, para garantir que é ainda válido e viável. O resultado do monitoramento informará se o SLA pode ser finalizado ou renegociado. Esta funcionalidade deve levar a uma otimização da execução de uma instância de uma aplicação, tarefa ou serviço baseado nos parâmetros SLA de forma a maximizar a probabilidade de satisfação do SLA. As requisições, submetidas a um SLA, são processadas para selecionar o melhor *host* entre todos os disponíveis. A definição de melhor para um *host* depende do estado de uma série de variáveis no sistema, tais como, os recursos disponíveis, os recursos que são necessários para satisfazer os requisitos do SLA e ainda os objetivos de otimização. Alguns objetivos estão diretamente relacionados ao conhecimento dos requisitos em um SLA, como a redução do tempo de conclusão, a mini-

mização de custos, maximização da probabilidade de sucesso. Enquanto outros objetivos estão relacionados com o estado do sistema, por exemplo, o balanceamento da carga de trabalho.

O processo de monitoração da execução de uma instância de um serviço relacionando-a com as definições de um SLA de forma a concluir se o contrato está sendo respeitado ou não. Durante a execução, se algum parâmetro associado aos SLOs, que são efetivamente os tópicos a serem medidos dentro do SLA, atingir um valor limite, identifica-se uma ameaça de violação e ações de recuperação podem ser ativadas a fim de preservar o SLA ou até mesmo minimizar as consequências de uma violação efetiva do contrato.

Entre as ações de recuperação, pode-se visualizar a re-alocação das tarefas de uma aplicação em recursos disponíveis ou ainda a aquisição de recursos adicionais. Uma opção interessante é enviar alertas de ameaça para o provedor para lhe capacitar a tomar medidas para tentar impedir a violação. Outra opção para esta funcionalidade é a possibilidade de oferecer informações para o usuário que assinou o SLA, informando-o sobre o status atual do SLA, tornando o processo mais transparente.

3.1.3 Emprego na Computação em Nuvem

Do ponto de vista dos usuários, o SLA existente nas nuvens comerciais como a Amazon e a Microsoft são simples porque são estáticos e pré-definidos pelos provedores(WU; BUYYA, 2010). O ciclo de vida do SLA, nestes casos, funciona como será descrito a seguir: O primeiro passo realizado pelo usuário é o de encontrar os fornecedores de serviços de acordo com suas necessidades. Os usuários encontram o provedor através de pesquisa na Internet, e depois exploram o web site dos fornecedores para coletar mais informações. Os provedores de serviços em nuvem oferecem documentos estáticos de SLA. Neste caso, os passos seguintes são a seleção de monitoramento e acompanhamento do serviço que ocorrem normalmente com ferramentas de terceiros. Em seu artigo, Buyya et al. (2010), relaciona os SLAS nas nuvens comerciais. Um resumo pode ser visto na Tabela 3. A Amazon disponibiliza para seus usuários um SLA relacionado somente à disponibilidade(YOUSEFF; BUTRICO; DA SILVA, 2008), ou seja, caso ocorram problemas de degradação de performance de processamento ou saturação da largura de banda o mecanismo de monitoramento informará o consumidor sobre o caso, sem tomar medidas de contingência proativas (GARFINKEL, 2007). No Windows Azure da Microsoft[6], o tempo de resposta é considerado, porém isso ocorre apenas em alguns dos serviços. Além de não cobrirem todas as métricas necessárias para serviços de nuvens computacionais, essas propostas de SLA são apenas documentos estáticos.

Nos middlewares de computação em nuvem, o Eucalyptus proporciona um SLA baseado em regiões ou zonas de disponbilidade (NURMI et al., 2008). Semelhante as zonas de disponibilidade do Amazon AWS, se um SLA for violado é possível migrar os serviços para outras zonas de disponbilidade. Porém, não há monitoramento integrado ao SLA e a migração dos

[6]http://www.windowsazure.com/en-us/support/legal/sla/

Tabela 3: SLA nas nuvens comerciais

Provedor	Compromisso do Serviço	Data criação	Percentual Uptime mensal (MUP)	Percentual crédito serviço
Amazon AWS EC2	"Usaremos nossos esforços para tornar a Amazon EC2 disponível com um percentual de uptime anual de pelo menos 99,95 durante o ano de serviço. No caso da Amazon EC2 não cumprir o compromisso de porcentagem anual de Uptime, você será elegível para receber um crédito de serviço" (AWS EC2 Service Level Agreement).	23/10/2008	99,95	10
Windows Azure	"Os SLAs para computação são separados de armazenamento. Para calcular, nós garantimos que quando você implanta duas ou mais instâncias de função de diferentes falhas e domínios de atualização suas funções voltadas para a Internet terão conectividade externa, pelo menos, 99,95 do tempo. Além disso, vamos monitorar todas as suas instâncias para garantir 99,9 do tempo e vamos detectar dentro de dois minutos, quando uma instância não está em execução e iniciar ações corretivas." (Windows Azure Service Level Agreement)	Não indicado	99,95	10

Fonte: Adaptado de (WU; BUYYA, 2010).

serviços não é disparada automaticamente. No Amazon AWS, não é possível mover instâncias de uma zona de disponbilidade para outra. Essas zonas são fixas durante a vigência da reserva e não podem ser alteradas. Assim, o mecanismo de migração de zonas para o SLA do Eucalyptus, no Amazon AWS não está disponível. Da mesma forma, o monitoramento nos middlewares de computação em nuvem fornecem apenas informações sobre o estado das VMs, se estão pendentes, ativas ou desligadas (CERBELAUD; GARG; HUYLEBROECK, 2009). Para um monitoramento mais efetivo é necessário recuperar informações direto dos hipervisores. Estes, disponibilizam informações como quantidade de CPU utilizada, quantidade de memória disponível e utilizada e tráfego de rede.

Pode-se observar que as soluções existentes de computação em nuvem proporcionam certo nível de SLA e qualidade de serviço, porém não apresentam dinamicidade na negociação do SLA. Os SLAs dessas soluções são estáticos. Serviços em nuvem estão sujeitos à flutuações de carga e violações de SLA são mais propensos a acontecer durante estas flutuações. A natureza destas flutuações são imprevisíveis e, portanto, um SLA estático para suportar essas condições não será eficiente.

Na comunidade acadêmica, existem esforços para criação de protocolos para automatizar a criação e monitoramento de SLAs em serviços Web. O WSLA (Web Service Level Agreement Language) (KELLER; LUDWIG, 2003) é um protocolo para definição de SLAs baseada em Web Services e XML, onde cria-se um XML Schema que engloba a definição das partes envolvidas, as garantias de serviços e a descrição do serviço. O WSLA tem os seguintes componentes principais: Parties, Service Definiton e Obligations. Parties descreve as partes

envolvidas no serviço (cliente ou provedor). Service Definition descreve os serviços ligados ao SLA, representando o entendimento de ambas as partes sobre os parâmetros do serviço descrito. Finalmente, Obligations define o nível de serviço que deve ser garantido com relação aos parâmetros definidos no Service Definition. O WSLA permite o gerenciamento das penalidades ou compensações caso ocorram violações. Porém, ele assume que o SLA já está criado e não permite a negociação para criação do SLA. No documento GFD-R-P.107 (ANDRIEUX et al., 2005), os autores especificam um protocolo baseado em XML, o WS-Agreement (Web Services Agreement Specification), para o estabelecimento de acordos de níveis de serviço e garantias das ofertas entre um provedor e um cliente de serviços Web. Nesta especificação, um acordo envolve múltiplos serviços e incluem atributos para as partes, às referências a acordos anteriores, definições de serviço e termos de garantia. A especificação é dividida em três partes que podem ser utilizados em uma forma de composição: uma estrutura para especificar um acordo, uma estrutura para especificar um modelo de contrato, e um conjunto de tipos de operações para o gerenciamento do ciclo de vida do acordo, incluindo a criação, validade e monitoramento de estados. Estes protocolos, de modo geral, tratam apenas a negociação de um serviço simples com a básica troca de mensagens. A negociação automática de múltiplos serviços e em várias etapas ainda carece de amadurecimento. Além disto, ambos protocolos são específicos para serviços Web e não foram aplicados aos cenários da computação em nuvem (ALHAMAD; DILLON; CHANG, 2010).

3.2 Trabalhos relacionados

Para o desenvolvimento desta proposta de sistema de SLA para computação em nuvem, foram analisados quatro *frameworks*. Para determinar a escolha dos trabalhos analisados foram levadas em consideração os modelos que procurassem fornecer SLAs para serviços de computação em nuvem ou na falta destes que fornecessem SLAs para serviços Web. Estes *frameworks* foram analisados em relação à interface de contratação, nível de especialização das métricas ou SLOs, se é dinâmico, se possui monitoramento integrado e se trabalham com múltiplas métricas. Neste sentido foram escolhidos propostas científicas que possuem características que de alguma forma atendem aos requisitos exigidos, mas diferem em aspectos importantes para a análise comparativa dos modelos.

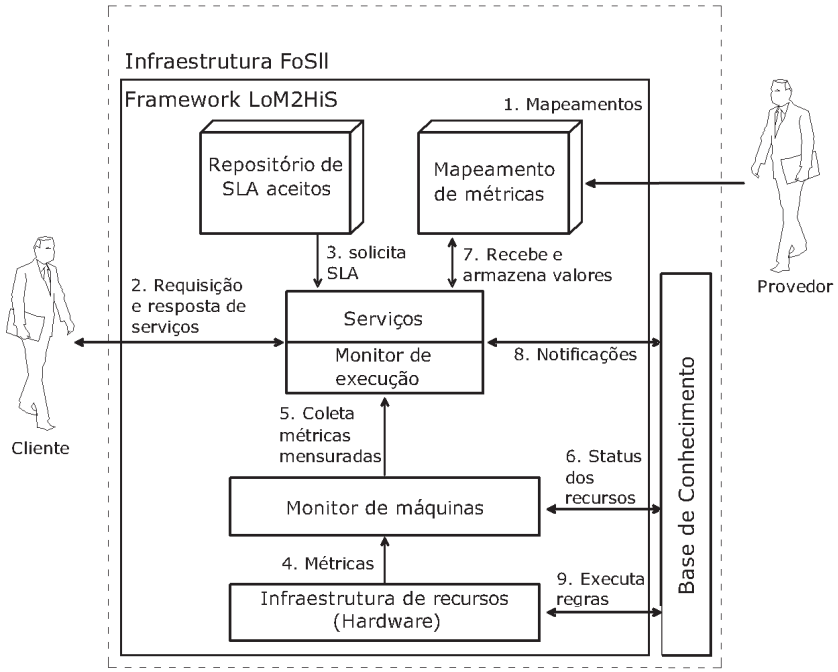
Entre os trabalhos considerados estão o LOM2HIS(EMEAKAROHA et al., 2010a), o DeSVi (EMEAKAROHA et al., 2010b), o SRV(KERTESZ; KECSKEMETI; BRANDIC, 2009) e a arquitetura de SLA para nuvens de pesquisa SaaS (NIEHöRSTER et al., 2010) que foi idealizada para provedores de SaaS de pesquisa científica e aplicações de HPC (*High-performance computing* ou Computação de Alta Performance). Ao final é feita uma comparação entre os modelos estudados. O WSLA (KELLER; LUDWIG, 2003) e o WS-Agreement (ANDRIEUX et al., 2005) foram desconsiderados pois são puramente protocolos de SLA para serviços Web e não foram aplicados aos cenários da computação em nuvem (ALHAMAD; DILLON; CHANG,

2010).

3.2.1 LoM2HIS

O LoM2HIS (EMEAKAROHA et al., 2010a) permite a contratação do SLA a partir de requerimentos QoS, também chamados de requisitos de alto nível, como tempo de resposta ao invés de requisitos de baixo nível da aplicação ou infraestrutura como quantidade de CPU ou memória. A Figura 4 apresenta a arquitetura do LoM2HiS.

Figura 4: Arquitetura LoM2HiS.



Fonte: Adaptado de (EMEAKAROHA et al., 2010a).

O componente de serviço representa a camada de aplicação, onde os serviços são implementados usando um conteiner Web. O monitor de tempo de execução é projetado para monitorar os serviços baseados nos SLAs negociados e acordados. Depois de concordar em termos de SLA, o provedor de serviço cria regras de mapeamento para os mapeamentos LoM2HiS (passo 1) usando Domain Specific Languages(DSLs). DSLs são linguagens simples que podem ser adaptadas a um determinado domínio do problema. Uma vez que o cliente solicita o fornecimento de um serviço acordado (passo 2), o tempo de execução do monitor carrega o SLA de serviço do repositório de SLA acordados (passo 3). O provisionamento de serviço baseia-se nos recursos de infraestrutura disponíveis. As métricas de recursos são medidos por agentes de monitoramento, e as métricas são mensuradas pelo monitor de máquinas (passo 4). O monitor de máquinas extrai as métricas e transmite-as periodicamente para o monitor em tempo de execução (etapa 5) e ao componente de conhecimento (passo 6). Ao receber as métricas, o monitor

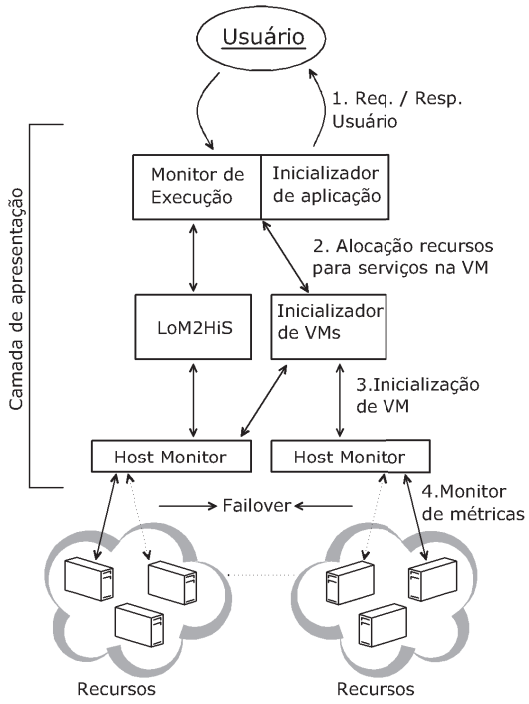
de tempo de execução e o monitor de métricas de baixo nível as compara com base em regras pré-definidas para determinar o SLA equivalente. O resultado de mapeamento é armazenado num repositório (passo 7). O monitor de tempo de execução usa o valores mapeados para monitorar o status da execução dos serviços. Se ocorrerem ameaças de violações futuras do SLA, o componente de conhecimento é notificado para ações preventivas (passo 8). O componente de conhecimento também recebe os limites predefinidos de ameaça (passo 8) para ajustes possíveis devido a mudanças no ambiente em tempo de execução. Este componente aciona uma ação apropriada preventiva para evitar futuras violações de SLA com base no estado de recurso (passo 6) e regras definidas. As decisões do componente de conhecimento (por exemplo, atribuir mais CPU para um *host* virtual) são executados no recursos de infraestrutura (Passo 9). O LoM2HIS não tem mecanismos para negociar o SLA porque ele assume que o processo de negociação do SLA já está concluído e os SLAs acordados já estão armazenados no repositório de provisionamento de serviços.

3.2.2 DeSVI

O DeSVI(EMEAKAROHA et al., 2010b) realiza a detecção de violações de SLA através de monitoramento de recursos de infraestrutura de computação em nuvem. Com base na solicitação do usuário o DeSVi aloca os recursos necessários para um serviço solicitado e organiza o seu *deployment* em um ambiente virtualizado. A detecção de possíveis violações de SLA é baseado no nível de serviço predefinido. Bases de dados de conhecimento são utilizados para gerenciar as violações de SLA. Essas bases de dados de conhecimento são implementadas utilizando técnicas de aprendizagem. Em seu artigo (EMEAKAROHA et al., 2010b), os autores apresentam resultados de execução em ambiente heterogêneos que mostram que o DeSVI é capaz de monitorar e prevenir violações de SLA, considerando cargas de trabalho e intervalos de medição diferentes. O ciclo de vida do serviço inclui atividades como a negociação de SLA, alocação de recursos para tarefas, monitoramento de recursos e detecção de violação do SLA. A arquitetura do DeSVI é apresentada na Figura 5.

A camada superior representa os usuários que solicitam serviço de provisionamento (passo 1) a partir do fornecedor de nuvem. O provedor manipula a solicitação de serviço do usuário com base no SLA negociado e acordado com o usuário. A aplicação que está localizada na mesma camada do tempo de execução monitor, aloca os recursos necessários para o serviço solicitado e organiza a sua implantação nas VMs (passo 2). A implantação de máquinas virtuais e sua configuração são realizadas (passo 3). O monitor anfitrião observa as métricas do *pool* de recursos que compreende máquinas virtuais e *hosts* físicos (passo 4). A relação entre as métricas de recursos (monitorada pelo monitor de *host*) e SLAs (monitorada pelo monitor de tempo de execução) é gerida pela estrutura do LoM2HiS. Na Figura 5 a seta de failover indica redundância no mecanimos de monitoramento. O monitor de *host* é projetado para usar agentes de monitoramento que são incorporados em cada nó do *pool* de recursos para monitorar

Figura 5: Arquitetura DeSVI e interação entre componentes.



Fonte: Adaptado de (EMEAKAROHA et al., 2010b).

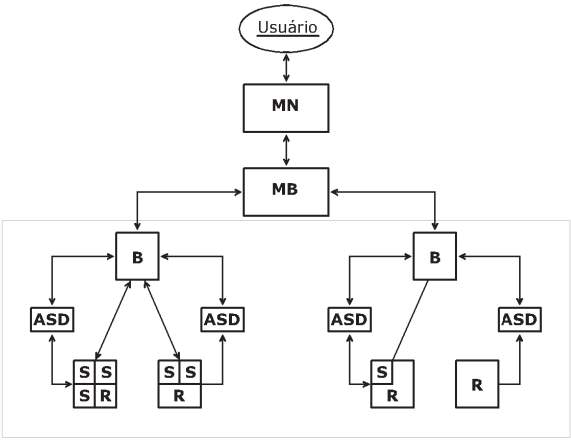
as métricas do nó. Tais agentes de monitoramento transmitem os valores monitorados para os outros agentes do mesmo *pool* de recursos, criando o possibilidade de acessar o status do pool total de recursos de qualquer nó do *pool*. O DeSVI tem capacidade de realizar o monitoramento e detecção de violações de SLA com *deployer* automático de VMs. Porém, o DeSVI não trata aplicações com grande variabilidade de consumo de recursos (EMEAKAROHA et al., 2010b). Além disso, o DeSVI não trata monitoramento e violação de SLA no nível de aplicações (somente infraestrutura).

3.2.3 SRV

O SRV (KERTESZ; KECSKEMETI; BRANDIC, 2009) tem três componentes principais: um componente de meta-negociação para a gestão de um SLA genérico, um componente intermediário para a gestão diversificada e um serviço de deploment automático que usa recursos de virtualização. A Figura 6 apresenta a arquitetura de serviço que demonstra os componentes. São eles: Usuário: Uma pessoa que quer usar um serviço. Meta Negociador (NM): Um componente que gerencia de nível de serviço. É a mediação entre o usuário e o Meta-Broker, seleciona protocolos apropriados para os acordos, negocia a criação de SLA e o tratamento das violações. Meta-Broker (MB): Sua função é selecionar um agente que é capaz de implantar um serviço com as exigências especificadas pelo usuário. Broker (B): Ele interage com os recuros físicos e virtuais, e no caso de o serviço necessário precisar ser implantado, ele interage diretamente com

o ASD. Implantação de Serviço Automático (ASD): Ele instala o serviço necessário no recurso selecionado na demanda. Service (S): O serviço que os usuários querem para implantar e / ou executar. Recursos (R): máquinas físicas, em que as máquinas virtuais podem ser implantados / instalados.

Figura 6: Arquitetura SRV.



Fonte: Adaptado de (KERTESZ; KECSKEMETI; BRANDIC, 2009).

A arquitetura mostra que a negociação do acordo e a implantação do serviço estão intimamente relacionados. O SRV contém um modelo base para negociação do SLA, porém, além de tratar somente a negociação, ele é genérico para ambientes virtualizados e não foi avaliado para ambientes de computação em nuvem.

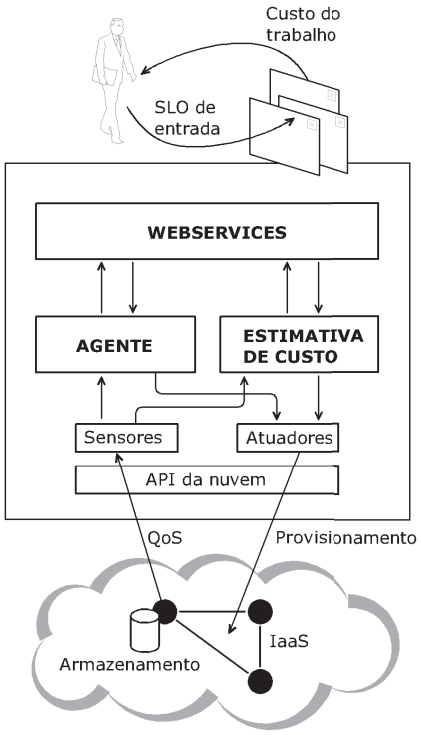
3.2.4 SLA para Nuvens de Pesquisa Científica

Oliver (NIEHöRSTER et al., 2010) propõe uma arquitetura de SLA para nuvens de pesquisa científica SaaS que utiliza algorimos de estimativas que são capazes de avaliar a viabilidade do SLA com antecedência a utilização dos serviços. Neste trabalho, são utilizados princípios de aprendizagem de máquina para criar algoritmos que controlam o provisionamento de máquinas virtuais para manter os SLAs.

A Figura 7 mostra a arquitetura de SLA SaaS para nuvens de pesquisa científica. A nuvem de infraestrutura pode ser vista, na parte inferior da figura, onde um fornecedor de IaaS oferta instâncias de máquinas virtuais dentro do seu centro de processamento de dados. Logo acima, uma camada chamada API de nuvem encapsula o acesso a infraestrutura. O armazenamento que contém os dados dos aplicativos estão localizados no interior da nuvem. Na próxima camada, há os sensores e atuadores. Este componentes interagem com a nuvem e as instâncias em execução. Especialmente os sensores são dependentes da aplicação porque eles são usados para controlar o estado da aplicação. Os atuadores são usados para criar, alterar, configurar e encerrar instâncias de máquinas virtuais. Existe ainda um agente e um módulo de estimativa de custos. O agente

controla o serviço durante a execução e garante o SLA. O módulo de estimativa de custos é inicialmente usado para verificar a viabilidade de solicitações de usuários e estimar os custos. No topo da arquitetura, é fornecida uma interface de serviço web, oferecendo acesso a partir de uma interface de usuário e usado para se comunicar com o agente e módulo de estimativa de custos.

Figura 7: Arquitetura para Nuvens de Pesquisa.



Fonte: Adaptado de (NIEHöRSTER et al., 2010).

A execução da estimativa de custos é feita utilizando heurísticas baseadas em execuções anteriores dos trabalhos. Os custos estimados são apresentados para o usuário e a arquitetura também fornece a alocação de recursos previamente. Após isto, o módulo Agente invoca os trabalhos. A submissão implica que o usuário e o prestador do serviço concordam com o SLA. A tarefa dos agentes é fazer valer o cumprimento da SLO e minimizar os recursos utilizados. Esta arquitetura foi idealizada para provedores de SaaS de pesquisa científica e aplicações de HPC (Computação de Alta Performance). Neste caso, um trabalho normalmente ao ser iniciado não é interrompido até que seja finalizado. Assim, o SLA só pode ser aceito uma vez e fica válido até o final da execução do trabalho. Não há renegociação dinâmica do SLA.

3.2.5 Comparação dos trabalhos estudados

Considerando os trabalhos estudados neste capítulo, podemos destacar alguns aspectos: interface de contratação, nível de especialização das métricas, se é dinâmico, se possui monitoramento integrado, qual modelo de nuvem é atendido, os pontos fortes e pontos fracos.

A **interface de contratação** do SLA diz respeito a como o usuário interage com o modelo na contratação do SLA. Requisitos ou métricas de aplicação normalmente são expressos em linguagem de baixo nível ou técnica, como os pacotes, quantidade de bytes recebidos e enviados. Além disto, estas métricas podem ser mensuradas usando diferentes unidades de medidas. Como o objetivo é comparar e negociar estes requisitos entre os usuários e provedores, unidades de medidas diferentes e expressas em linguagem técnica podem aumentam a complexidade da comparação. Uma interface de expressão de requisitos para parâmetros de QoS, também chamada de alto nível, é expressa de forma a ser menos técnica e como é direcionada a avaliação de qualidade, permite comparações com menor complexidade. O LoM2HIS e arquitetura de SLA para nuvens de pesquisa SaaS utilizam interface de contratação de alto nível. Outros trabalhos demonstraram bons resultados com a utilização deste *framework* (STOEGERER et al., 2011).

As **métricas de contratação** utilizadas pelos modelos podem ser específicas para computação em nuvem ou não. Nenhum dos modelos avaliados utiliza métricas específicas para computação em nuvem. Além disso, os modelos estudados utilizam as métricas de forma isoladas umas das outras. Uma lacuna destes modelos é a não utilização das métricas de forma conjunta. Um sistema de qualificação poderia priorizar ações quando ocorrerem violações de métricas específicas atribuindo diferentes pesos, prioridades ou grau de importância para um grupo ou tipo de métricas.

A **dinamicidade** indica se o modelo permite a negociação e renegociação do SLA de forma dinâmica, acompanhando o comportamento elástico da nuvem. Dos modelos estudados o DeSVI é o único que é projetado para lidar com o ciclo de vida do SLA, incluindo atividades de negociação, alocação de recursos baseado no monitoramento e previsão de violações utilizando técnicas de aprendizagem. Com exceção do SRV, os modelos estudados realizam **monitoramento** dos recursos integrados ao SLA. Um resumo dessa avaliação pode ser visto na Tabela 4, onde ainda são destacados os pontos fortes, pontos fracos e o modelo de nuvem empregado. Pode-se notar que o *framework* LOM2HIS (EMEAKAROHA et al., 2010a) permite a transformação de requisitos de aplicação (baixo nível) em requisitos de qualidade - QoS (alto nível) para a contratação do SLA, porém não permite a negociação do SLA. O DeSVi (EMEAKAROHA et al., 2010b) permite a detecção de violações de SLA através de monitoramento de recursos, porém não trata grande variabilidade de consumo de recursos (elasticidade). E o SRV (KERTESZ; KECSKEMETI; BRANDIC, 2009) é um modelo base para negociação do SLA, porém não foi aplicado a computação em nuvem. E arquitetura de SLA para nuvens de pesquisa SaaS (NIEHöRSTER et al., 2010) foi idealizada para provedores de SaaS de pesquisa científica e aplicações de HPC (Computação de Alta Performance). Neste caso um trabalho normalmente ao ser iniciado não é interrompido até que seja finalizado. Assim, o SLA só pode ser aceito uma vez e fica válido até o final da execução do trabalho. Não há renegociação dinâmica do SLA.

Tabela 4: Trabalhos relacionados.

Modelo	Interface	Métricas	Dinamic.	Monit.	Pontos Fortes	Pontos Fracos	Modelo
LoM2HIS	Alto nível	Apenas infraestrutura e não trabalha com múltiplas métricas	Estático	Sim	Permite realizar a contratação de SLA em linguagem de mais alto nível	Foco pontual na linguagem para parser dos requisitos para métricas e não permite a negociação do SLA	IaaS
DeSVI	Baixo nível	Apenas infraestrutura e não trabalha com múltiplas métricas	Dinâmico mas não suporta elasticidade	Sim	*deployer* automático de VMs	Não trata aplicações com grande variabilidade de consumo de recursos. Não trata monitoramento e violação de SLA no nível de aplicações.	IaaS
SRV	Baixo nível	Genéricas para ambientes virtualizados e não trabalha com múltiplas métricas	Estático	Não	Negociação	É genérico para ambientes virtualizados por isso não trata especificades da nuvem	NA
Arq. Nuvens SaaS	Alto nível	SLA não renegociável e não trabalha com múltiplas métricas	Estático	Sim	Estimativa de Custo	Não trata negociação dinâmica do SLA	SaaS

Fonte: Elaborado pelo autor.

3.3 Considerações Finais

Esse capítulo apresentou os trabalhos encontrados na literatura apresentando os principais aspectos abordados relacionados a definição de SLAs para computação em nuvem. Também contribuiu para a identificação das lacunas nos trabalhos existentes, tais como, o tratamento das complexidades inerentes as características da computação em nuvem, como a elasticidade, a negociação e renegociação dinâmica do SLA baseada nos requisitos de qualidade da aplicação e o monitoramento de métricas específicas e de forma ativa e integrada ao SLA para garantir a qualidade dos serviços prestados nas nuvens computacionais. Na próxima seção, propõe-se a elaboração de um modelo que trata essas lacunas e trabalha com múltiplas métricas e utilizando um sistema de qualificação, pode realizar a priorização de ações quando ocorrerem violações de métricas específicas em função dos pesos destas métricas.

4 SLAD@CLOUD - UM SISTEMA DE SLA PARA COMPUTAÇÃO EM NUVEM

Este capítulo tem por objetivo apresentar o SLAd@Cloud, um sistema para prover SLAs dinamicamente nos serviços de computação em nuvem. Ele foi desenvolvido para realizar a negociação e renegociação do SLA automática de serviços baseadas nas necessidades de QoS da aplicação, utilizar métricas específicas de computação em nuvem e o monitoramento das métricas contratadas com acionamento de ações em caso de necessidade. Em especial, sua principal contribuição diz respeito ao uso de múltiplas métricas.

Este capítulo mostrará como o SLAd@Cloud cumpre tais objetivos da forma que segue. A Seção 4.1 apresenta as decisões de projeto do sistema. A Seção 4.2 apresenta a arquitetura e módulos do SLAd@Cloud em detalhes. A Seção 4.3 apresentará o funcionamento do SLAd@Cloud e por fim, a Seção 4.4 mostra os principais tópicos escritos nesse capítulo.

4.1 Decisões de Projeto

O SLAd@Cloud foi construído para possibilitar a contratação de SLAs com múltiplas e específicas métricas nos serviços de computação em nuvem. Sua principal contribuição é o acompanhamento do comportamento elástico e das características on-demand dos serviços de nuvem utilizando múltiplas métricas num SLA dinâmico e com métricas priorizadas de acordo com as necessidades do usuário. O SLAd@Cloud faz isto oferecendo a possibilidade de negociar e renegociar dinamicamente o SLA e utilizar pesos diferentes para os requisitos. Em especial, esta última ideia é alcançada com a utilização de um sistema de priorização. Esta priorização suporta as etapas de negociação do SLA e ocorre em função dos pesos definidos para cada métrica. As subseções a seguir descrevem como isto ocorre no SLAd@Cloud.

4.1.1 Normalização dos Pesos e Regras de Notificação

Na computação em nuvem há três níveis de serviços, IaaS, PaaS e SaaS. Alhamad et al. (ALHAMAD; DILLON; CHANG, 2010) listou as métricas específicas e mais importantes para um cenário de computação em nuvem. Como pode ser observado na Tabela 5, um modelo de SLA precisa considerar essas métricas e os termos de cada nível de serviço apresentando uma lista de requisitos que podem ser comparáveis em termos de SLA. Nos trabalhos estudados, os termos tipicamente tratam o modelo de nível de serviço IaaS. Porém, nota-se que os termos de SLA para nuvens PaaS e SaaS não são amplamente tratados. Isto ocorre porque, essas métricas específicas são qualitativas, como a "confiabilidade" e "usabilidade da interface" e consequentemente de difícil comparação. Como quantificar estas métricas para torná-las comparáveis?

Para resolver isto o SLAd@Cloud normaliza a mesma escala de valores as métricas requeridas e as métricas ofertadas utilizando a lógica *fuzzy* (ZADEH, 1965). Na lógica *fuzzy* um conjunto de valores expressos numa determinada escala é convertido em outra comparável, ex-

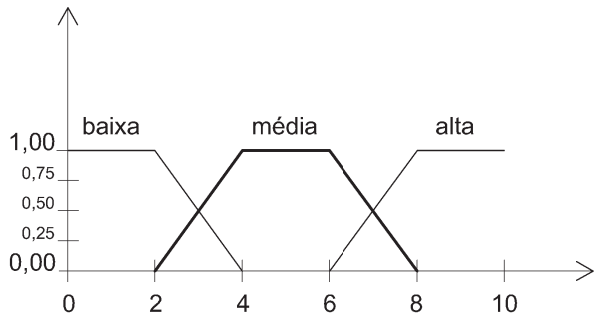
Tabela 5: SLOs ou requisitos de QoS para Nuvens.

Modalidade	Métricas
Infraestrutura como Serviço	Quantidade de CPU, tamanho da memória, tamanho da imagem, largura de banda, tempo de boot, capacidade de armazenamento, número máximo e mínimo de servidores por usuário, tempo para aumentar e diminuir o número de servidores;
Plataforma como Serviço	Integração com serviços de outras plataformas, capacidade para uso de um grande número de usuários (escalabilidade), custo, versões de servidores e browses, número simultâneo de desenvolvedores;
Software como Serviço	Inteface amigável e de fácil uso (usabilidade), capacidade de customização para diferentes tipos de usuário, tempo disponível dos serviços, capacidade para um grande número de usuários;
Gerais	Tempo disponível dos serviços (disponibilidade), tempo de resposta (*performance*), a capacidade de estar em operação em situações diferentes (confiabilidade), quais os métodos de monitoramento, custo do serviço e como é calculado, criptografia, autenticação e autorização (segurança), comunicação (vazão, balanceamento de carga), metodos de apoio e suporte aos serviços, como os dados são armazenados e transferidos (privacidade), localização e legislação.

Fonte: Adaptado de (ALHAMAD; DILLON; CHANG, 2010).

presso numa escala normalizada (0-1). Na primeira etapa da lógica de *fuzzy*, também conhecida como *fuzzificação*, dados de entrada numéricos são transformados em variáveis linguísticas. Para exemplificar essa etapa, pode-se considerar como exemplo de variável *fuzzy*, a métrica "confiabilidade". A essa variável foram atribuídos os valores *fuzzy* "baixa", "média" e "alta". Uma função de pertinência, usada para verificar o quanto um dado pertence a uma determinada classificação faz o mapeamento dos dados numéricos para os valores *fuzzy*, considerando uma escala de 0 até 10. A Figura 8 apresenta a função de pertinência para a métrica "confiabilidade".

Figura 8: Função de pertinência usada na *Fuzzificação*.



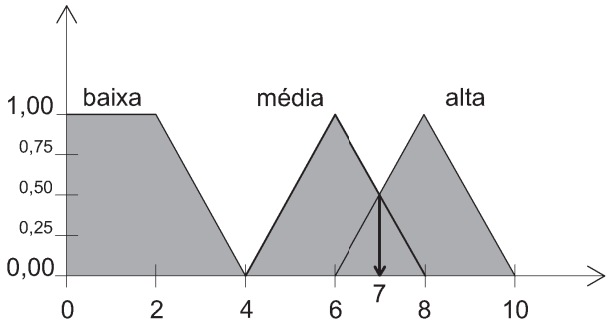
Fonte: Elaborado pelo autor.

A segunda etapa da lógica de *fuzzy* é a inferência. Neste momento, de posse dos graus de pertinência de cada conjunto, são aplicadadas as regras de mapeamento para novos conjuntos. As regras são determinadas pela experiência dos especialistas no assunto em questão ou por decisores do processo. As regras são do tipo Se <*situação*> Então <*ação*>. As regras informadas pelos usuários também alimentam o motor de monitoramento do SLAd@Cloud para acompanhamento e avaliação dos SLAs.

E por fim, a última etapa da lógica de *fuzzy* é a *desfuzzificação*. Ele é ilustrado na Figura 9. Esta é a etapa em que os valores *fuzzy* são convertidos em números reais. Um dos metódos de *desfuzzificação* (COX, 1994) é aplicado para que o processo retorne um valor que melhor

represente a informação constante no conjunto *fuzzy*, tendo assim um conjunto de saída matematicamente definido. Dentre eles o *Centróide* que retorna o centro geométrico dos valores de saída *fuzzy* e o *Média dos Máximos* que retorna a média entre os dois elementos extremos que correspondem aos maiores valores da função de pertinência do conjunto *fuzzy* de saída. Com os valores das métricas expressas em números reais os SLAs já podem ser comparados.

Figura 9: Exemplo de *Desfuzzificação*.



Fonte: Elaborado pelo autor.

Outros métodos para a normalização das métricas foram avaliados, como a variação linear (BEINAT; NIJKAMP, 2007) e o *Z-Score* (BOSSARD, 1999), porém estes precisam de um número grande de amostras para permitir o cálculo de médias e desvios padrões com algum significado. A lógica *fuzzy* mostra-se eficiente para executar funções de controle, configuração, ajuste e combinação de variáveis (FUZZY THINKING: THE NEW SCIENCE OF FUZZY LOGIC, 1994). O tratamento de métricas específicas para computação em nuvem é um dos diferenciaisdesse trabalho em relação aos demais estudados.

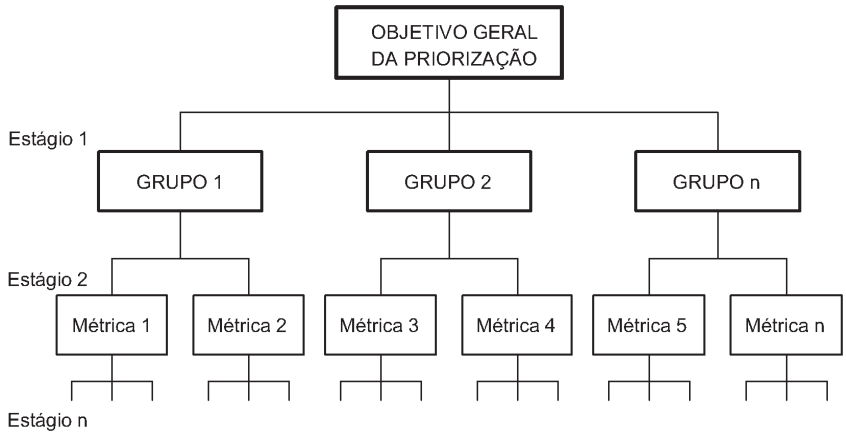
4.1.2 Priorização das métricas

Outra dificuldade encontrada no tratamento com múltiplas métricas é como se deve quantificar a importância relativa de cada uma delas, somado ao fato das mesmas possuírem graus de importância diferentes. Portanto, tornou-se necessário definir no modelo como tratar a importância relativa de cada métrica no processo de contratação e negociação do SLA. Isto é feito atribuindo-se um determinado peso a cada uma das métricas. Embora não se possa afirmar que exista um método consensual para a definição de pesos, encontram-se na literatura várias propostas de procedimentos para esta definição. Existem técnicas de atribuir pesos as métricas, tais como, o ordenamento das métricas, a escala de pontos a distribuição de pontos (MEYERS; GAMST; GUARINO, 2006) e os procedimentos baseados em comparações par a par (SAATY, 1980), muito promissores para a definição e obtenção de pesos para várias métricas.

A abordagem utilizada pelo SLAd@Cloud para a definição dos pesos das métricas foi a metodologia de comparação par a par desenvolvida pelo matemático Thomas Saaty (SAATY, 1980) no contexto de uma técnica denominada *Analytic Hierarchy Process* (AHP). A metodo-

logia tem como objetivo facilitar a solução de problemas complexos relacionados à tomada de decisão. Por meio dessa, pesos e prioridades são derivados a partir de um conjunto de julgamentos subjetivos realizados por avaliadores ou participantes envolvidos no processo. As importâncias relativas dos diversos atributos são traduzidas em um denominador comum através de um processo de comparações pareadas no qual as relevâncias dos atributos são confrontadas duas a duas em uma estrutura hierárquica, conforme descrito Figura 10.

Figura 10: Hierarquização das métricas ou atributos.



Fonte: Elaborado pelo autor.

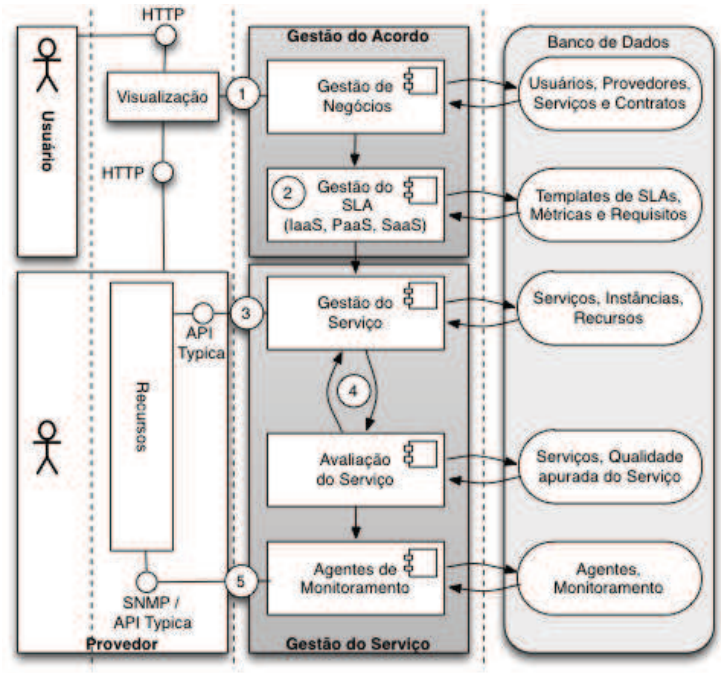
O método ao defrontar-se com um grande número de elementos, que abrangem uma situação complexa, agrega estes elementos em grupos, segundo prioridades comuns. Dada a hierarquia entre os estágios, o método permite identificar a importância relativa de cada indicador dentro de cada estágio. Desta forma, consegue-se, através de um mecanismo gradual, avaliar o peso individual de cada indicador na decisão final a ser tomada, no caso do SLAd@Cloud, a priorização de determinadas métricas.

Outros métodos para definição dos pesos foram avaliados, como árvores de decisão, redes neurais artificiais e algoritmos genéticos, porém estes representam enorme complexidade para o tratamento de múltiplas métricas. As árvores de decisão, por exemplo, tornam-se mais complexas a medida que o número de fatores aumentam, sendo necessário revisá-las a cada mudança, o que consome tempo. No que se refere à avaliação de pesos, sempre que for necessário expressar as prioridades de um determinado grupo de métricas ou critérios, o método de comparação par a par (MALCZEWSKI, 1999) é fortemente recomendado.

4.2 Arquitetura

Esta seção apresenta o SLAd@Cloud, seus componentes, a divisão de responsabilidades entre os módulos e como eles se relacionam. A Figura 11 apresenta a arquitetura geral do SLAd@Cloud utilizando o padrão de modelagem SAP-TAM (SAP, 2007). O primeiro módulo é denominado **Gestão do Acordo**. Ele é responsável pela gestão do SLA. Ele provê a interface de

Figura 11: Arquitetura do SLAd@Cloud.



Fonte: Elaborado pelo autor.

definição, contratação e fornecimento do SLA tanto para os usuários quanto para os provedores. Ele também é o módulo responsável pelo gerenciamento do SLA e do seu ciclo de vida nos três modelos de serviço de nuvem, IaaS, PaaS e SaaS. Estas responsabilidades são realizadas por dois componentes, o componente de Gestão de Negócios e o Componente de Gestão do SLA.

O segundo módulo é denominado **Gestão do Serviço**. Este módulo é responsável pelo gerenciamento do serviço antes e depois da contratação. Uma vez contratado, este módulo inicia e controla a execução dos serviços, realiza o gerenciamento dos recursos e instâncias e também provê informações sobre a qualidade do serviço para preservar os parâmetros de QoS. Estas responsabilidades são realizadas por três componentes, o componente de Gestão do Serviço, o componente de Avaliação do Serviço e os Agentes de Monitoramento. As subseções a seguir descrevem o funcionamento de cada componente.

4.2.1 Componente Gestão de Negócios

O componente Gestão de Negócios provê a **interface de contratação e oferta de SLAs** para os usuários e provedores. Na Figura 11 esta interface pode ser visualizada no número 1. Por meio desta interface os usuários da computação em nuvem fornecem requisitos de SLO por cadastramento em entrada Web ou por meio de Web Services (CABLE et al., 2002). Para este último, o componente Gestão do SLA utiliza o protocolo WS-Agreement (ANDRIEUX et al., 2005) como protocolo de comunicação. Como citado na Seção 3.1.3, o WS-Agreement é um protocolo baseado em XML para o estabelecimento de acordos de níveis de serviço e garantias

das ofertas entre um provedor e um cliente de serviços web.

Figura 12: Representação de um SLA em XML.

```
<wsag:Agreement AgreementId="xs:string">
   <wsag:Name>
       xs:string
   </wsag:Name>
   <wsag:AgreementContext>
      wsag:AgreementContextType
   </wsag:AgreementContext>
   <wsag:Terms>
       wsag:TermCompositorType
   </wsag:Terms>
</wsag:Agreement>
```

Fonte: (ANDRIEUX et al., 2005)

A Figura 12 representa um XML de um SLA utilizando o protocolo WS-Agreement. A criação de atributos adicionais que comportam a criação de um grupo de métricas e dos pesos dessas métricas para a posterior priorização foi incorporada à estrutura do XML do SLA por este trabalho. Isto foi possível pois o WS-Agreement comporta adições de campos extras no corpo do SLA.

Após a escolha do serviço e dos SLOs requeridos pelo usuário, o próximo passo é a classificação dos requisitos em grupos. Esse passo é importante para o tratamento das múltiplas métricas. Esse agrupamento é equivalente ao primeiro estágio da hierarquização das métricas. O módulo de Gestão de Negócios fornece a interface para que o usuário agrupe os SLOs. Um exemplo deste agrupamento pode ser visto na Tabela 6.

Tabela 6: Agrupamento dos requisitos de QoS ou SLOs no SLAd@Cloud.

Grupo	Métricas
Desempenho	Tempo de execução, tempo de boot, tempo de resposta ou tempo para completar uma requisição, quantidade de acessos simultâneos, balanceamento de carga;
Qualidade	Confiabilidade, tipo de criptografia, requisitos de segurança, redundância;
Custo	Menor custo energético, menor custo do serviço;

Fonte: Elaborado pelo Autor.

A partir deste ponto, os usuários fornecem as prioridades ou pesos para cada um dos grupos e cada um dos requisitos individualmente. Para que seja possível realizar a comparação par a par, as métricas são cadastradas junto com os referidos pesos obedecendo a escala proposta por Saaty (SAATY, 1980), composta por nove níveis numéricos, conforme a Tabela 7.

O módulo de Gestão de Negócios garante por meio de sua interface que as métricas sejam comparadas entre si pelo usuário. Por exemplo, quando se compara dois atributos, A_i em relação a A_j, se a entrada for 9, o sistema considera que o atributo A_i é extremamente mais importante que o atributo A_j. No mecanismo de entrada, as métricas e os pesos informados, composto por percepções objetivas e subjetivas dos usuários, são transformados em um equivalentes numéricos. Obviamente, quando se compara o atributo A_i consigo mesmo, a importância

Tabela 7: Escala de Comparação de Critérios.

Valor	Definição	Descrição
1	Igual importância	Os dois critérios contribuem de uma forma idêntica para o objetivo
2	Intermediário	Intermediário entre o de igual importância e pouco mais importante
3	Pouco mais importante	Um critério é um pouco mais importante que o outro
4	Intermediário	Intermediário entre o pouco mais importante e o muito mais importante
5	Muito mais importante	Um critério é claramente mais importante que o outro
6	Intermediário	Intermediário entre o muito mais importante e o bastante mais importante
7	Bastante mais importante	Um dos critérios é predominante para o objetivo
8	Intermediário	Intermediário entre o bastante mais importante e o extremamente mais importante
9	Extremamente mais importante	Um dos critérios é absolutamente predominante para o objetivo

Fonte: Adaptado de (SAATY, 1980).

relativa é equivalente a 1. Se a importância relativa de A_i em relação a A_j é x_{ij}, o sistema estabelece que a comparação inversa do atributo A_j em relação ao atributo A_i implica uma importância relativa igual a $1/x_{ij}$. A inversão no equivalente numérico é consistente com o fato de que, se A_i é em algum grau mais importante que A_j, então A_j é, no mesmo grau, menos importante que A_i.

Dados *n* atributos, tem-se a seguinte matriz de comparações pareadas A=[a_{ij}], onde x_{ij} pertencente a {1; 2; 3; 4; 5; 6; 7; 8; 9} corresponde ao equivalente numérico da importância relativa do atributo A_i em relação ao atributo A_j. De acordo com a Figura 13, os critérios são colocados na mesma ordem nas linhas e nas colunas da matriz, que é preenchida de acordo com a Tabela 7.

Figura 13: Matriz Resultante da Comparação.

	Critério A	Critério B	Critério C	Critério D	Critério E	Critério F
Critério A	1	1/2	2	2	3	1
Critério B	2	1	4	4	6	2
Critério C	1/2	1/4	1	1	2	1/2
Critério D	1/2	1/4	1	1	2	1/2
Critério E	1/3	1/6	1/2	1/2	1	1/3
Critério F	1	1/2	2	2	3	1

Fonte: Elaborado pelo autor.

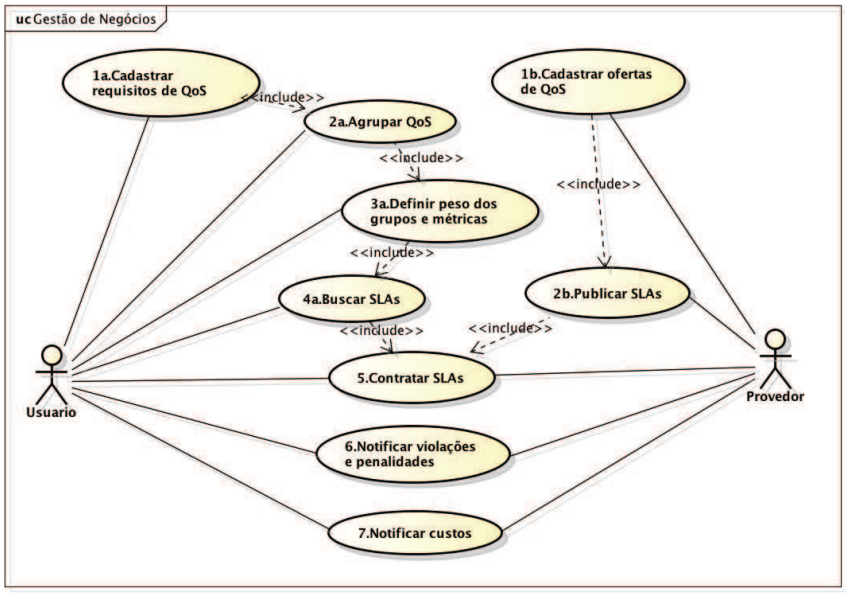
A comparação par a par, entre as diferentes métricas, é armazenada pelo módulo de Gestão de Gestão de Negócios, numa matriz quadrada *n x n*, na qual as métricas estão dispostas na mesma ordem ao longo das linhas e das colunas. Portanto, o valor a_{ij} representa a importância da métrica da linha *i* em relação à métrica da coluna *j*, conforme a Equação 4.1.

$$a_{ij} = \begin{cases} \frac{1}{a_{ij}} & \text{quando } i \neq j, \\ 1 & \text{quando } i = j. \end{cases} \tag{4.1}$$

Esta matriz é recíproca. Por exemplo, se a métrica da linha *i*=1 é cinco vezes mais importante que a métrica da coluna *j*=3, então $a_{13} = 5$ e $a_{31} = 1/5$. Isso implica que apenas a metade triangular superior direita da matriz, destacada na Figura 13, é avaliada já que a outra metade

deriva desta e a diagonal principal assume valores unitários.

Figura 14: Casos de Uso de Gestão de Negócios.



Fonte: Elaborado pelo autor.

O componente de Gestão de Negócios também é responsável por fazer a gestão e manter as relações entre os usuários e os provedores. Ele notifica os custos e as penalidades se ocorrerem violações das garantias registradas no SLA. No SLAd@Cloud, o usuário informa quais métricas serão avaliadas para a violação do SLA e o grau de tolerância (desvio) aceito em cada ocorrência de violação. Estas informações alimentarão o motor de inferências *fuzzy* para tomada de decisões e comporão as regras de monitoramento. A interação do módulo Gestão de Negócios com os usuários e provedores é demonstrado no diagrama de caso de uso na Figura 14 utilizando o padrão UML (Unified Modeling Language). Esta interação dá-se pela execução dos seguintes casos de uso: (1a) Usuário cadastra os requisitos de QoS desejados. (2a) Usuário agrupa os requisitos de QoS. (3a) Usuário define o peso dos grupos e das métricas. Até aqui os casos de uso do usuário são dependentes e são executados em ordem. De forma paralela, o provedor cadastra ofertas de QoS (1b) e publica os SLAs ofertados (2b). Então, os usuários já podem realizar a busca dos SLAs (4a). A contratação ocorre no caso de uso (5). A notificação de violações e penalidades no caso de uso (6) e a notificação de custos no caso de uso (7). Nesses últimos três casos citados, a interação ocorre simultaneamente entre os usuários e provedor.

4.2.2 Componente Gestão do SLA

O componente Gestão do SLA é o mais importante do SLAd@Cloud. Ele é responsável pela **conversão de requisitos de baixo nível em alto nível, negociação e renegociação dinâmica dos SLAs** entre usuários e provedores. Na Figura 11 este componente pode ser visualizado no

número 2. Os requisitos de entrada tanto podem ser de baixo nível, como quantidade de CPU ou memória, quanto de alto nível, como valor de disponibilidade.

Após receber os requisitos fornecidos pelos usuários e provedores, o componente Gestão do SLA faz a **conversão de requisitos** de baixo nível como requisitos de infraestrutura, software e plataforma em requisitos de alto nível, ou requisitos de QoS, os SLOs propriamente ditos. Esta conversão se faz necessária, uma vez que, a falta de ligação entre as métricas de alto nível (QoS) e de baixo nível (aplicação e infraestrutura) é um grande e conhecido obstáculo para a definição e gestão dos SLAs e dificulta o planejamento, previsão e negociação dos SLAs (EMEAKAROHA et al., 2010a). A conversão pode ter diferentes graus de complexidade. Um exemplo da conversão dos requisitos pode ser visto na Tabela 8. Nessa tabela, o tempo parado representa o MTTR (Tempo Médio para Reparo - Mean Time to Repair) que denota o tempo médio gasto para um sistema retornar após uma falha. O tempo em funcionamento representa o MTBF (Tempo Médio entre Falhas - Mean Time Between Failures) que denota o tempo médio entre uma parada e outra do sistema. $T_{entrada}$ é o tempo de resposta para requisitar um serviço e é calculado como $\frac{tamanhodopacote}{bandaentrada-bytesentrada}$ em milisegundos. T_{saida} é o tempo de resposta para responder um serviço e é calculado como $\frac{tamanhodopacote}{bandasaida-bytessaida}$ em milisegundos. Os requisitos também podem ser informados pelos usuários e provedores já na forma de requisitos de QoS.

Tabela 8: Exemplo de conversão de requisitos.

Requisitos ou Recursos	Parâmetro de QoS	Conversão
Tempo Parado, Tempo em Funcionamento	Disponibilidade (D)	$D = 1 - \frac{TempoParado}{TempoemFuncionamento}$
Bytes entrada, Bytes saída, tamanho pacote, banda entrada, banda saída	Tempo de Resposta (T_{total})	$T_{total} = T_{entrada} + T_{saida}(ms)$

Fonte: Adaptado de (EMEAKAROHA et al., 2010a).

Com os requisitos de QoS ou SLOs definidos tanto pelo usuário, quanto pelo provedor, o componente Gestão do SLA tem as informações necessárias para iniciar a **negociação** compondo os SLAs requeridos e os SLAs ofertados. Uma vez que os grupo de requisitos de QoS foram priorizados, estes pesos são informações utilizadas no módulo de Gestão do SLA para a negociação e renegociação de SLA. Por exemplo, requisitos relacionados ao "custo" podem ter um peso maior se o usuário quiser que as decisões de negociação de SLA e priorização de decisões, quando ocorrerem violações, sejam tomadas em favor do menor custo. Por outro lado requisitos de "desempenho" podem ter um peso maior se o usuário quiser que nas decisões, ações sejam tomadas em função desse fator. Para compor o SLA requerido o componente realiza o processamento das múltiplas métricas calculando os pesos de cada critério. Com a matriz A=[a_{ij}] definida é realizada, pelo componente de Gestão do SLA, a avaliação dos pesos que expressam a prioridade de cada usuário, calcula-se o peso final de cada métrica para o processo de contratação do SLA. Esta etapa utiliza os conceitos matemáticos de autovalores e autovetores de Saaty (SAATY, 1980) que estabelece que os pesos relativos w_i de cada métrica pode ser calculada através da Equação 4.2.

$$w_i = \frac{\sum_{j=1}^{n} {w_i}^j}{n}, com\ {w_i}^j = \frac{x_{ij}}{\sum_{i=1}^{n} x_{ij}} \tag{4.2}$$

O resultado é traduzido pela soma dos valores de cada coluna da Matriz *A*, seguida pela divisão de cada elemento da matriz pelo somatório da coluna a que pertence, obtendo-se assim uma matriz de comparação par a par normalizada, e, por fim, a divisão da soma dos *scores* normalizados de cada linha da matriz pelo número de critérios avaliados. A Tabela 9, exibe o resultado de exemplo para sete métricas. Assim, os pesos relativos podem ser comparados cardinalmente. Desta forma, o módulo Gestão do SLA pode avaliar as múltiplas métricas ponderando os indicadores de cada métrica pelos seus pesos relativos e globais.

Tabela 9: Exemplo do resultado da matriz normalizada.

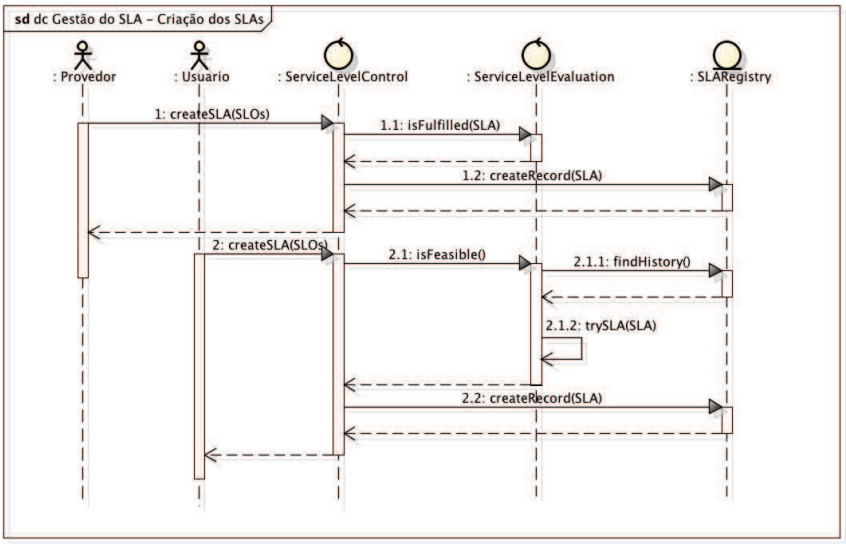
Grupo	Peso do Grupo	Métrica	Peso da Métrica	Peso global
A	50,70%	1	66,87%	33,90%
		2	24,31%	12,33%
		3	8,82%	4,47%
		Total	100,00%	
B	30,84%	4	77,01%	23,75%
		5	22,99%	7,09%
		Total	100,00%	
C	18,46%	6	60,23%	11,12%
		7	39,77%	7,34%
		Total	100,00%	
Total	100%			100%

Fonte: Elaborado pelo Autor.

O diagrama de sequência demonstrado na Figura 15 descreve as interações deste módulo com o usuário e provedor para a negociação dos SLAs. Na negociação, duas verificações são feitas com os respectivos SLAs. Os SLAs ofertados pelos provedores são avaliados para assegurar que as garantias de QoS oferecidas são realizáveis (item 1.1 da Figura 15) e o SLA requerido pelo usuário é avaliado para assegurar que as garantias de QoS solicitadas são realísticas (item 2.1 da Figura 15). Para realizar estas verificações, o SLAd@Cloud realiza buscas nos registros de execuções anteriores (item 2.1.1 da Figura 15), realiza uma execução em escala reduzida do SLA requisitado (item 2.1.2 da Figura 15) ou em último caso, confia nas garantias de QoS nos SLAs ofertados pelo provedor. Se estas duas condições não forem satisfeitas novos requisitos precisam ser informados.

Se estas condições são satisfeitas, o componente Gestão do SLA cria os SLAs e passa a verificar sua equivalência. O diagrama de sequência demonstrado na Figura 16 descreve as interações deste módulo com o usuário e provedor para a negociação dos SLAs. Cada SLA ofertado pelo provedor é comparado com o SLA requerido pelo usuário e o resultado é armazenado numa matriz ordenada B=[b_{ij}]. Esta comparação ocorre novamente utilizando a comparação par a par com a Equação 4.2. Nesta matriz, os SLAs com maior compatibilidade são os primeiros listados. Se mais de um SLA compatível for ofertado ou se o SLA solicitado não for identificado, o componente Gestão do SLA oferta o próximo SLA compatível da matriz. O componente decide pelo SLA que mais se aproxima ou melhor atende os requisitos de QoS de determinado usuário (item 1.2 da Figura 16). Uma vez identificado um SLA compatível com os requisitos de QoS

Figura 15: Diagrama de Sequência de Gestão do SLA - Criação dos SLAs.



Fonte: Elaborado pelo autor.

requeridos pelo usuário, disponível de acordo com os requisitos de QoS ofertado pelo provedor, realizável e realístico, o SLA pode ser **contratado** manual ou automaticamente pelo usuário (item 2 da Figura 16).

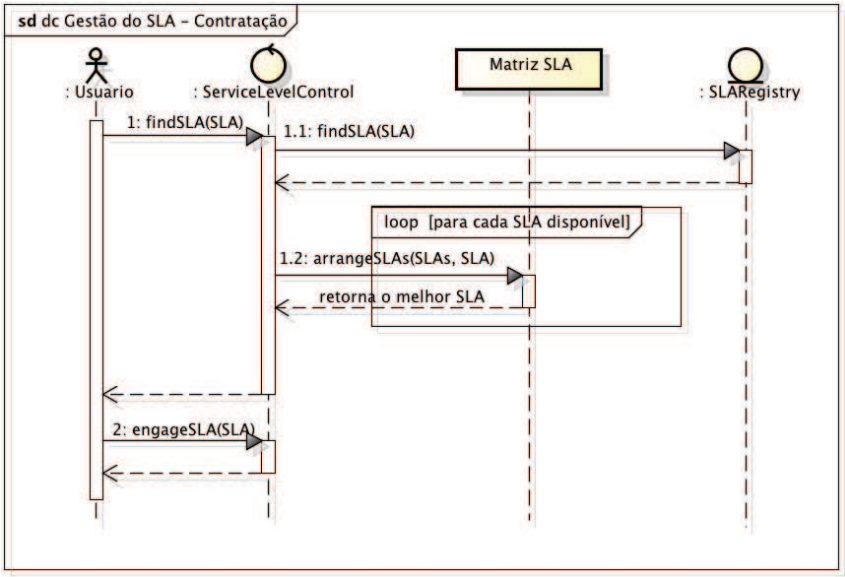
Neste ponto, outro diferencial do SLAd@Cloud pode ser percebido, a **negociação e renegociação dinâmica**. A matriz *B* é dinâmica e é alterada cada vez que um requisito de QoS é alterado no SLA requerido ou novos QoS são ofertados nos SLAs pelos provedores. Então, o módulo refaz o cálculo dos pesos. Considerando a dinamicidade do ciclo de vida do SLA (SMIT; STROULIA, 2011), tanto na contratação, quanto nos momentos de avaliação, se ocorrerem violações do SLA ou o SLA requerido exigir mais recursos aumentando os requisitos de QoS após o início da execução dos serviços, o módulo Gestão do SLA notifica o módulo Gestão de Negócios para comunicação com o usuário. Neste momento, além de ser possível finalizá-lo também é possível renegociar o acordo para um SLA com requisitos de QoS menos ou mais exigentes. Para isto o módulo Gestão do SLA propõe novos SLAs considerando o de maior relevância na matriz ordenada *B*. Assim, o SLAd@Cloud consegue acompanhar o comportamento elástico dos serviços de computação em nuvem. Se os requisitos descritos pelo usuário forem aceitos, a etapa de negociação é encerrada e os serviços inicializados.

4.2.3 Componente Gestão do Serviço

O componente Gestão do Serviço realiza o provisionamento, planeja a utilização dos recursos e é responsável por preservar os parâmetros de QoS do SLA. Para tal, ele acompanha alterações na disponibilidade dos recursos como migração, adição ou remoção de recursos.

Uma vez realizada a negociação do SLA, o componente Gestão do Serviço é responsável pela **alocação de recursos no provedor** respeitando os requisitos acordados e invoca os servi-

Figura 16: Diagrama de Sequência de Gestão do SLA - Contratação dos SLAs.



Fonte: Elaborado pelo autor.

ços. Este módulo conhece os serviços implementados e faz o controle dos elementos necessários para instanciar um serviço. Ele faz a reserva e provisionamento dos recursos necessários para atendimento do SLA. Para comunicar-se com os provedores de nuvem, sejam eles middlewares para nuvem ou provedores comerciais o módulo de Gestão do Serviço utiliza a API Typica[7]. Typica é uma API escrita em Java para acessar uma variedade de serviços Web da Amazon, baseados em AWS, como EC2, SQS, SimpleDB e o DevPay. Esta API pode interagir com o CloudWatch, que é o serviço de monitoramento da Amazon. Na Figura 11 estas ações ocorrem na interface que ser visualizada no número 3.

O módulo de Gestão do Serviço também é o responsável pela manutenção da qualidade dos serviços. Este módulo dispara ações caso sejam detectadas violações do acordo. Para violações de disponibilidade, as ações podem ser a adição de mais recursos para lidar com um determinado número de falhas e para violações de desempenho a movimentação de VMs para uma outra máquina física se a máquina atual está sobrecarregada. Além disso, caso o custo dessas ações seja alto para manter o acordo, o componente de Gestão de Serviço comunica-se com o componente de Gestão de SLA sugerindo a renegociação do SLA. Para estas ações o SLAd@Cloud considera o grupo de requisitos de QoS com maior prioridade definido no SLA.

O módulo de Gestão do Serviço também é responsável por assegurar que a as execuções prévias foram registradas e esses registros estarão disponíveis para experiências futuras, manter os serviços em execução, incluindo ações de alocação de mais VMs ou migração para outros *hosts* e executar ações de gerencialmento.

[7]http://code.google.com/p/typica/

4.2.4 Componente Avaliação do Serviço

Com a contratação do SLA, o monitoramento é iniciado para acompanhar a execução dos serviços e recursos. Esta ação é responsabilidade do componente de Avaliação do Serviço que realiza testes para certificar-se que o SLA está sendo cumprido. O módulo também alimenta com informações do monitoramento os módulos de Gestão de Serviço e Gestão do SLA para as decisões de negociação de SLA. Na Figura 11 estas ações ocorrem nos processos que podem ser visualizados no número 4.

Os serviços de computação em nuvem vão além do modelo de infraestrutura, IaaS. Logo, os componentes de Gestão de Serviços e Avaliação de Serviços também são preparados para gerenciar outros modelos de serviços como PaaS e SaaS citados na Seção 2.2. Isto é feito pela utilização de QoS específicos de nuvem como, por exemplo, capacidade de uso e quantidade de usuários ou desenvolvedores (escalabilidade) e custo do serviço.

4.2.5 Agentes de Monitoramento

Para assegurar o correto funcionamento dos seviços na nuvem, o monitoramento realizado pelo SLAd@Cloud atua diferente do monitoramento tradicional, geralmente centrado em métricas e parâmetros fixos de elementos individuais de infraestrutura. O SLAd@Cloud foca em monitoramento de serviços respeitando as características individuais de cada modelo da nuvem e, principalmente garantindo uma visão do cliente, haja visto o foco no SLA. Assim, os Agentes de Monitoramento realizam o monitoramento através de sondas ativas que acompanham os serviços localizados em ambientes híbridos, virtualizados ou não. Eles fazem isto recolhendo informações dos middlewares de nuvem ou diretamente dos hipervisores. Para esta coleta, os Agentes de Monitoramento utilizam duas formas: o protocolo SNMP (CASE et al., 1990) e a API Typica. Na Figura 11 isto pode ser verificado no número 5. Todo o monitoramento é registrado em log e uma base de conhecimento é gerada para futuras decisões. O processamento das informações coletadas é feito pelo componente de Avaliação do Serviço fora do ambiente de processamento, com o objetivo de reduzir a carga de processamento pelo monitoramento. Cada agente atua independentemente e o nível de informações coletadas pode aumentar ou reduzir, caso um recurso de apresente instável. Os Agentes de Monitoramento levam em consideração, além da disponibilidade, os custos de energia. A redução dos custos de energia tornou-se uma preocupação muito comum no consumo de serviços de tecnologia (VOORSLUYS et al., 2011).

A periodicidade da coleta das informações, o intervalo entre as coletas e quantidade de coletas pode ser configurada pelo usuário no SLA.

4.3 Funcionamento do SLAd@Cloud

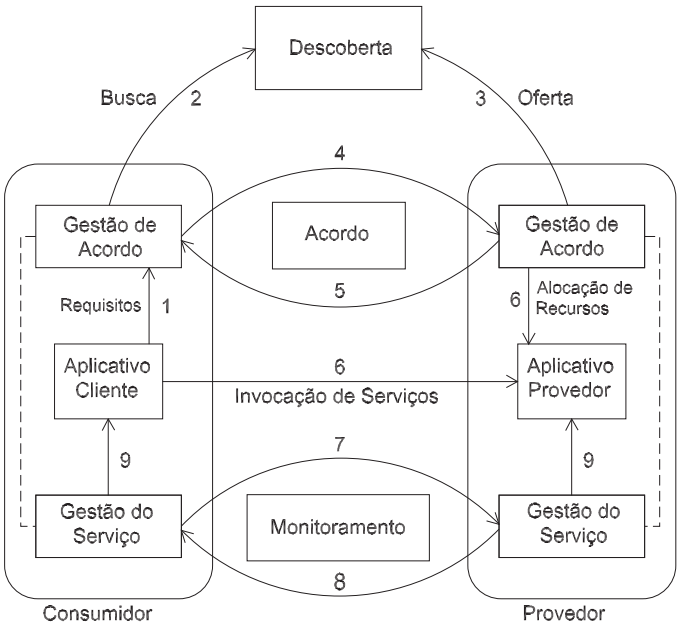
Esta seção apresenta o funcionamento do SLAd@Cloud. A Figura 17, ilustra a integração entre os módulos e o funcionamento de cada etapa do SLAd@Cloud. As etapas são descritas a seguir:

- 1. Os requisitos da aplicação compõem o SLA no módulo de gestão de acordo.
- 2. Este módulo busca os serviços na Nuvem utilizando os requisitos especificados pelos usuários.
- 3. Do lado dos provedores, o módulo de gestão de acordo oferta os serviços considerando sua capacidade de recursos disponíveis em dado momento.
- 4,5. O SLAd@Cloud confronta nos módulos de gestão de acordo dos usuários e provedores a oferta com as requisições e toma decisões para viabilizar o acordo. Várias negociações podem ser necessárias.
- 6. Quando o SLA for definido, os recursos são alocados pelo fornecedor e o serviço invocado pelo usuário.
- 7,8. O SLAd@Cloud permite ao usuário acompanhar o cumprimento do SLA.
- 9. Em caso de falhas no provimento dos serviços, ações automáticas são tomadas para readequação dos serviços, dentre estas ações a alocação de mais recursos por parte do provedor ou a redefinição do SLA para atender as novas condições. Pode-se notar que os módulos de gestão de serviços, tanto no provedor, quando para o usuário, se comunicam (linha tracejada) com os módulos de gestão do acordo para viabilizar essas ações.

4.4 Considerações Finais

Este capítulo apresentou como o SLAd@Cloud tratou as lacunas identificadas nos trabalhos existentes. O SLAd@Cloud propõe um sistema para prover SLA dinâmicos para serviços de computação em nuvem. Utilizando um motor de priorização, baseado na metodologia de comparação par a par (SAATY, 1980), o SLAd@Cloud é capaz de tratar **múltiplas métricas** simultaneamente e identificar o peso (importância) de cada requisito de SLO para o usuário. Este peso é utilizado nas decisões de negociação e renegociação do SLA que ocorrem **dinamicamente** respeitando o ciclo de vida do SLA. Esta dinamicidade é oferecida pelo SLAd@Cloud através de uma matriz que classifica os SLAs ofertados de acordo com a importância dos requisitos requeridos pelo usuário. Esta matriz é atualizada no momento da contratação, renegociação e nos momentos de avaliação do SLA, quando o SLAd@Cloud monitora a execução dos serviços

Figura 17: Funcionamento do SLAd@Cloud.



Fonte: Elaborado pelo autor.

e recursos e provê informações que permitem avaliar se os termos do SLA estão sendo cumpridos. Com esta constante atualização, o SLAd@Cloud garante ofertar o SLA mais aderente aos requisitos do usuário de forma frequente. As métricas utilizadas pelo SLAd@Cloud são **específicas para computação em nuvem**. Um motor de inferências *fuzzy* atua na avaliação destas métricas e na tomada de decisões com informações do monitoramento realizado. Desta forma, mesmo métricas qualitativas como usabilidade são tratadas pelo SLAd@Cloud. Adicionamente o SLAd@Cloud fornece uma interface de contratação e oferta de SLA que permite aos usuários trabalhar com requisitos de QoS de baixo ou alto nível. Caso os requisitos de QoS de baixo nível sejam informados o SLAd@Cloud **faz a conversão para requisitos de alto nível**. Essa interface contempla cadastramento em tela Web ou por meio de Web Services.

5 AVALIAÇÃO DO SLAD@CLOUD

Este capítulo tem por objetivo apresentar as estratégias que foram utilizadas para avaliar o SLAd@Cloud e os resultados obtidos. O presente capítulo é segmentado em quatro seções. A Seção 5.1 apresenta as ferramentas utilizadas na avaliação. A Seção 5.2 apresenta a avaliação do SLAd@Cloud quanto a priorização das métricas e negociação do SLA para contratação. A Seção 5.3 apresenta como o SLAd@Cloud se comportou nas avaliações quanto a dinamicidade do SLA e, por fim, a Seção 5.4 discute os principais resultados obtidos.

5.1 Plataforma de desenvolvimento

Para avaliação do modelo foi codificado um protótipo através do emprego da plataforma de desenvolvimento Java. A escolha deu-se devido a característica multiplataforma da linguagem, o que possibilita sua execução em diferentes sistemas operacionais sem a necessidade de recompilação. A linguagem Java oferece classes para a escrita de aplicações distribuídas que escondem detalhes técnicos de desenvolvedores. Além disto, Java possui um sistema de gerenciamento de memória automático comumente denominado *garbage collector*, que isenta o desenvolvedor de atividades complexas. Por fim, Java tem se tornado uma opção crescente para a computação distribuída e de alto desempenho (TABOADA; TOURIñO; DOALLO, 2009).

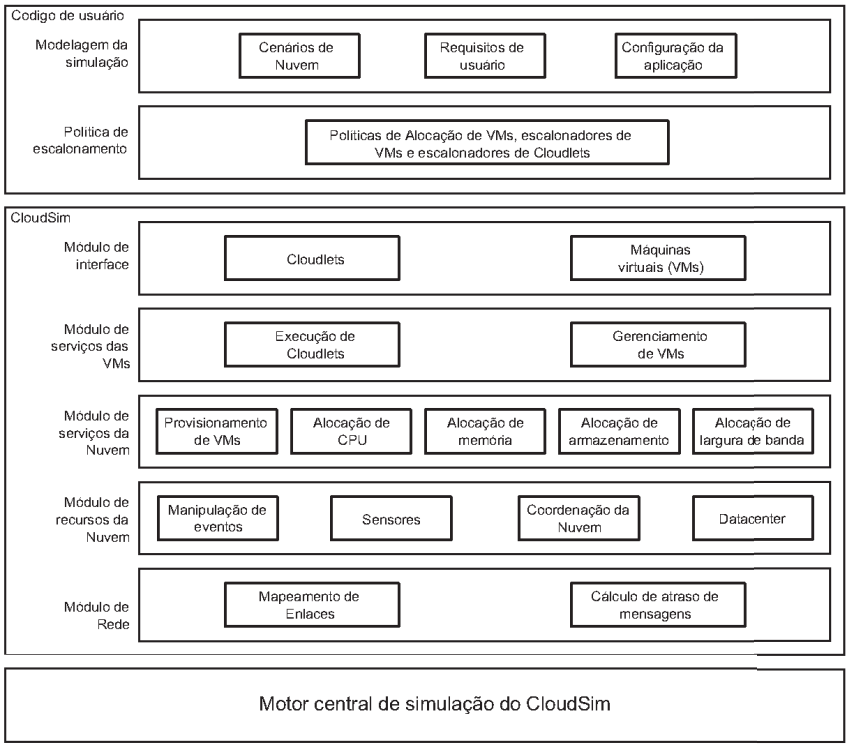
Os serviços de computação em nuvem tem requisitos, configuração, composição e mecanismos de provisionamento complexos. Avaliar o desempenho de fornecimento de serviços, modelos de cargas de trabalho e recursos em estruturas físicas reais de nuvens são tarefas extremamente difíceis (CALHEIROS et al., 2011). Uma solução para a utilização de estruturas complexas no ambiente computacional é utilização de modelagem através de simuladores (LAW; KELTON, 2000). Os simuladores permitem realizar avaliações de modelos de forma confiável e com um custo muito menor, se comparados aos custos de utilização de estruturas físicas reais. Para simulação dos ambientes de nuvem nesta avaliação, foram utilizados os recursos oferecidos pelo *framework* CloudSim (CALHEIROS et al., 2011).

O CloudSim visa oferecer os recursos necessários para a simulação de ambientes computacionais em nuvem. Desenvolvido em Java e licenciado pela *General Public Licence* (GPL), é extensível, facilmente adaptável e permite a criação de simulações em grande escala com alto grau de customização (CALHEIROS et al., 2011). O CloudSim tem sido muito utilizado pela comunidade científica (JUNG; KIM, 2012) (SHI; JIANG; YE, 2011) (CANEDO et al., 2012) (LI et al., 2011).

A Figura 18 ilustra as camadas que compõem a arquitetura do CloudSim. No nível mais inferior, observa-se o motor de simulação, responsável pelas operações de criação, gerenciamento e exclusão das entidades simuladas. A camada seguinte ilustra as principais classes que compõem o *framework*. No módulo de rede são realizados o mapeamento de enlaces entre *datacenters* (provedores de nuvem) e usuários, e o cálculo de atraso das mensagens trocadas entre

os mesmos. O módulo de recursos da nuvem realiza a manipulação e coordenação dos eventos da simulação, além de gerenciar os dados relativos a infraestrutura oferecida por meio dos *datacenters* simulados. Em seguida, o módulo de serviços da nuvem ilustra as ações de provisão de máquinas virtuais e alocação de recursos como memória de sistema, processamento, armazenamento e largura de banda. Como sua função consiste no gerenciamento destes recursos, ressalta-se sua grande integração com o módulo anterior. Observa-se então o módulo de serviços das máquinas virtuais, onde são realizadas a gerência e a execução das tarefas enviadas pelos clientes, denominadas *cloudlets*. Por fim, a comunicação das entidades que compõem a nuvem com os clientes que utilizam seus recursos é feita por meio do módulo de interface, no qual máquinas virtuais e *cloudlets* podem ser manipuladas.

Figura 18: Arquitetura do CloudSim.

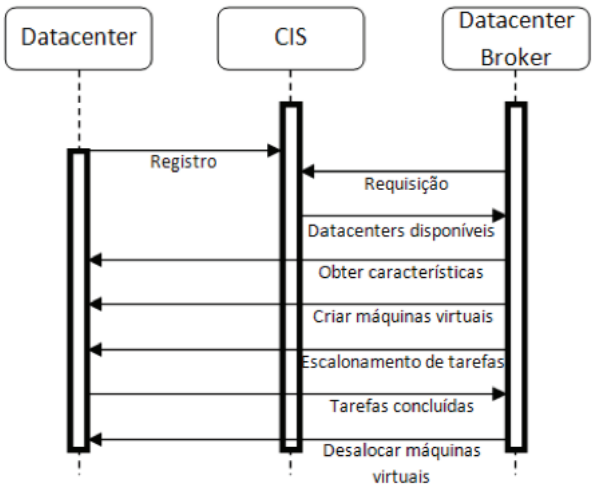


Fonte: Adaptado de (CALHEIROS et al., 2011).

A última camada da Figura 18 representa o código que o usuário do *framework* deve implementar para a criação dos ambientes de simulação. O módulo de política de escalonamento indica a criação das políticas de decisão e escalonadores que nortearão os processos de simulação. Além disso, o ambiente a ser simulado pode ser detalhadamente modelado, especificando-se os recursos que compõem a nuvem e os perfis de utilização referentes a cada um dos clientes. Além dos mecanismos de decisão denominados políticas de *broker*, o CloudSim permite a implementação de políticas de alocação de máquinas virtuais entre os *hosts* de um mesmo *datacenter*, escalonadores de máquinas virtuais em *hosts* e escalonadores de *cloudlets* em máquinas virtuais.

A Figura 19 ilustra o fluxo de comunicação entre as principais entidades de simulação do

Figura 19: Fluxo de comunicação do CloudSim.



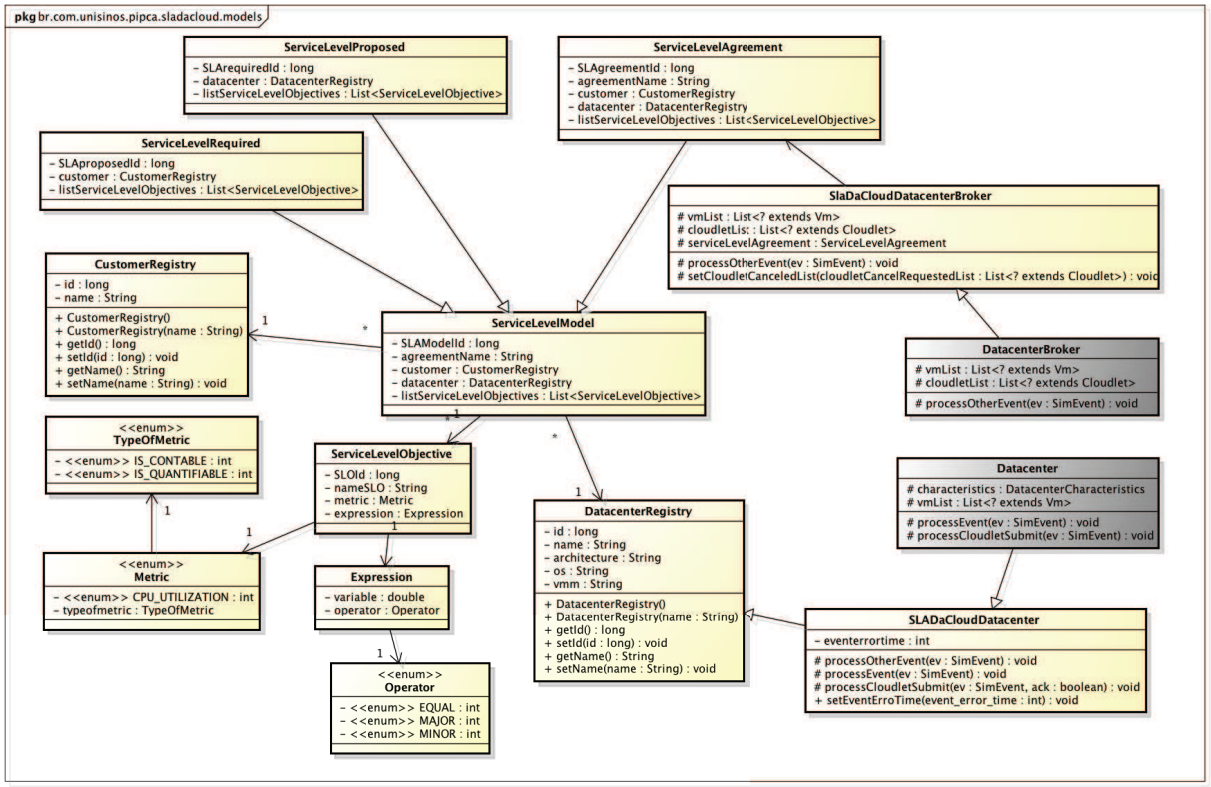
Fonte: Adaptado de (CALHEIROS et al., 2011).

CloudSim. Inicialmente, todos os *datacenters* criados são registrados no *Cloud Information Service* (CIS), entidade responsável pelo provimento de informações relativas ao ambiente simulado. Posteriormente, as requisições por recursos dos usuários, realizadas por meio de seus *brokers*, são respondidads com uma lista dos *datacenters* disponíveis. As características de cada um dos *datacenters* são então analisadas para se verificar a viabilidade de alocação das máquinas virtuais do usuário, que, após esta análise, são enviadas para criação. Em seguida, o usuário envia tarefas para execução, recebendo os resultados como resposta. A comunicação se encerra com a desalocação das máquinas virtuais.

O CloudSim não traz suporte a SLA. Para avaliar o SLAd@Cloud, foi necessário estender e alterar alguns dos componentes diretos do CloudSim. Um exemplo é o componente *Datacenter*. Ele é o responsável pela execução das *cloudlets*. Um novo componente, o *SlaDaCloudDatacenter* foi criado e implementou o método *setEventErrorTime*. Esse método gera no momento informado, no relógio da simulação, um evento de erro simulando uma falha no datacenter. Outro exemplo é o componente *DatacenterBroker*. Como demonstrado na Figura 19 ele é o responsável por mediar as negociações entre os usuários e os provedores de Nuvem e respectivamente a distribuição das *cloudlets* para os *datacenters*.

Um novo componente, o *SlaDaCloudDatacenterBroker* faz a comunicação entre sistema e o CloudSim. Nesse componente, o método *processOtherEvent* processa ações quando o evento de falha no datacenter ocorre. Dentre elas, o método *setCloudletCanceledList* cancela as *cloudlets* em execução no momento da falha. A Figura 20 demonstra o diagrama das classes alteradas no CloudSim. Nessa figura, as classes mostradas em amarelo foram criadas por meio deste trabalho, enquanto as partes mostradas em cinza representam componentes do CloudSim. Também é possível observar as classes de definição de um SLA no SLAd@Cloud e como elas se relacionam com as entidades do CloudSim. Um SLA é composto por vários SLOs. Cada SLO define um requisito de desempenho de uma característica única do serviço necessário para atingir um

Figura 20: Representação do SLA no SLAd@Cloud.

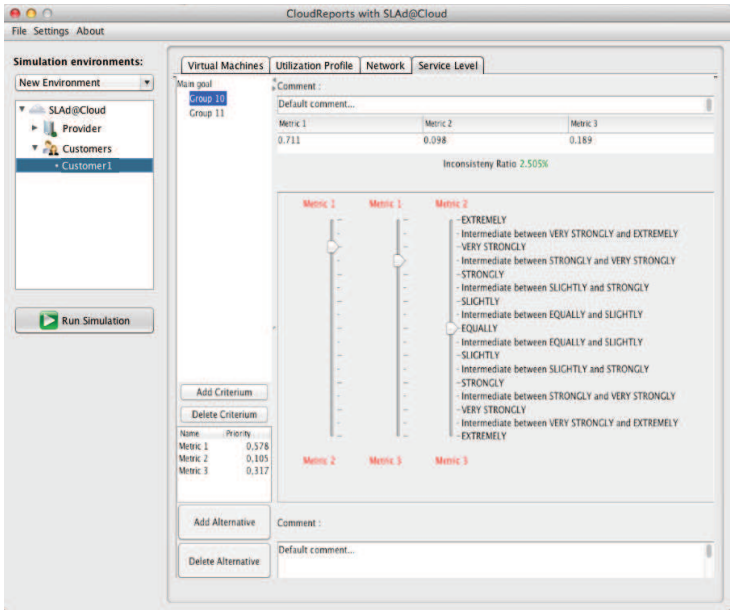


Fonte: Elaborado pelo autor.

determinado objetivo. Um SLO é determinado por uma propriedade que tem uma métrica e um operador de validação para que seja possível fazer sua avaliação. O *SlaDaCloudDatacenterBroker* submete as *cloudlets* solicitadas pelo usuário observando o SLA acordado, com as métricas já priorizadas pelo módulo de Gestão do SLA do SLAd@Cloud, para execução. Outro ajuste necessário foi a inserção de um fator multiplicador nos custos financeiros. No CloudSim os custos estão diretamente associados aos recursos disponibilizados. Entendemos que estes custos podem ser considerados como sendo custos primários para disponibilização e manutenção dos recursos pelos provedores e que dependendo do grau de exigência do SLA, um custo extra, configurado por um fator multiplicador, é adicionado aos serviços para cobrir os riscos e eventuais reforços na prestação de serviços para atendimento de SLAs mais exigentes.

O CloudSim não fornece interface gráfica. Para que fosse possível usufruir da interface gráfica disponibilizada pelo módulo de Gestão de Negócios do SLAd@Cloud com o CloudSim sem que fosse necessário desenvolver telas adicionais para a simulação, foi utilizado o CloudReports (Sá; SOARES; GOMES, 2011). O CloudReports oferece a interface gráfica para o CloudSim e a integração para geração de relatórios com os resultados da simulação. O CloudReports também é licenciado pela *General Public Licence* GPL. O CloudReports foi alterado, por este trabalho, para inclusão das telas de parametrização do SLA oferecidas pelo SLAd@Cloud. Um exemplo pode ser visto na Figura 21. Nessa figura, pode-se ver a tela em que o usuário associa os SLO requeridos aos grupos e realiza as comparações par a par, grupo com grupo e métrica

Figura 21: Tela de priorização de métricas do SLAd@Cloud.



Fonte: Elaborado pelo autor.

com métrica. Componentes *JSliders* (deslizantes) permitem a entrada dos dados das comparações. A tela exibe os pesos das métricas atualizadas a cada interação para que o usuário saiba quais os pesos relativos e absolutos das métricas.

A viabilidade de desenvolvimento da comparação par a par em java foi demonstrado no trabalho de Chou (CHOU, 2013). Como seu código não encontra-se disponível a comparação foi totalmente implementada neste trabalho.

Por fim, para as implementações do motor *fuzzy*, o SLAd@Cloud utilizou o *jFuzzyLogic* (CINGOLANI; ALCALA-FDEZ, 2012) uma biblioteca licenciada pela GPL para sistemas *fuzzy*.

5.2 Análise da negociação do SLA

A principal proposta do presente trabalho é um sistema que permite realizar a negociação para contratação de um SLA com múltiplas métricas entre o usuário e o provedor de uma nuvem computacional. Estas métricas são priorizadas pelo usuário de acordo com sua necessidade e o SLAd@Cloud, utilizando um motor de priorização, garante a contratação do SLA que melhor atende os pesos definidos pelo usuário. Esta seção tem como objetivo demonstrar a eficiência desta abordagem em relação ao método tradicional, o que não trabalha com múltiplas métricas. Na Subseção 3.1.3 observamos que os SLAs oferecidos pelos provedores de computação em nuvem comerciais são simples, pré-definidos e somente a disponibilidade é avaliada, ou seja, uma única métrica.

O resultado esperado desta avaliação é demonstrar um mecanismo que permite negociar o SLA avaliando mais de uma métrica e considerando a importância de cada métrica informada

pelo usuário, resultando na contratação de um SLA mais aderente aos requerimentos de QoS de um usuário de computação em nuvem. Para avaliar a negociação do SLA foram realizadas simulações compostas por dois tipos principais de agentes: um provedor de nuvem, que possui um *datacenter* e um ou mais usuários com diferentes requisitos de SLA. Os usuários utilizam os serviços oferecidos pelo provedor para o envio e alocação de máquinas virtuais que, por sua vez, executam um conjunto de tarefas, as *cloudlets*. Visando demonstrar a aplicação da negociação do SLA com múltiplas métricas para diferentes modelos de serviço de nuvem, essa avaliação considerou dois cenários. No primeiro, foram trabalhadas métricas do modelo IaaS, infraestrutura como serviço. No segundo foram consideradas métricas do modelo SaaS, software como serviço.

5.2.1 Aplicação da negociação no modelo IaaS - Infraestrutura como Serviço

Nessa subseção, a avaliação demontrará a abordagem com múltiplas métricas no modelo IaaS, em que foram utilizadas as métricas de disponibilidade, utilização de CPU, memória e seus respectivos custos financeiros. Para a abordagem que não utiliza múltiplas métricas foi utilizada, semelhante ao que algumas nuvens comerciais oferecem, a métrica disponibilidade. As Tabelas 10 e 11 demonstram as configurações realizadas na nuvem simulada.

A composição deste primeiro cenário simulado é exibida na Tabela 10 que mostra as características da máquina virtual e cloudlets. O usuário tem 1.000 tarefas para executar em 10 máquinas virtuais, cada uma com 1.000 MIPS (milhões de instruções por segundo). Cada tarefa tem 5.000 MIPS e o escalonador de *cloudlets* é o "compartilhado no espaço", o que significa que as *cloudlets* são alocadas em momentos distintos e executadas com dedicação exclusiva por cada um processadores ou VMs.

Tabela 10: Configuração da VM e *cloulets* na negociação do SLA no modelo IaaS.

Tipo	Parâmetro	Valor
VM	Quantidade de VMs	10
	Tamanho da imagem da VM (MB)	500
	Tamanho da memória RAM da VM (MB)	512
	Tamanho da CPU da VM (MIPS)	1.000
	Largura de banda da VM (kbps)	1.000
	Sistema Operacional da VM	Linux
	Escalonador de *cloudlets*	Compartilhado no espaço
cloudlet	Quantidade de *cloudlets*	1.000
	Tamanho da *cloudlet* (MIPS)	5.000
	Tamanho do arquivo de entrada da *cloudlet* (byte)	500
	Tamanho do arquivo de saída da *cloudlet* (byte)	500
	Quantidade de processadores para execução	1

Fonte: Elaborado pelo Autor.

A Tabela 11 demonstra as características do SLA requerido pelo usuário da nuvem. Na coluna "Grupo" pode-se visualizar os grupos definidos pelo usuário para agrupamento das métricas. Este agrupamento é o primeiro estágio da hierarquização das métricas. As colunas

"Métrica" e "Expressão" demonstram os requisitos de SLO definidos. Nesta primeira avaliação, escolheu-se métricas quantitativas e as expressões foram definidas em percentual(%), ou seja, o valor expresso é referente ao total de recursos solicitados em cada métrica que deve ser entregue pelo provedor da nuvem. Os valores exibidos na colunas "Pesos do Grupo", "Peso da Métrica" e "Pelo Global" são resultados da primeira etapa de cálculos realizados pelo motor de priorização, ou o resultado da matriz de comparação par a par normalizada (no modelo a matriz A=[a_{ij}]), que obteve como entrada na interface para escala de comparação de critérios (SAATY, 1980) os seguintes parâmetros: a) entre grupos: funcional "pouco mais importante que" não funcional (valor 3), funcional "muito mais importante que" custos financeiros (valor 7) e não funcional "pouco mais importante que" custos financeiros (valor 3). b) entre métricas: em cada grupo, todas receberam "mesma equivalência (valor 1) considerando que para as *cloudlets* em questão, a utilização e custos de CPU e memória tem igual relevância. É possível observar na coluna "Peso Global" que os custos financeiros neste SLA tem menor peso.

Tabela 11: Configuração do SLA requerido pelo usuário no modelo IaaS.

Grupo	Peso do Grupo	Métrica	Expressão	Peso da Métrica	Peso Global
Funcional	60,00%	A - Quantidade de CPU	>= 98%	50,00%	30,00%
		B - Quantidade de memória	>=98%	50,00%	30,00%
			Total	100,00%	
Não funcional	29,00%	C - Disponibilidade	> 99%	100,00%	29,00%
			Total	100,00%	
Custos financeiros	11,00%	D - Custo da unidade de CPU (1 MI)(R$/s)	<= 0,01	50,00%	5,50%
		E - Custo da unidade de memória (1 MB)(R$/s)	<= 0,01	50,00%	5,50%
			Total	100,00%	
Total	100,00%				100,00%

Fonte: Elaborado pelo Autor.

A infraestrutura do provedor simulado foi composta por um *datacenter*. As Tabelas 12 e 13 demonstram as configurações realizadas no provedor da nuvem. A Tabela 12 mostra detalhadamente os recursos disponíveis neste *datacenter*, os valores de custo de utilização de processamento, memória, armazenamento e banda bem como a arquitetura de processamento, o sistema operacional e o hipervisor utilizado são atributos únicos de um *datacenter*, sendo aplicado a todas as entidades a ele pertencente, como os seus nós ou *hosts*. Por outro lado, o poder de processamento, a quantidade de memória de sistema e a capacidade de armazenamento são características de *hosts*, que podem ser customizados individualmente, criando-se *datacenters* com recursos heterogêneos. O *datacenter* aqui simulado é composto por nós homogêneos, ou seja, todos os *hotst* possuem a mesma configuração. O *datacenter* tem 3 servidores com 5 CPUs cada. Cada CPU com 1.000 MIPS (milhões de instruções por segundo). O escalonador de *VMs* é o "compartilhado no tempo", o que significa que as *VMs* concorrem com um mesmo núcleo de processamento.

A Tabela 13 demonstra as características dos SLAs ofertados pelo provedor para os usuários da nuvem. Os três SLAs diferem nas garantias oferecidas e consequentemente, nos custos. Quanto maior as garantias oferecidas, maior o custo adicional para fornecimento destas ga-

Tabela 12: Configuração do *datacenter* e *hosts* no provedor da negociação do SLA no modelo IaaS.

Tipo	Parâmetro	Valor
datacenter	Quantidade de *hosts*	3
	Arquitetura	x86
	Sistema Operacional	Linux
	Hipervisor	Xen
	Escalonador de *VMs*	Compartilhado no tempo
	Custo da unidade de CPU (1.000 MIPS) (R$)	0,01
	Custo da unidade de memória (1 MB) (R$)	0,01
	Custo da unidade de armazenamento (1GB) (R$)	0,01
	Custo da unidade de banda (1 kbps)(R$/s)	0,01
host	Quantidade de CPUs	5
	Tamanho da memória RAM do *host* (MB)	16.384
	Tamanho de cada CPU do *host* (MIPS)	1.000
	Largura da banda do *host* (kbps)	10.000
	Capacidade de armazenamento do *host* (MB)	1.000.000

Fonte: Elaborado pelo Autor.

rantias. O custo adicional e a penalidade são calculados sobre o valor do custo primário dos recursos do *datacenter*. Quando uma violação de uma métrica ocorre o valor referente a penalidade é devolvida ao usuário e o valor do custo adicional da garantia não é cobrada, uma vez que ela não foi realizada. A informação do histórico é proveniente dos módulos de monitoramento e representa a média dos resultados de desempenho que cada métrica em execuções anteriores do SLA.

Tabela 13: Configuração dos SLAs ofertados pelo provedor do SLA no modelo IaaS.

SLA	Métricas ou Termos	Garantia	Custo adicional garantia	Penalidade
1	A - Quantidade de CPU	> 95,00%	1%	2%
	B - Quantidade de memória			
	C - Disponibilidade	> 99,00%		
2	A - Quantidade de CPU	= 100%	2,5%	5%
	B - Quantidade de memória			
	C - Disponibilidade	> 99,50%		
3	A - Quantidade de CPU	= 100%	5%	10%
	B - Quantidade de memória			
	C - Disponibilidade	> 99,95%		

Fonte: Elaborado pelo Autor.

De posse dos SLAs ofertados, o motor de priorização realiza a segunda etapa de cálculos, que é comparar e ordenar os SLAs ofertados. Essa comparação é realizada classificando os SLAs ofertados de acordo com o nível de atendimento de cada métrica. A Figura 22 exibe o racional desta comparação. Nessa figura, pode-se observar que cada métrica é comparada entre os SLAs ofertados, obtendo-se notas da escala de comparação de critérios (SAATY, 1980). Por exemplo, pode-se observar que a oferta de disponibilidade (métrica C) é melhor no SLA3 que no SLA2 e também melhor no SLA2 que no SLA1.

A Figura 23 exibe os próximos passos. Na parte a) é possível ver que, após a comparação, o motor de priorização do SLAd@Cloud tem os pesos relativos de cada métrica em cada SLA. Na parte b) é possível verificar o resultado do cálculo final dos pesos considerando a multiplicação

Figura 22: Matrizes de comparação pareadas dos indicadores entre os SLAs ofertados.

Métrica A	SLA1	SLA2	SLA3
SLA1	1,00	0,33	0,33
SLA2	3,00	1,00	1,00
SLA3	3,00	1,00	1,00

Métrica B	SLA1	SLA2	SLA3
SLA1	1,00	0,33	0,33
SLA2	3,00	1,00	1,00
SLA3	3,00	1,00	1,00

Métrica C	SLA1	SLA2	SLA3
SLA1	1,00	0,50	0,25
SLA2	2,00	1,00	0,25
SLA3	4,00	4,00	1,00

Métrica D	SLA1	SLA2	SLA3
SLA1	1,00	0,50	0,25
SLA2	2,00	1,00	0,25
SLA3	4,00	4,00	1,00

Métrica E	SLA1	SLA2	SLA3
SLA1	1,00	0,50	0,25
SLA2	2,00	1,00	0,25
SLA3	4,00	4,00	1,00

Fonte: Elaborado pelo autor.

de cada peso relativo de cada métrica (parte a) com os pesos globais de cada métrica (coluna "peso global" na Tabela 11). Já é possível saber qual SLA ofertado tem maior peso, observando os requisitos do SLA requerido. Este é o SLA que o SLAd@Cloud irá sugerir para o usuário na primeira negociação. O resultado é armazenado na matriz B=[b_{ij}] do sistema.

Figura 23: Resultados globais de comparação dos SLAs ofertados.

(a)	SLA1	SLA2	SLA3
A	0,1422	0,4289	0,4289
B	0,1422	0,4289	0,4289
C	0,1033	0,2911	0,6056
D	0,6056	0,2911	0,1033
E	0,6056	0,2911	0,1033

(b)	SLA1	SLA2	SLA3
A	0,0427	0,1287	0,1287
B	0,0427	0,1287	0,1287
C	0,0300	0,0844	0,1756
D	0,0333	0,0160	0,0057
E	0,0333	0,0160	0,0057
Peso	0,1819	0,3738	0,4443

Fonte: Elaborado pelo autor.

Neste cenário foram realizadas 50 execuções da simulação e os resultados exibidos em colunas como demonstrado na Tabela 14. Nessa tabela, a coluna "a" exibe o identificador da execução. As colunas de "b", "c" e "d" demostram o resultado da comparação dos pesos calculados para cada SLA na simulação. A coluna "e" exibe qual foi o SLA selecionado nesta negociação. As colunas de "f", "g" e "h" mostram a quantidade de recursos entregues (em percentual referente ao que foi solicitado). Esta quantidade de recursos entregues compõem o histórico de execução para decisões futuras. As colunas "i" e "j" apresentam os custos indivivuais dos recursos em determinada execução. A coluna "k" mostra o custo do SLA, ou seja, o custo cobrado pelo fornecedor para dar a garantia adicional. A coluna "l" exibe o custo da violação, se esta aconteceu, enquanto a coluna "m" exibe a métrica que foi violada. E as colunas "n" e "o" mostram respectivamente o custo e o tempo total da execução.

Como esperado, na primeira execução, o SLAd@Cloud sugere o SLA 3 na negociação. O SLA 3 tem as melhores ofertas das métricas CPU, memória e disponibilidade para o usuário, uma vez que, essas métricas representam os maiores pesos no SLA requerido, respectivamente 30,00%, 30,00% e 29,00% da distribuição dos pesos. Apesar do SLA 3 ter o maior custo financeiro, as métricas de custo representam juntas somente 11,00% da distribuição dos pesos.

Ainda observando os resultados de cada execução na Tabela 14 é possível destacar alguns

Tabela 14: Parâmetros observados nas avaliações de negociação para IaaS.

Exec. (a)	Pesos calculados			SLA selecionado (e)	Recursos entregues (%)			Custos			Penalidade		Custo total (n)	Tempo (s) (o)
	SLA 1 (b)	SLA 2 (c)	SLA 3 (d)		CPU (f)	MEM (g)	DISP (h)	CPU (i)	MEM (j)	SLA (k)	Custo (l)	Métrica (m)		
1	0,1819	0,3738	0,4443	3	100,00	100,00	100,00	50,00	51,20	5,06	0,00		106,26	500,00
2	0,2726	0,3512	0,3762	3	100,00	100,00	100,00	50,00	51,20	5,06	0,00		106,26	500,00
3	0,2726	0,3512	0,3762	3	100,00	100,00	100,00	50,00	51,20	5,06	0,00		106,26	500,00
4	0,2726	0,3512	0,3762	3	100,00	100,00	100,00	50,00	51,20	5,06	0,00		106,26	500,00
5	0,2726	0,3512	0,3762	3	100,00	100,00	100,00	50,00	51,20	5,06	0,00		106,26	500,00
6	0,2726	0,3512	0,3762	3	100,00	100,00	100,00	50,00	51,20	5,06	0,00		106,26	500,00
7	0,2726	0,3512	0,3762	3	100,00	100,00	100,00	50,00	51,20	5,06	0,00		106,26	500,00
8	0,2726	0,3512	0,3762	3	100,00	100,00	100,00	50,00	51,20	5,06	0,00		106,26	500,00
9	0,2726	0,3512	0,3762	3	100,00	100,00	100,00	50,00	51,20	5,06	0,00		106,26	500,00
10	0,2726	0,3512	0,3762	3	100,00	100,00	100,00	50,00	51,20	5,06	0,00		106,26	500,00
11	0,2726	0,3512	0,3762	3	100,00	100,00	100,00	50,00	51,20	5,06	0,00		106,26	500,00
12	0,2726	0,3512	0,3762	3	100,00	100,00	100,00	50,00	51,20	5,06	0,00		106,26	500,00
13	0,2726	0,3512	0,3762	3	100,00	100,00	100,00	50,00	51,20	5,06	0,00		106,26	500,00
14	0,2726	0,3512	0,3762	3	100,00	100,00	100,00	50,00	51,20	5,06	0,00		106,26	500,00
15	0,2726	0,3512	0,3762	3	100,00	100,00	100,00	50,00	51,20	5,06	0,00		106,26	500,00
16	0,2726	0,3512	0,3762	3	100,00	100,00	100,00	50,00	51,20	5,06	0,00		106,26	500,00
17	0,2726	0,3512	0,3762	3	95,00	95,00	100,00	50,00	51,20	0,00	10,12	a, b	91,08	526,32
18	0,2895	0,3773	0,3328	2	100,00	100,00	100,00	50,00	51,20	2,53	0,00		103,73	500,00
19	0,2895	0,3773	0,3328	2	100,00	100,00	100,00	50,00	51,20	2,53	0,00		103,73	500,00
20	0,2895	0,3773	0,3328	2	100,00	100,00	100,00	50,00	51,20	2,53	0,00		103,73	500,00
21	0,2895	0,3773	0,3328	2	100,00	100,00	100,00	50,00	51,20	2,53	0,00		103,73	500,00
22	0,2895	0,3773	0,3328	2	100,00	100,00	100,00	50,00	51,20	2,53	0,00		103,73	500,00
23	0,2895	0,3773	0,3328	2	100,00	100,00	100,00	50,00	51,20	2,53	0,00		103,73	500,00
24	0,2895	0,3773	0,3328	2	100,00	100,00	100,00	50,00	51,20	2,53	0,00		103,73	500,00
25	0,2895	0,3773	0,3328	2	100,00	100,00	100,00	50,00	51,20	2,53	0,00		103,73	500,00
26	0,2895	0,3773	0,3328	2	100,00	100,00	100,00	50,00	51,20	2,53	0,00		103,73	500,00
27	0,2895	0,3773	0,3328	2	100,00	100,00	99,00	50,00	51,20	0,00	5,06	c	96,14	505,00
28	0,2817	0,3905	0,3325	2	100,00	100,00	99,00	50,00	51,20	0,00	5,06	c	96,14	505,00
29	0,2817	0,3905	0,3325	2	100,00	100,00	99,00	50,00	51,20	0,00	5,06	c	96,14	505,00
30	0,2817	0,3905	0,3325	2	100,00	100,00	99,00	50,00	51,20	0,00	5,06	c	96,14	505,00
31	0,2817	0,3905	0,3325	2	100,00	100,00	99,00	50,00	51,20	0,00	5,06	c	96,14	505,00
32	0,2817	0,3905	0,3325	2	100,00	100,00	99,00	50,00	51,20	0,00	5,06	c	96,14	505,00
33	0,2817	0,3905	0,3325	2	100,00	100,00	99,00	50,00	51,20	0,00	5,06	c	96,14	505,00
34	0,2817	0,3905	0,3325	2	100,00	100,00	99,00	50,00	51,20	0,00	5,06	c	96,14	505,00
35	0,2817	0,3905	0,3325	2	100,00	100,00	99,00	50,00	51,20	0,00	5,06	c	96,14	505,00
36	0,3027	0,3483	0,3536	3	95,00	95,00	100,00	50,00	51,20	0,00	10,12	a, b	91,08	526,32
37	0,3027	0,3483	0,3536	3	95,00	95,00	100,00	50,00	51,20	0,00	10,12	a, b	91,08	526,32
38	0,3027	0,3483	0,3536	3	95,00	95,00	100,00	50,00	51,20	0,00	10,12	a, b	91,08	526,32
39	0,3027	0,3483	0,3536	3	95,00	95,00	100,00	50,00	51,20	0,00	10,12	a, b	91,08	526,32
40	0,3027	0,3483	0,3536	3	95,00	95,00	100,00	50,00	51,20	0,00	10,12	a, b	91,08	526,32
41	0,3027	0,3483	0,3536	3	95,00	95,00	100,00	50,00	51,20	0,00	10,12	a, b	91,08	526,32
42	0,3027	0,3483	0,3536	3	95,00	95,00	100,00	50,00	51,20	0,00	10,12	a, b	91,08	526,32
43	0,3027	0,3483	0,3536	3	95,00	95,00	100,00	50,00	51,20	0,00	10,12	a, b	91,08	526,32
44	0,3027	0,3483	0,3536	3	95,00	95,00	100,00	50,00	51,20	0,00	10,12	a, b	91,08	526,32
45	0,4151	0,2420	0,3473	1	100,00	100,00	100,00	50,00	51,20	0,00	2,02		102,212	500,00
46	0,4151	0,2420	0,3473	1	100,00	100,00	100,00	50,00	51,20	0,00	2,02		102,212	500,00
47	0,4151	0,2420	0,3473	1	100,00	100,00	100,00	50,00	51,20	0,00	2,02		102,212	500,00
48	0,4151	0,2420	0,3473	1	100,00	100,00	100,00	50,00	51,20	0,00	2,02		102,212	500,00
49	0,4151	0,2420	0,3473	1	100,00	100,00	100,00	50,00	51,20	0,00	2,02		102,212	500,00
50	0,4151	0,2420	0,3473	1	100,00	100,00	100,00	50,00	51,20	0,00	2,02		102,212	500,00

Fonte: Elaborado pelo Autor.

pontos interessantes. Os pesos calculados para cada SLA sofreram alterações a partir da primeira execução. Isto ocorreu devido a realização de execuções dos SLAs 1 e 2 sem que ocorressem violações. O SLAd@Cloud incorporou ao cálculo dos pesos a média histórica da execução anterior (recursos entregues) de cada SLA. Como a execução dos SLAs 1 e 2 apresentaram melhores resultados que suas garantias os pesos foram distribuídos de uma forma mais equilibrada

entre os SLAs.

Outras alterações nos pesos também podem ser percebidas nas execuções 2, 18, 28, 36 e 45, justamente porque nas execuções anteriores a estas ocorreram fatos que influenciaram no histórico de execução. Ou seja, como veremos a seguir, o histórico da execução dos SLAs no SLAd@Cloud influência as futuras decisões de negociação.

Durante as simulações foram inseridos eventos de falhas nas entregas dos recursos sem que nenhuma ação dinâmica, como tentativa de recuperação ou renegociação do SLA, fosse tomada. As ações de dinamicidade serão avaliadas nas próximas subseções deste trabalho. No evento 17 ocorreu uma violação de entrega das métricas de CPU e memória. Isto resultou em variação dos custos e tempo de execução, além da atualização dos pesos dos SLAs. O SLAd@Cloud recalculou os pesos de cada SLA e passou a sugerir o SLA 2 a partir das próximas negociações (execução 18). Analisando os dados, é possível notar, que esta indicação aconteceu preventivamente, até mesmo antes da média do histórico da entrega de CPU e memória tornar o SLA 3 não passível de contratação. Isto ocorreu porque a violação deu-se nas métricas com os maiores pesos. O histórico influenciou na decisão na razão de 50%. Este valor pode ser configurado pelo usuário durante a negociação do SLA.

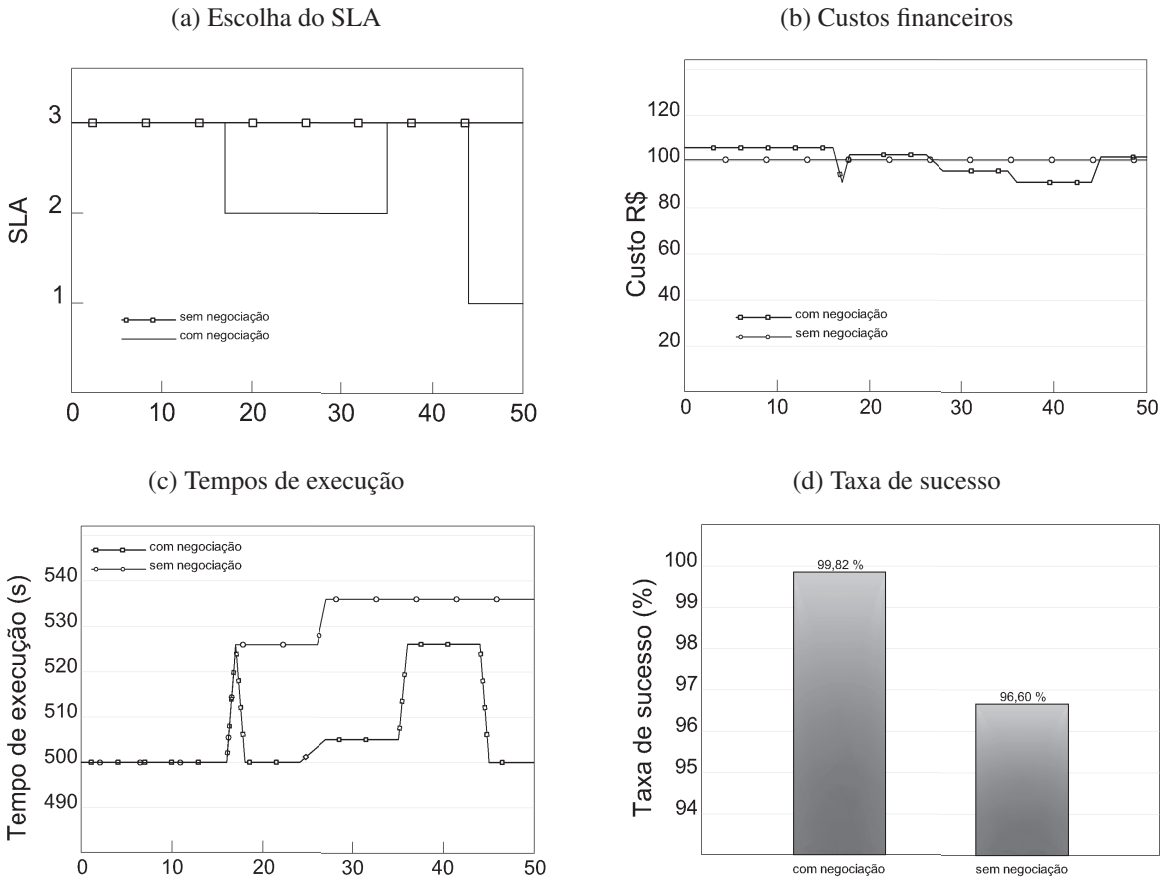
A partir do evento 27 foram geradas violações de disponibilidade na entrega do SLA 2. Estas violações também geraram variações nos custos e tempo de execução. Semelhante a primeira violação, os pesos sofreram ajustes em decorrência do histórico da entrega, porém a influência ocorreu numa métrica de menor relevância o que não foi suficiente para que o SLAd@Cloud sugerisse outro SLA imediatamente. A troca do SLA só ocorreu na execução 36 quando historicamente a média de entrega de disponibilidade do SLA 2 o tornou não passível de contratação (média menor que 99% que é o requisito de contratação). Então o SLAd@Cloud voltou a sugerir o SLA 3 nas negociações. Como não houve ações de dinamicidade para reestabelecer sua capacidade de entrega, as execuções ainda ocorrem com violações das métricas de CPU e memória. Mesmo assim, o SLA 3 ainda tem um peso maior que os demais. Isto se explica porque a média histórica influência em 50%, enquanto que a garantia dada no SLA influência em outros 50%. A garantia dada no SLA 3 juntamente com seu histórico para as métricas de CPU e memória ainda são melhores que no SLA 1.

A partir do evento 44 a média histórica de entrega de CPU e memória do SLA 3 torna-o não passível de contratação (média menor que 95% que é o requisito de contratação). Então o SLAd@Cloud passa a sugerir o SLA 1 nas próximas negociações. Inicialmente imaginou-se haver um erro neste ponto porque o SLA 1 não atende, nas garantias, os requisitos de CPU e memória, métricas que tem maior peso na negociação. O SLAd@Cloud poderia não ofertar nenhum SLA, porém, considerando os resultados dos calculos dos pesos, o histórico de execução do SLA 1 demonstrou que a entrega das métricas CPU e memória ocorreu em valor superior ao solicitado nos requisitos, o que justifica a oferta. Outra dúvida que poderia surgir, seria, se a média histórica influência 50% e as garantias outros 50%, isto poderia levar a crer que o SLAd@Cloud poderia decidir igualmente pelos SLAs 1 ou SLA 3, porém neste caso, os

custos financeiros, menores no SLA 1 são critérios de desempate.

É possível concluir com estes resultados que o SLAd@Cloud ofertou sempre o melhor SLA, de acordo com os requisitos priorizados pelo usuário, considerando a melhor garantia e considerando o histórico de cumprimento destas garantias. Isto ocorre porque o SLAd@Cloud retroalimenta as negociações com os resultados do monitoramento das execuções anteriores. Na Figura 24 pode-se observar a comparação da negociação, custos financeiros, tempo de execução e taxa de sucesso do SLAd@Cloud e da abordagem tradicional que não faz a negociação do SLA. O gráfico na parte 24a da Figura mostra que no SLAd@Cloud o SLA é negociado e um melhor SLA é oferecido para o usuário conforme as variações de entrega ocorrem, enquanto que, na abordagem tradicional um mesmo SLA contratado na primeira execução permanece até o final, ou seja, não há negociação do SLA.

Figura 24: Resultados da avaliação de negociação do SLA em IaaS



Fonte: Elaborado pelo Autor.

O gráfico na parte 24b da Figura mostra os custos financeiros das duas abordagens. Mesmo com o custo adicional da garantia cobrada pelo provedor para manter o SLA, na simulação configurada em 5% (SLA 2) e 10% (SLA 3), a execução resultou numa redução de 0,73% do valor quando o usuário utilizou o SLAd@Cloud. No SLAd@Cloud o custo médio foi de R$ 100,46 e no cenário em que não há negociações do SLA o custo médio foi de R$ 101,20. Isto

aconteceu porque, com a negociação, os novos SLAs oferecerem melhor oferta de recursos. Se não fosse realizada a cobrança de valores adicionais pelas garantias, a mesma simulação, resulta numa redução de 2,78% do valor para o usuário. No SLAd@Cloud o custo médio seria de R$ 98,38. Pode-se notar que a variação do custo na adoção de um modelo de SLA com garantias, mesmo com cobrança adicional por parte do provedor, afeta positivamente o usuário. Se ocorrerem violações o SLAd@Cloud opta por SLAs com históricos de execução favoráveis o que reduz os custos financeiros.

O gráfico na parte 24c da Figura mostra os tempos de execução. As simulações com o SLAd@Cloud terminaram em média com 506,16 segundos, enquanto que as simulações com a abordagem sem negociação terminaram em média com 522,95 segundos. Uma redução de 3% do tempo de execução. Semelhante aos resultados financeiros, o ganho obtem-se em função da oferta na negociação de SLAs com históricos de execução melhores.

O gráfico na parte 24d da Figura mostra as taxas de sucesso de execução das *cloudlets*. Na abordagem tradicional 96,60% das *cloudlets* enviadas foram executadas, enquanto que com o SLAd@Cloud99,82% das *cloudlets* enviadas foram executadas.

5.2.2 Aplicação da negociação no modelo SaaS - Software como Serviço

Nessa subseção, a avaliação demonstrará a abordagem com múltiplas métricas no modelo SaaS. Para avaliar a negociação do SLA na abordagem com múltiplas métricas no modelo SaaS foram utilizadas as métricas usabilidade, confiabilidade, desempenho, disponibilidade e custo financeiro. A usabilidade representa o grau de facilidade com que se pode utilizar um software para a realização de uma tarefa e a confiabilidade representa a probabilidade de um sistema funcionar sem a ocorrência de falhas num período e ambiente especificados. Estas métricas foram consideradas essenciais para provedores SaaS (CANCIAN; RABELO; WANGENHEIM, 2009). No CloudSim as aplicações SaaS também são simuladas pelas classes *cloudlets*. As *cloudlets* são as tarefas enviadas pelos usuários e podem ser caracterizadas por aplicações Web, aplicações HPC (*High-Performance Computing* ou Computação de Alto Desempenho), redes sociais (RAWAT; SAROHA; BARTHWAL, 2012).

A composição deste segundo cenário simulado é constituída por um usuário. A Tabela 15 mostra as características das cloudlets. O usuário tem 1.000.000 tarefas para executar. Cada tarefa tem 50 MIPS e o escalonador de *cloudlets* é o "compartilhado no tempo", o que significa que as *cloudlets* são alocadas no mesmo momento e executadas simultaneamente por cada processador ou VM. A ideia é aproximar-se do comportamento de uma aplicação que faz requisições pequenas mas em grande quantidade.

A Tabela 16 demonstra as características do SLA requerido pelo usuário da nuvem. O motor de priorizações calculou os pesos conforme os seguintes parâmetros informados pelo usuário: a) entre grupos: funcional "muito mais importante que" não funcional (valor 5), funcional "muito mais importante que" custos financeiros (valor 5) e não funcional "pouco mais importante que"

Tabela 15: Configuração das *cloulets* no usuário da negociação do SLA no modelo SaaS.

Tipo	Parâmetro	Valor
cloudlet	Quantidade de *cloudlets*	1.000.000
	Tamanho da *cloudlet* (MIPS)	50
	Tamanho do arquivo de entrada da *cloudlet* (byte)	50
	Tamanho do arquivo de saída da *cloudlet* (byte)	50

Fonte: Elaborado pelo Autor.

custos financeiros (valor 3). b) entre métricas: a confiabilidade "bastante mais importante que" a usabilidade (valor 7) e a disponibilidade "bastante mais importante que" o desempenho (valor 7). Estes valores mantiveram escalas semelhantes a priorização das métricas SaaS observadas em (CANCIAN; RABELO; WANGENHEIM, 2009).

Tabela 16: Configuração do SLA requerido pelo usuário no modelo SaaS.

Grupo	Peso do Grupo	Métrica	Expressão	Peso da Métrica	Peso Global
Funcional	65,90%	A - Usabilidade	= Alta	12,50%	8,23%
		B - Confiabilidade	= Alta	87,50%	57,66%
			Total	100,00%	
Não funcional	25,00%	C - Desempenho	< 0,05s	12,50%	3,13%
		D - Disponibilidade	> 99%	87,50%	21,88%
			Total	100,00%	
Custos financeiros	9,10%	D - Custo financeiro	<= 0,01	100,00%	9,10%
			Total	100,00%	
Total	100,00%				100,00%

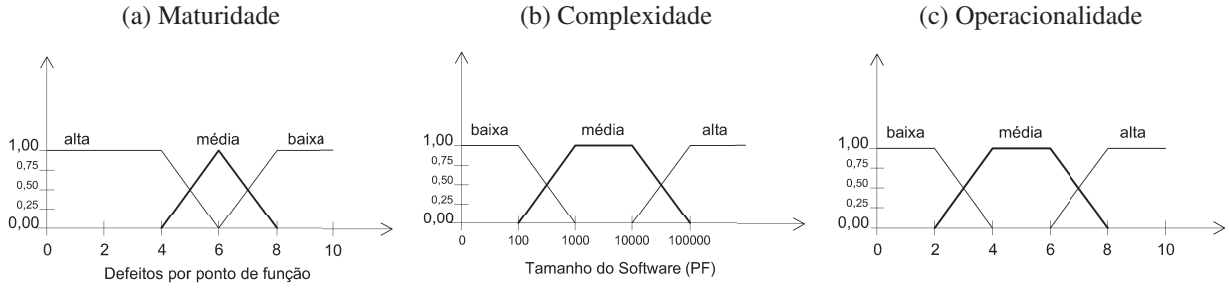
Fonte: Elaborado pelo Autor.

As métricas do grupo "Funcional" (na Tabela 16) são qualitativas e tiveram a expressão ou requisito definido como "alta". O SLAd@Cloud consegue avaliar e monitorar esta expressão utilizando a lógica *fuzzy* (ZADEH, 1965). A lógica *fuzzy* foi explanada na Subseção 4.1.1 do presente trabalho. As etapas realizadas para configuração do motor *fuzzy* são descritas a seguir:

Fuzzificação: A norma ISO/IEC 9126 de 1991 ou NBR 13596 de 1996 representa a padronização para itens de qualidade de software. Para confiabilidade, são consideradas: maturidade (frequencia com que o software apresenta defeitos), tolerância a falhas (como o software reage quando ocorrem falhas e recuperabilidade (se o software é capaz de recuperar dados em caso de falhas). Para usabilidade, são consideradas: compreensibilidade (grau de facilidade para entender o conjunto lógico e aplicabilidade), apreensibilidade (grau de facilidade para aprender a utilizar) e operacionalidade (grau de facilidade para operar e controlar o software).

O SLAd@Cloud permite a configuração de mais de um conjunto *fuzzy* por métrica, porém, para simplificar, nessa avaliação a confiabilidade considerou somente a maturidade e complexidade do software e a usabilidade considerou somente a operacionalidade. As funções de pertinência para estes conjuntos podem ser visualizados na Figura 25. Considerou-se o *Survey* (JONES, 2012) como referência para os valores de maturidade e complexidade de software. Na parte da Figura 26a, a densidade de defeitos é representada pela quantidade de defeitos dividida pela quantidade de pontos de função (MARTHALER, 2004). Na parte da Figura 25b, a com-

Figura 25: Funções de pertinência dos conjuntos *fuzzy* para métricas SaaS



plexidade do software é representado pela quantidade de PF (ponto de função). A aplicação simulada tem sua complexidade representada como alta.

Inferência: A Tabela 17 exibe as regras de inferências configuradas para a avaliação.

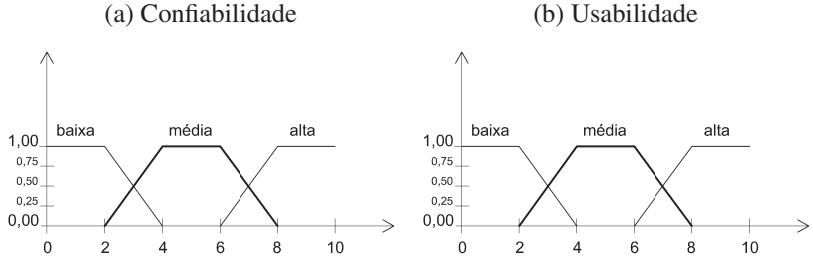
Tabela 17: Inferências para conjuntos de entrada *fuzzy* das Métrica SaaS.

Maturidade	Complexidade	Confiabilidade
Baixa	Baixa	Média
Baixa	Média	Baixa
Baixa	Alta	Baixa
Média	Baixa	Alta
Média	Média	Média
Média	Alta	Baixa
Alta	Baixa	Alta
Alta	Média	Alta
Alta	Alta	Alta

Operacionalidade	Complexidade	Usabilidade
Baixa	Baixa	Média
Baixa	Média	Baixa
Baixa	Alta	Baixa
Média	Baixa	Alta
Média	Média	Média
Média	Alta	Baixa
Alta	Baixa	Alta
Alta	Média	Alta
Alta	Alta	Alta

Desfuzifficação: A Figura 26 demonstra as funções de pertinência configuradas para os conjuntos de saída na avaliação. A função de *desfuzzificação* utilizada foi a *Centroíde*, que é a mais utilizada (NETO; COELHO, 2006).

Figura 26: Funções de pertinência dos conjuntos de saída *fuzzy* para métricas SaaS



Na composição deste cenário considerou-se dois provedores, cada um com seu *datacenter*. Nessa configuração o SLAd@Cloud está instalado em um *broker*. Um *broker* é terceiro agente que faz a integração e distribuição dos serviços entre usuários e provedores de nuvens. Os SLAs ofertados serão distribuídos entre os provedores para demonstrar a capacidade do SLAd@Cloud de trabalhar com múltiplos provedores.

As Tabelas 18 e 19 demonstram as configurações realizadas nos provedores da nuvem. A Tabela 18 mostra os recursos disponíveis nos *datacenters*. Os dois *datacenters* tem configurações semelhantes, a mesma quantidade de *hosts* e o mesmo tamanho de CPU por *hosts* e VMs. Neste cenário, as VMs ficam configuradas nos provedores, abstraindo esta camada dos usuários.

Tabela 18: Configuração do *datacenter* e *hosts* nos provedores da negociação do SLA no modelo SaaS.

Tipo	Parâmetro	Valor
provedor	Quantidade de *hosts*	5
	Arquitetura	x86
	Sistema Operacional	Linux
	Hipervisor	Xen
	Escalonador de *VMs*	Compartilhado no tempo
	Custo da unidade de CPU (1.000 MIPS) (R$)	0,01
	Custo da unidade de memória (1 MB) (R$)	0,01
	Custo da unidade de armazenamento (1GB) (R$)	0,01
	Custo da unidade de banda (1 kbps)(R$/s)	0,01
VM	Quantidade de VMs	50
	Tamanho da imagem da VM (MB)	500
	Tamanho da memória RAM da VM (MB)	512
	Tamanho da CPU da VM (MIPS)	1.000
	Largura de banda da VM (kbps)	1.000
	Sistema Operacional da VM	Linux
	Escalonador de *cloudlets*	Compartilhado no tempo

Fonte: Elaborado pelo Autor.

A Tabela 19 demonstra as características dos SLAs ofertados pelo provedor para os usuários da nuvem. Os SLAs oferecem diferentes garantias para usabilidade, confiabilidade, desempenho e disponibilidade. O SLA 2 é ofertado por um provedor diferente dos demais. Os custos das garantias e penalidades também diferem de um SLA para outro.

Tabela 19: Configuração dos SLAs ofertados pelo provedor do SLA no modelo SaaS.

SLA	Provedor	Métricas ou Termos	Garantia	Custo adicional garantia	Penalidade
1	Provedor 1	A - Usabilidade	Média	1%	2%
		B - Confiabilidade	Média		
		C - Desempenho	< 0,06s por transação		
		D - Disponibilidade	> 99,50%		
2	Provedor 2	A - Usabilidade	Baixa	2,5%	5%
		B - Confiabilidade	Alta		
		C - Desempenho	< 0,05s por transação		
		D - Disponibilidade	> 99,00%		
3	Provedor 1	A - Usabilidade	Média	5%	10%
		B - Confiabilidade	Alta		
		C - Desempenho	< 0,04s por transação		
		D - Disponibilidade	> 99,50%		

Fonte: Elaborado pelo Autor.

Na primeira tentativa de execução, o SLAd@Cloud informou ao usuário que nenhum SLA ofertado atende totalmente aos requisitos e nenhum SLA é negociado. Isto ocorreu porque o requisito informado inicialmente para a métrica usabilidade foi “alto” e não há SLA com garantias ou histórico de execução que atenda este requisito. Com o auxílio da interface de

contratação do SLAd@Cloud o requisito foi ajustado para “médio” e esta primeira tentativa foi descartada.

A seguir foram realizadas 50 execuções da simulação e os resultados consolidados em colunas como demonstrado na Tabela 20. Nessa tabela, a coluna “a” exibe as execuções agrupadas. As colunas de “b”, “c” e “d” demostram o resultado da comparação dos pesos calculados para cada SLA na simulação. A coluna “e” exibe qual foi o SLA sugerido nas negociações. As colunas de “f”, “g” ,“h” e “i” mostram a quantidade de recursos entregues. Para coletar o desempenho da entrega dos recursos para as métricas qualitativas, o SLAd@Cloud apresenta para o usuário um questionário ao final da execução dos serviços. Nesse questionário o usuário insere um valor numérico na escala de 1 a 10 para informar como ele foi atendido em cada uma das métricas qualitativas. O SLAd@Cloud usa novamente as funções de pertinência da Figura 26 para converter e registrar os valores no histórico de execução. As colunas “j’ e “k” apresentam o custo e tempo médios de execução em determinado intervalo.

Na primeira execução, desta vez bem sucedida, o SLAd@Cloud sugere o SLA 3 na negociação. O SLA 3 tem as melhores ofertas das métricas confiabilidade e disponibilidade, métricas que representam os maiores pesos no SLA requerido. Diferente da avaliação anterior os pesos calculados para cada SLA não sofreram alterações na primeiras execuções(1-6). Isto ocorreu porque os históricos de execução dos SLAs foram bem próximos dos requisitos ofertados, o que manteve os pesos semelhantes.

Tabela 20: Parâmetros observados nas avaliações de negociação para SaaS.

Exec. (a)	Pesos calculados			SLA contratato (e)	Recursos entregues				Custo total médio (j)	Tempo (min) (k)
	SLA 1 (b)	SLA 2 (c)	SLA 3 (d)		Conf. (f)	Usab. (g)	Disp. %(h)	Desem. (s)(i)		
1-6	0,2238	0,3257	0,4505	3	Alta	Média	100,00	0,04	10.250,00	13,88
7-14	0,1983	0,4590	0,3423	2	Alta	Baixa	100,00	0,05	10.250,00	16,66
15-23	0,2087	0,4381	0,3528	2	Alta	Baixa	99,25	0,05	10.166,00	17,25
24-31	0,2085	0,4350	0,3561	2	Alta	Baixa	99,25	0,06	9.500,00	20,98
32-42	0,2127	0,4267	0,3603	2	Alta	Baixa	98,50	0,06	9.500,00	21,14
43-50	0,2789	0,2946	0,4265	3 (não contratado)	Média	Média	100,00	0,04	-	-

Fonte: Elaborado pelo Autor.

Durante as simulações foram inseridos eventos de falhas nas entregas dos recursos sem que nenhuma ação dinâmica, como tentativa de recuperação ou renegociação do SLA, fosse tomada. No evento 6 ocorreu uma violação de entrega da métrica confiabilidade no SLA 3. O usuário informou a ocorrência de um volume de defeitos que reduziu a maturidade e consequentemente a confiabilidade entregue pelo SLA de “alta” para “média”. O SLAd@Cloud recalculou os pesos de cada SLA e passou a sugerir o SLA 2 a partir das próximas negociações (execução 7 a 14). Nenhuma ação de dinamicidade é realizada nesta avaliação. O histórico influenciou na decisão na razão de 50%.

A partir do evento 7 o SLA 2 é ofertado. O SLA 2 neste momento tem as melhores ofertas das métricas com maior importância para o usuário. A SLA 2 oferece usabilidade “baixa” en-

quanto o SLA 3 oferta usabilidade “média” que é o atual requisito válido para esta métrica. No entanto, a usabilidade representa um peso de 8,23%, enquanto que a confiabilidade representa um peso de 57,66% do total das métricas. Por isso, o SLAd@Cloud o SLA 2 pois a oferta ocorre em função da prioridade das métricas informadas pelo usuário. O tempo de execucão nas simulações 7 a 14 aumentou em relação as execuções 1 a 6. Isto foi decorrente do tempo necessário para atendimento de cada requisição ter sido 0,01s maior no SLA 2 que no SLA 3. O custo total médio manteve-se o mesmo apesar do SLA 3 ter custo superior ao SLA 2 para oferta das garantias. Isto explica-se porque com a violação gerada no evento 6 parte do valor cobrado retorna ao usuário como multa, equalizando os valores médios.

Na execução 14 ocorreu a redução do atendimento da métrica disponibilidade. A disponibilidade nas execuções anteriores era de 100%. A partir da execução 14 passou a ser de 99,25%. Nenhuma violação ocorreu porque este valor de disponibilidade atende os requisitos informados pelo usuário no SLA. O cálculo dos pesos indicou que o SLA 2 continua sendo sugerido por ter o maior peso, porém pode-se notar uma redução no valor do peso deste SLA em decorrência da redução do atendimento de uma de suas métricas. Outro ponto interessante, é que mesmo sem a ocorrência de violações, o usuário foi notificado da redução do nível de atendimento. Isto foi possível porque o componente de Avaliação do Serviço do SLAd@Cloud permite a configuração de alertas com expressões semelhantes as configuradas no SLA. Assim, o usuário pode ser avisado de eventos que podem anteceder violações do SLA.

Na execução 23 ocorreu a violação da métrica de desempenho. O SLA 2 tinha como requisito entregar cada requisição em até 0,05s e a partir desta execução as requisições obtiveram tempo de execução em 0,06s. Como resultado, ocorreram variações nos custos e tempo de execução total nas execuções de 24-31. Os pesos dos SLAs sofreram ajustes em decorrência do histórico da entrega, porém a influência ocorreu numa métrica de menor relevância o que não foi suficiente para que o SLAd@Cloud sugerisse outro SLA. O custos financeiros são menores devido aos valores decorrentes das multas pelas violações.

Na execução 31 ocorreu, em adição a violação da métrica de desempenho, nova redução do atendimento da métrica disponibilidade. Assim, pode-se notar nas execuções de 32 a 42, a redução da disponibilidade para o valor de 98,50%, ou seja, ocorreram violações também dessa métrica nessas execuções. O cálculo dos pesos apontaram nova redução do peso do SLA 2, mas este ainda manteve o peso superior aos demais SLAs.

Na execução 42 ocorreu a violação da métrica confiabilidade do SLA 2. Ele passou a entregar confiabilidade média ao invés de alta. Esta métrica tem o maior peso, assim os pesos foram recalculados e o SLA 3 obteve o maior peso total em função das métricas usabilidade, desempenho e disponibilidade maiores. Porém, neste ponto da simulação, não existem mais SLAs com históricos de atendimento da métrica confiabilidade superiores ao requisitado pelo usuário. Assim nas execuções de 43 a 50 nenhum SLA é sugerido ou contratado.

É possível concluir nessa avaliação que o SLAd@Cloud realizou a negociação de SLAs em nuvens SaaS oferecendo sempre o SLA que melhor atendia os requisitos priorizados pelo

usuário. O sistema utilizou o motor de priorizações e classificou os SLAs de acordo com pesos. Ao perceber violações em métricas de maior prioridade (execução 6) ofertou SLAs que melhor atendiam os requisitos de maior peso (quando disponível). Violações em métricas de menor prioridade (execuções 23 e 31) não afetaram as ofertas dos SLAs. O SLAd@Cloud foi capaz de utilizar métricas qualitativas realizando sua priorização, monitoramento e tomada de decisões como realizou com as métricas quantitativas. Outra observação realizada, foi que, o SLAd@Cloud não ofertou SLA com histórico de execução que apontasse resultado inferior aos requisitos de QoS solicitados pelo usuário.

5.3 Análise da dinamicidade - Desempenho e Custo

Até este ponto foi possível perceber que o SLAd@Cloud é capaz de negociar múltiplas métricas e oferecer para contratação o SLA que melhor atende os requisitos priorizados pelo usuário. Esta negociação ocorreu, na avaliação da Seção 5.2, somente nos momentos **antes** da inicialização dos serviços. Durante a prestação dos serviços não houve interação, o SLA permaneceu estático. Na Subseção 3.1.3 entendemos que o método estático descreve uma interação entre os usuários com os servidores de nuvem sem qualquer tipo **renegociação dinâmica** do SLA. Esta seção apresenta outra proposta do presente trabalho, uma abordagem dinâmica do SLA.

Na abordagem dinâmica, a interação entre o usuário e o provedor ocorre com a ajuda do SLAd@Cloud e o SLA pode ser renegociado de maneira dinâmica **durante** a prestação dos serviços. Cada provedor possui os registros de SLAs disponíveis, seus respectivos custos e oferece ao usuário SLAs que melhor atendem seus requisitos de QoS. Mesmo após contratar os serviços, o usuário pode renegociar o SLA com o provedor e migrar de SLA caso este não atenda sua necessidade. O resultado esperado utilizando este método é aumentar o cumprimento dos requisitos de QoS do usuário . Nesta avaliação foram utilizados os mesmos cenários e configurações da Seção 5.2.1. Assim, será possível avaliar o ganho total obtido com a negociação e dinamicidade do SLAd@Cloud. Durante a simulação foram avaliadas situações de *estresse* no sistema e de aumento de recursos por parte dos provedores para avaliar o sistema quanto a decisões de oferta de novos SLAs durante a prestação dos serviços.

5.3.1 Aplicação da dinamicidade no modelo IaaS - Infraestrutura como Serviço

Nessa subseção, a avaliação demontrará a abordagem dinâmica no modelo IaaS. Na avaliação da Seção 5.2.1 foram utilizadas as métricas de disponibilidade, utilização de CPU, memória e seus respectivos custos financeiros. Nela foi possível avaliar a capacidade de negociação de múltiplas métricas que ocorriam antes da inicialização dos serviços mas nenhuma ação dinâmica foi executada durante a execução dos serviços. Na presente avaliação foram adicionadas ao mesmo cenário as ações dinâmicas. O SLAd@Cloud executa ações como adição ou remo-

ção de recursos para manter os requisitos de QoS do usuário. A Figura 27 exibe um arquivo de exemplo com a configuração das ações. A primeira linha da figura significa que a métrica CPU é monitorada e caso o seu valor alcance 90% do valor requerido no SLA, a operação *setcpu+* pode ser chamada para aumentar a quantidade de CPUs em uma unidade.

Figura 27: Exemplo de arquivo de configuração de ações dinâmicas.

```
<?xml version="1.0" encoding="UTF-8"?>
<actions>
 <action metric="cpu"          if="greater" threshold="90%" op="setcpu+"  template="A">1</action>
 <action metric="cpu"          if="less"    threshold="50%" op="setcpu-"  template="A">1</action>
 <action metric="memory"       if="greater" threshold="90%" op="setmem+"  template="A">1</action>
 <action metric="memory"       if="less"    threshold="50%" op="setmem-"  template="A">1</action>
 <action metric="responsetime" if="greater" threshold="90%" op="newvm"    template="A">1</action>
 <action metric="responsetime" if="less"    threshold="50%" op="removevm" template="A">1</action>
</actions>
```

Fonte: Elaborado pelo autor.

Outro diferencial do SLAd@Cloud, é que, antes de invocar qualquer ação, ele avalia o impacto de cada uma delas a luz da priorização das métricas informada pelo usuário no momento da contratação. Desta forma garante-se que a ação tomada estará de acordo com as métricas priorizadas pelo usuário. Por exemplo, se a métrica "custo" tiver maior importância, o SLAd@Cloud reduzirá a alocação de recursos ao mínimo possível buscando o menor custo. Neste contexto, a primeira linha da Figura 27 é executada caso o valor obtido pelo monitoramento indique que o uso de CPU alcançou 90% e a métrica de CPU tem maior relevância que o custo financeiro. Isto só é possível porque o SLAd@Cloud trabalha com múltiplas métricas e com um sistema de priorização destas métricas. Dentro os trabalhos relacionados estudados, este é o único trabalho com esta preocupação.

A Figura 28 exibe um exemplo do racional da avaliação de impacto da ação para tomada de decisão. Nessa figura, pode-se observar que cada ação é avaliada no contexto das garantias ofertadas, obtendo-se notas da escala de comparação de critérios (SAATY, 1980). Por exemplo, pode-se observar que adicionar novas VMs no contexto da métrica A (quantidade de CPU) tem pesos maiores que manter e remover VMs respectivamente. Já no contexto da métrica D (custo financeiro da CPU), adicionar novas VMs tem peso menor que manter e remover VMs respectivamente.

Figura 28: Avaliação de impacto da ação nas métricas.

Métrica A	newvm	removevm	keepvm
newvm	1,00	3,00	3,00
removevm	0,33	1,00	3,00
keepvm	0,33	0,33	1,00

Métrica B	newvm	removevm	keepvm
newvm	1,00	3,00	3,00
removevm	0,33	1,00	3,00
keepvm	0,33	0,33	1,00

Métrica C	newvm	removevm	keepvm
newvm	1,00	3,00	3,00
removevm	0,33	1,00	3,00
keepvm	0,33	0,33	1,00

Métrica D	newvm	removevm	keepvm
newvm	1,00	0,33	0,33
removevm	3,00	1,00	0,33
keepvm	3,00	3,00	1,00

Métrica E	newvm	removevm	keepvm
newvm	1,00	0,33	0,33
removevm	3,00	1,00	1,00
keepvm	3,00	1,00	1,00

Fonte: Elaborado pelo autor.

A Figura 29 exibe o cálculo realizado após a obtenção dos pesos. É possível ver que, após a comparação, o motor de priorizações do SLAd@Cloud entra em ação novamente. A parte a) demonstra os pesos relativos de cada ação em cada métrica e com estes pesos, na parte b) é possível calcular qual ação mais adequada de acordo com as priorizações de métricas fornecidas pelo usuário. O peso relativo de cada ação (parte a) é multiplicado pelos pesos globais de cada métrica (coluna "peso global" na Tabela 11).

Figura 29: Pesos resultantes da ação após avaliação.

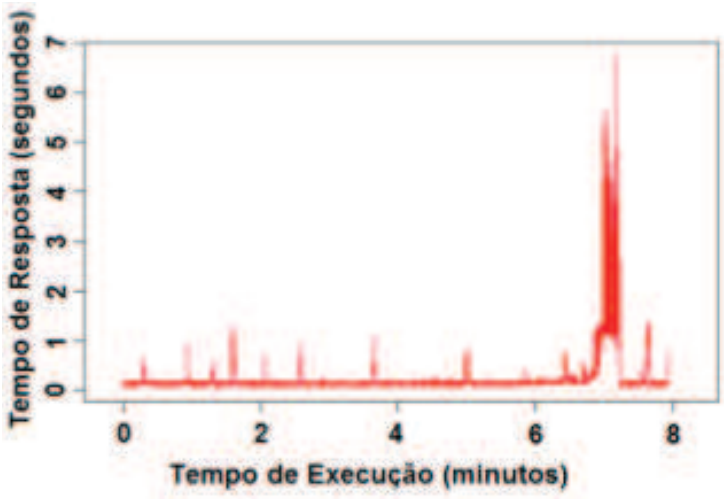
a	newvm	removevm	keepvm
A	0,6056	0,1033	0,2911
B	0,6056	0,1033	0,2911
C	0,6056	0,1033	0,2911
D	0,1033	0,6056	0,2911
E	0,1033	0,6056	0,2911

b	newvm	removevm	keepvm
A	0,1817	0,0310	0,0873
B	0,1817	0,0310	0,0873
C	0,1756	0,0300	0,0844
D	0,0057	0,0333	0,0160
E	0,0057	0,0333	0,0160
Peso	0,5503	0,1586	0,2911

Fonte: Elaborado pelo autor.

As Tabelas 10, 11, 12 e 13 demonstram as configurações realizadas na nuvem simulada. As 50 execuções da simulação foram repetidas. Durante as simulações foram inseridos eventos de falhas nas entregas dos recursos nas mesmas execuções da Seção 5.2.1, porém com ações de dinamicidade ativadas. No evento 17 ocorreu uma falha de entrega das métricas de CPU e memória. O gráfico na Figura 30 mostra a oferta de CPU e o SLA durante essa execução. Diferente do experimento anterior, o SLAd@Cloud ao detectar a falha de entrega de CPU executou uma ação de recuperação, neste caso a adição de mais uma VM, evitando a violação da métrica de CPU. A adição de mais uma VM gerou custo adicional, porém o custo foi definido pelo usuário como uma métrica de menor peso, o que justifica a ação.

Figura 30: Comportamento do SLA dinâmico.



Fonte: Elaborado pelo autor.

Os resultados obtidos nas simulações são demonstrados na Tabela 21. Nessa tabela, a co-

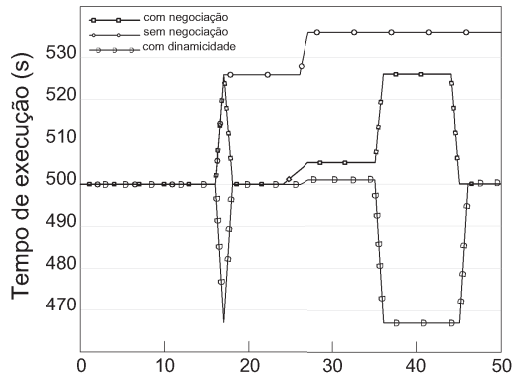
luna “a” exibe o identificador da execução. As colunas de “b”, “c” e “d” demostram o resultado da comparação dos pesos calculados para cada SLA na simulação. A coluna “e” exibe qual foi o SLA selecionado nesta negociação. As colunas de “f”, “g” e “h” mostram a quantidade de recursos entregues (em percentual referente ao que foi solicitado). Esta quantidade de recursos entregues compõem o histórico de execução para decisões futuras. As colunas “i” e “j” apresentam os custos indivivuais dos recursos em determinada execução. A coluna “k” mostra o custo do SLA, ou seja, o custo cobrado pelo fornecedor para dar a garantia adicional. A coluna “l” exibe o custo da violação, se esta aconteceu, enquanto a coluna “m” exibe a métrica que foi violada. E as colunas “n” e “o” mostram respectivamente o custo e o tempo total da execução.

No evento 27 foram geradas falhas de entrega da métricas de disponibilidade. Diferente do experimento anterior, o SLAd@Cloud ao detectar a interupção do serviço executou uma ação de recuperação, neste caso a substituição da VM defeituosa, evitando a violação da métrica de disponibilidade.

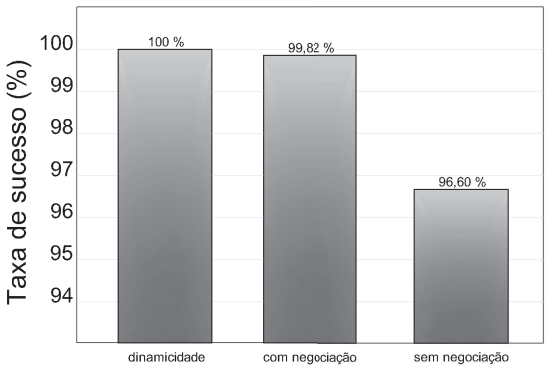
A Figura 31 apresenta os resultados com o tempo de execução e a taxa de sucesso da abordagem dinâmica em comparação com as demais. O gráfico na parte 31a da figura, mostra que o menor tempo de execução é realizado na abordagem dinâmica. Este resultado é possível porque o SLAd@Cloud manteve e recuperou o SLA escolhido que neste caso priorizou a métrica disponibilização de CPU.

Figura 31: Resultados da avaliação de dinamicidade do SLA em IaaS

(a) Tempo de execução



(b) Taxa de sucesso



Fonte: Elaborado pelo Autor.

O gráfico na parte 31b da figura, mostra as taxas de sucesso nas três abordagens. É possível perceber que as ações de manutenção dos requisitos de QoS foram tomadas no mesmo momento em que ocorreram as falhas e os requisitos de QoS foram atendidos nos SLAs sem que ocorressem violações. Com estas ações obteve-se 100% de sucesso na execução das *cloulets*.

Tabela 21: Parâmetros observados nas avaliações de dinamicidade para IaaS.

Exec. (a)	Pesos calculados			SLA selecionado (e)	Recursos entregues (%)			Custos			Penalidade		Custo total (n)	Tempo (s) (o)
	SLA 1 (b)	SLA 2 (c)	SLA 3 (d)		CPU (f)	MEM (g)	DISP (h)	CPU (i)	MEM (j)	SLA (k)	Custo (l)	Métrica (m)		
1	0,1819	0,3738	0,4443	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
2	0,2726	0,3512	0,3762	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
3	0,2726	0,3512	0,3762	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
4	0,2726	0,3512	0,3762	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
5	0,2726	0,3512	0,3762	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
6	0,2726	0,3512	0,3762	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
7	0,2726	0,3512	0,3762	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
8	0,2726	0,3512	0,3762	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
9	0,2726	0,3512	0,3762	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
10	0,2726	0,3512	0,3762	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
11	0,2726	0,3512	0,3762	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
12	0,2726	0,3512	0,3762	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
13	0,2726	0,3512	0,3762	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
14	0,2726	0,3512	0,3762	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
15	0,2726	0,3512	0,3762	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
16	0,2726	0,3512	0,3762	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
17	0,2395	0,3684	0,3921	3	105,00	105,00	100,00	50,00	56,32	0,05	0,00	0	111,63	496,19
18	0,2395	0,3684	0,3921	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
19	0,2395	0,3684	0,3921	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
20	0,2395	0,3684	0,3921	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
21	0,2395	0,3684	0,3921	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
22	0,2395	0,3684	0,3921	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
23	0,2395	0,3684	0,3921	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
24	0,2395	0,3684	0,3921	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
25	0,2395	0,3684	0,3921	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
26	0,2395	0,3684	0,3921	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
27	0,2627	0,3562	0,3811	3	100,00	100,00	99,80	50,00	51,20	0,05	0,00	0	106,26	501,00
28	0,2627	0,3562	0,3811	3	100,00	100,00	99,80	50,00	51,20	0,05	0,00	0	106,26	501,00
29	0,2627	0,3562	0,3811	3	100,00	100,00	99,80	50,00	51,20	0,05	0,00	0	106,26	501,00
30	0,2627	0,3562	0,3811	3	100,00	100,00	99,80	50,00	51,20	0,05	0,00	0	106,26	501,00
31	0,2627	0,3562	0,3811	3	100,00	100,00	99,80	50,00	51,20	0,05	0,00	0	106,26	501,00
32	0,2627	0,3562	0,3811	3	100,00	100,00	99,80	50,00	51,20	0,05	0,00	0	106,26	501,00
33	0,2627	0,3562	0,3811	3	100,00	100,00	99,80	50,00	51,20	0,05	0,00	0	106,26	501,00
34	0,2627	0,3562	0,3811	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	501,00
35	0,2627	0,3562	0,3811	3	100,00	100,00	99,80	50,00	51,20	0,05	0,00	0	106,26	501,00
36	0,2627	0,3562	0,3811	3	100,00	100,00	99,80	50,00	51,20	0,05	0,00	0	106,26	500,00
37	0,2627	0,3562	0,3811	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
38	0,2627	0,3562	0,3811	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
39	0,2627	0,3562	0,3811	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
40	0,2627	0,3562	0,3811	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
41	0,2627	0,3562	0,3811	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
42	0,2627	0,3562	0,3811	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
43	0,2627	0,3562	0,3811	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
44	0,2627	0,3562	0,3811	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
45	0,2627	0,3562	0,3811	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
46	0,2627	0,3562	0,3811	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
47	0,2627	0,3562	0,3811	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
48	0,2627	0,3562	0,3811	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
49	0,2627	0,3562	0,3811	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00
50	0,2627	0,3562	0,3811	3	100,00	100,00	100,00	50,00	51,20	0,05	0,00	0	106,26	500,00

Fonte: Elaborado pelo Autor.

5.4 Considerações Finais

O objetivo desse capítulo foi de apresentar os procedimentos metodológicos que foram efetuados para fazer a verificação e avaliação, bem como a análise dos resultados do SLAd@Cloud. A avaliação se deu através de implementação de um protótipo em Java que proveu a interface

de definição e contratação do SLA e o gerenciamento do SLA e do seu ciclo de vida, mesmo após a contratação, utilizando monitoramento integrado à provisão dos serviços. O protótipo interagiu com o *framework* CloudSim, um simulador de ambientes de nuvens computacionais.

As avaliações foram realizadas em duas etapas, numa verificou-se a eficiência do mecanismo de negociar um SLA com múltiplas métricas utilizando a classificação e priorização destes métricas em diferentes modelos de computação em nuvem e na outra verificou-se a capacidade do SLAd@Cloud de oferecer um SLA dinâmico realizando ações de recuperação e renegociação do SLA em tempo de execução dos serviços.

A avaliação da primeira etapa, a negociação do SLA, realizada na Seção 5.2 utilizou, no modelo IaaS, como requisitos de QoS métricas de disponibilidade, utilização de CPU, mémoria e seus custos financeiros. No modelo SaaS, foram utilizados como requisitos de QoS métricas de usabilidade, confiabilidade, desempenho, disponibilidade e seus custos financeiros. Cada métrica foi informada pelo usuário com seu respectivo requerimento, indicando o requisito ou expressão a ser cumprida e também seu peso ou importância relevantes. Três SLAs diferentes foram configurados para serem ofertados pelo provedor em cada caso. Em cada cenário foram executadas 50 execuções da simulação e entre essas execuções foram inseridos eventos de falhas nas entregas dos recursos.

Na avaliação, o SLAd@Cloud auxiliou o usuário, através da interface de entrada, a informar qual métrica e grupo de métrica tinha maior ou menor relevância que outro, utilizando a escala de comparação de critérios da Tabela 7. Os dados referentes a comparação foram processados, por um motor de priorizações, resultando nos pesos das métricas e os serviços foram inicializados para atender as tarefas na nuvem de acordo com os pesos informados pelo usuário. A cada execução ou evento de falha ocorrido, os SLAs eram reavaliados, com base nas métricas priorizadas, e o SLA com maior peso, ou mais aderente aos requisitos de QoS era ofertado para negociação garantindo o melhor cumprimento dos requisitos. Os resultados demonstraram ganhos no tempo de execução, nos custos financeiros e na taxa de sucesso de cumprimento das tarefas, que superou o modelo que tradicional ou que não trabalha com múltiplas métricas, em 3,40%. Cabe destacar também a capacidade do motor de priorizações de trabalhar com métricas SaaS, qualitativas. Um motor *fuzzy* auxiliou na normalização e conversão destas métricas em escalas comparáveis.

A avaliação da segunda etapa, a dinamicidade do SLA, realizada na Seção 5.3 utilizou, no modelo IaaS, os requisitos de QoS disponibilidade, utilização de CPU, mémoria e seus custos financeiros. Cada métrica foi informada pelo usuário com seu respectivo requerimento e os três SLAs da avaliação anterior de IaaS foram mantidos como oferta. Em cada cenário foram executadas 50 execuções da simulação e entre essas execuções foram repetidos os eventos de falhas nas entregas dos recursos.

A avaliação demontrou que o SLAd@Cloud oferece uma abordagem dinâmica do SLA. Nesta abordagem, a execução dos serviços foi monitorada e durante estas execuções ações de recuperação do atendimento dos requisitos de QoS foram disparadas. Vale ressaltar que estas

ações interagiram com o motor de priorizações, ou seja, foram classificadas de acordo com a importância de cada métrica dada pelo usuário.

Nos experimentos, o SLAd@Cloud coordenou a adição de mais VMs quando percebeu uma queda no tempo de resposta das tarefas. Nos resultados observou-se menores tempos de execução das tarefas e aumento da taxa de sucesso de realização destas tarefas.

É importante salientar que, como descrito na Seção 4.3, o modelo auxilia desde a contratação do acordo e sua manutenção até a finalização, seguindo o ciclo do SLA. A aplicabilidade dos requisitos definidos pelo usuário, bem como sua aderência as suas necessidades não é tratada pelo modelo. Assim é possível que o usuário defina e priorize requisitos que não contribuirão para o melhor desempenho das aplicações. Como o SLAd@Cloud permite a entrada dos requisitos em interface de alto nível a possibilidade de falhar humanas é reduzida.

Pode-se concluir finalmente que o SLAd@Cloud realizou a negociação dos SLAs em nuvens IaaS e SaaS oferecendo sempre o SLA que melhor atendia os requisitos priorizados pelo usuário. O monitoramento foi realizado e ações dinâmicas foram disparadas em consonância com os requisitos priorizados. O tratando as múltiplas métricas com o sistema de priorização e dinamicidade do SLAd@Cloud proporcionou maior aderência dos serviços desejados pelo usuário que as abordagens estática e que não trabalham com múltiplas métricas.

6 CONCLUSÃO

Considerando que o número de serviços ofertados nas nuvens computacionais vem crescendo a dificuldade no gerenciamento da oferta e comercialização dos serviços e das suas garantias tende também a ficar cada vez mais complexa. A partir do momento em que as aplicações críticas do negócio dos usuários são migradas para a nuvem, SLAs que acompanhem as características dinâmicas da computação em nuvem são de suma importância.

Os provedores de computação em nuvem ainda não são capazes de prover os acordos de nível de serviço que garantam a qualidade e eficiência aderente aos requisitos de QoS dos usuários. Os SLAs existentes nas nuvens comerciais são estáticos e por isto não acompanham a elasticidade da nuvem.

Assim, espera-se que soluções de SLA para computação em nuvem potencializem o cumprimento de QoS dos serviços considerando a priorização das métricas dada pelo usuário e os aspectos desejáveis e existentes de QoS dos serviços, tudo isso suportado por um intensivo uso monitoramento e gestão do ciclo de vida do SLA, que é dinâmico.

Desta forma é extremamente relevante, a questão de pesquisa realizada na Seção 1.4: “Como assegurar a qualidade dos serviços de computação em nuvem utilizando um acordo de nível de serviço (SLA) que possa ser contratado dinamicamente e adaptar-se automaticamente tanto as necessidades do consumidor quanto a capacidade dos provedores?”.

O avanço científico deste trabalho, na resposta a essa questão, foi a concepção de um modelo de negociação dinâmica de SLA para computação em nuvem que integra de forma transparente e ágil a priorização de múltiplas métricas, específicas de computação em nuvem, com o monitoramento e realização de ações para manter as garantias acordadas.

Os trabalhos existentes foram estudados e as lacunas identificadas foram tratadas no modelo proposto. Um protótipo foi desenvolvido e avaliado em três cenários simulados que envolveram diferentes modelos de nuvens computacionais.

Em síntese, tratou-se os aspectos de avaliação de múltiplas métricas de computação em nuvem com um motor de priorização que considera a importância de cada métrica e realiza ações para manter o SLA de acordo com esta importância. Um motor de inferências *fuzzy* atua no modelo tratando das métricas específicas de computação em nuvem, especialmente as de difícil mensuração, como no modelo SaaS e a dinamicidade, essencial para os serviços nuvens computacionais foi provida pelo monitoramento fortemente integrado com ações orientadas pelos requisitos de QoS do SLA.

6.1 Contribuições

Considerando as avaliações realizadas foi possível chegar a conclusão que o modelo proposto neste trabalho, o SLAd@Cloud, endereçou as lacunas identificadas nos trabalhos existentes. Um resumo das características do SLAd@Cloud e sua comparação com os traba-

lhos estudados pode ser visto na Tabela 22. A primeira contribuição aparece no motor de priorizações, baseado na metodologia de comparação par a par (SAATY, 1980) que habilitou o SLAd@Cloud a tratar **múltiplas métricas** simultaneamente e identificar o grau de importância de cada métrica em relação a outra. Esta importância é convertida em pesos e esses pesos são os guias nas decisões de negociação e renegociação do SLA que ocorrem respeitando o ciclo de vida do SLA. Nenhum dos trabalhos estudados trabalha com múltiplas métricas.

A segunda contribuição aparece no que tange ao modelo SaaS e SLA. Percebeu-se que os trabalhos estudados praticamente não atacam este aspecto. Isto dá-se pela complexidade de tratamento de **métricas específicas de computação em nuvem**, especialmente as qualitativas, de difícil mensuração, como "usabilidade" e "confiabilidade". Este foi o principal desafio encontrado ao longo do desenvolvimento deste trabalho. Para solucionar este problema foi inserido no modelo um motor de inferências *fuzzy*. Assim, o SLAd@Cloud inclui as bases para tratamento de tais aspectos e permite a comparação de métricas qualitativas e o monitoramento do cumprimento dessas métricas. Esta mesma solução pode ser aplicável ao modelo PaaS, pela natureza qualitativa de suas métricas.

Além disso, o SLAd@Cloud provê um sistema de SLA **dinâmico** para serviços de computação em nuvem. Enquanto os modelos tradicionais, como as nuvens comerciais, tratam o SLA de uma forma estática, o SLAd@Cloud com o monitoramento integrado realiza ações como alocação de recursos, remoção de recursos e renegociação do SLA dinamicamente durante a prestação dos serviços. Com estas ações o modelo obtém a dinamicidade necessária para tratar a elasticidade inerente a computação em nuvem. O SLA pode requisitar ou liberar recursos com a mudança dos requisitos de suas métricas que as ações registradas no SLAd@Cloud são executadas, para garantir que não ocorreram violações do SLA na entrega dos serviços. O DeSVI (EMEAKAROHA et al., 2010b) é o único trabalho estudado com alguma dinamicidade, porém sua dinamicidade não está vinculada ao monitoramento e não tratou os SLAs nos modelos SaaS e PaaS.

Adicionamente, o SLAd@Cloud, demonstrou capacidade de **converter requisitos de baixo para alto nível**, num dos experimentos por exemplo, monitorando o "tempo de resposta" e convertendo a ação em "quantidade de CPU" necessária. Neste ponto, ele equiparou-se ao LOM2HIS (EMEAKAROHA et al., 2010a) e a arquitetura de SLA para nuvens de pesquisa SaaS (NIEHöRSTER et al., 2010).

6.2 Trabalhos Futuros

Alguns aspectos não puderam ser focados com profundidade, como o tempo de resposta ou custo da negociação do SLA. Apesar de ser extremamente relevante num cenário real, foi considerado de "menor impacto" neste estudo, que basicamente visou investigar uma abordagem dinâmica do SLA, a viabilidade do uso de múltiplas métricas e o uso de métricas específicas de computação em nuvem. Este ponto está relacionado ao desempenho do algoritmo de com-

Tabela 22: Comparação do SLAd@Cloud com os trabalhos estudados.

Modelo	Interface	Métricas	Dinamic.	Monit.	Pontos Fortes	Pontos Fracos	Modelo
LoM2HIS	Alto nível	Apenas infraestrutura e não trabalha com múltiplas métricas	Estático	Sim	Permite realizar a contratação de SLA em linguagem de mais alto nível	Foco pontual na linguagem para parser dos requisitos para métricas e não permite a negociação do SLA	IaaS
DeSVI	Baixo nível	Apenas infraestrutura e não trabalha com múltiplas métricas	Dinâmico com limitação	Sim	*deployer* automático de VMs	Não trata aplicações com grande variabilidade de consumo de recursos. Não trata monitoramento e violação de SLA no nível de aplicações.	IaaS
SRV	Baixo nível	Genéricas para ambientes virtualizados e não trabalha com múltiplas métricas	Estático	Não	Negociação	É genérico para ambientes virtualizados por isso não trata especificades da nuvem	NA
Arq. Nuvens SaaS	Alto nível	SLA não renegociável e não trabalha com múltiplas métricas	Estático	Sim	Estimativa de Custo	Não trata negociação dinâmica do SLA	SaaS
SLAd@Cloud	Alto nível	Múltiplas métricas simultâneas e métricas específicas de cloud	Sim	Dinâmico	Sistema de priorização para tratar múltiplas métricas, SLA dinâmico e integrado ao monitoramento.	Funcionalidades básicas de monitoramento	IaaS, PaaS e SaaS

Fonte: Elaborado pelo autor.

paração (par a par), já que sua ordem de complexidade computacional é quadrática. Este fator de complexidade é preocupante para aplicações cujos dados de entrada possam crescer como no caso do SLAd@Cloud. Em função disto, percebe-se a necessidade de um apronfundamento que permita melhorar a eficiência do mecanismo de comparação par a par.

Embora fora do escopo deste trabalho, outro ponto importante é a realização de avaliação utilizando middlewares de nuvens comerciais. Este é um próximo passo para que o modelo possa ser aplicado no futuro em cenários reais empresariais.

E finalmente, tendo em vista a importância para o desenvolvimento deste trabalho, foram implementadas as funcionalidades básicas do monitoramento para uso do protótipo. Estas funcionalidades foram descritas nos componentes de avaliação do serviço. Em futuros trabalhos, novas funcionalidades devem ser implementadas para um gerenciamento mais intuitivo dos atuais módulos.

REFERÊNCIAS

AKHANI, J.; CHUADHARY, S.; SOMANI, G. Negotiation for resource allocation in IaaS cloud. In: FOURTH ANNUAL ACM BANGALORE CONFERENCE, 2011, Bangalore, India. Proceedings. . . ACM, 2011. p. 15:1–15:7. (COMPUTE ’11).

ALHAMAD, M.; DILLON, T.; CHANG, E. Conceptual SLA framework for cloud computing. In: DIGITAL ECOSYSTEMS AND TECHNOLOGIES (DEST), 2010, Dubai, United Arab Emirates. Anais. . . IEEE Computer Society, 2010. p. 606–610.

ANDRIEUX, A.; CZAJKOWSKI, K.; DAN, A.; KEAHEY, K.; LUDWIG, H.; NAKATA, T.; PRUYNE, J.; ROFRANO, J.; TUECKE, S.; XU, M. Web Services Agreement Specification (WS-Agreement). Boston, USA: Global Grid Forum, 2005.

ARENAS, A.; WILSON, M. Contracts as Trust Substitutes in Collaborative Business. Compu- ter, Los Alamitos, CA, USA, v. 41, n. 7, p. 80–83, 2008.

ARMBRUST, M.; FOX, A.; GRIFFITH, R.; JOSEPH, A. D.; KATZ, R. H.; KONWINSKI, A.; LEE, G.; PATTERSON, D. A.; RABKIN, A.; STOICA, I.; ZAHARIA, M. Above the Clouds: a berkeley view of cloud computing. Berkeley, California: EECS Department, University of California, 2009. (UCB/EECS-2009-28).

BEINAT, E.; NIJKAMP, P. Multicriteria Analysis for Land-Use Management. [S.l.]: Sprin- ger, 2007. (Environment & Management).

BOSSARD, E. Envisioning neighborhood quality of life using conditions in the neigh- borhood access to and from conditions in the surroungding region. [S.l.]: Computers in Urban Planning and Urban Management on the Edge of the Millenium., 1999.

BRANTNER, M.; FLORESCU, D.; GRAF, D.; KOSSMANN, D.; KRASKA, T. Building a da- tabase on S3. In: ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, 2008., 2008, Vancouver, Canada. Proceedings. . . ACM, 2008. p. 251–264. (SIG- MOD ’08).

BREITMAN, K.; ENDLER, M.; PEREIRA, R.; AZAMBUJA, M. When TV Dies, Will It Go to the Cloud? Computer, Los Alamitos, CA, USA, v. 43, n. 4, p. 81–83, Apr. 2010.

BUYYA, R.; YEO, C. S.; VENUGOPAL, S.; BROBERG, J.; BRANDIC, I. Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Gener. Comput. Syst., Amsterdam, The Netherlands, The Netherlands, v. 25, n. 6, p. 599–616, June 2009.

CABLE, S.; TOUSSAINT, A.; MODI, T.; GALBRAITH, B.; MILBURY, J.; BASHA, J. Pro- fessional Java Web Services. 1st. ed. Birmingham, UK, UK: Wrox Press Ltd., 2002.

CALHEIROS, R. N.; RANJAN, R.; BELOGLAZOV, A.; DE ROSE, C. A. F.; BUYYA, R. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evalu- ation of resource provisioning algorithms. Softw. Pract. Exper., New York, NY, USA, v. 41, n. 1, p. 23–50, Jan. 2011.

CANCIAN, M. H.; RABELO, R. J.; WANGENHEIM, C. G. von. Uma proposta para elabo- ração de Contrato de Nível de Serviço para Software-as-a-Service (SaaS). 2009.

CANEDO, E.; SOUSA JUNIOR, R. de; OLIVEIRA ALBUQUERQUE, R. de; MENDONCA, F. de. File Exchange in a Private Cloud Supported by a Trust Model. In:

CYBER-ENABLED DISTRIBUTED COMPUTING AND KNOWLEDGE DISCOVERY (CYBERC), 2012 IN- TERNATIONAL CONFERENCE ON, 2012, Sanya, China. Anais. . . IEEE Computer Society, 2012. p. 89–96.

CASE, J. D.; FEDOR, M.; SCHOFFSTALL, M. L.; DAVIN, J. Simple Network Management Protocol (SNMP). United States: RFC Editor, 1990.

CERBELAUD, D.; GARG, S.; HUYLEBROECK, J. Opening the clouds: qualitative overview of the state-of-the-art open source vm-based cloud management platforms. In: ACM/IFIP/U- SENIX INTERNATIONAL CONFERENCE ON MIDDLEWARE, 10., 2009, Urbanna, Illinois. Proceedings. . . Springer-Verlag New York: Inc., 2009. p. 22:1–22:8. (Middleware ’09).

CHOU, C. huei. Article: design and implementation of jahp: a java-based analytic hierarchy process application. International Journal of Computer Applications, New York, USA, v. 62, n. 15, p. 35–41, January 2013. Published by Foundation of Computer Science.

CINGOLANI, P.; ALCALA-FDEZ, J. jFuzzyLogic: a robust and flexible fuzzy-logic inference system language implementation. In: FUZZY SYSTEMS (FUZZ-IEEE), 2012 IEEE INTER- NATIONAL CONFERENCE ON, 2012, Brisbane, Australia. Anais. . . IEEE Computer Society, 2012. p. 1–8.

CIURANA, E. Developing with Google App Engine (Firstpress). 1st. ed. [S.l.]: Apress, 2009.

COMELLAS, J. O. F.; PRESA, I. G.; FERNANDEZ, J. G. SLA-driven Elastic Cloud Hos- ting Provider. In: EUROMICRO CONFERENCE ON PARALLEL, DISTRIBUTED AND NETWORK-BASED PROCESSING, 2010., 2010, Washington, DC, USA. Proceedings... IEEE Computer Society, 2010. p. 111–118. (PDP ’10).

COSTA, R.; BRASILEIRO, F.; SOUZA FILHO, G. L. de; SOUSA, D. M. Just in Time Clouds: enabling highly-elastic public clouds over low scale amortized resources. Cam- pina Grande, Paraíba, Brazil: Universidade Federal de Campina Grande, 2010.

COX, E. The fuzzy systems handbook: a practitioner’s guide to building, using, and maintai- ning fuzzy systems. San Diego, CA, USA: Academic Press Professional, Inc., 1994.

EMEAKAROHA, V. C.; BRANDIC, I.; MAURER, M.; DUSTDAR, S. Low level Metrics to High level SLAs - LoM2HiS framework: bridging the gap between monitored metrics and sla parameters in cloud environments. In: HIGH PERFORMANCE COMPUTING AND SI- MULATION (HPCS), 2010 INTERNATIONAL CONFERENCE ON, 2010, Caen, France. Anais. . . IEEE Computer Society, 2010. p. 48 –54.

EMEAKAROHA, V.; CALHEIROS, R.; NETTO, M.; ROSE IVONA BRANDIC, C. D. DeSVi: an architecture for detecting sla violations in cloud computing infrastructures. 2nd Internatio- nal ICST Conference on Cloud Computing (CloudComp 2010), Barcelona, Spain, 2010.

FOUQUET, M.; NIEDERMAYER, H.; CARLE, G. Cloud computing for the masses. In: ACM WORKSHOP ON USER-PROVIDED NETWORKING: CHALLENGES AND OPPORTUNI- TIES, 1., 2009, Rome, Italy. Proceedings. . . ACM, 2009. p. 31–36. (U-NET ’09).

Fuzzy Thinking: the new science of fuzzy logic. [S.l.]: Hyperion, 1994. GARFINKEL, S. L. An Evaluation of Amazon's Grid Computing Services: ec2, s3 and sqs.

[S.l.]: Center for, 2007.

JONES, C. Software Quality in 2012: a survey of the state of the art. In: SOFTWARE PRO- DUCTIVITY RESEARCH LLC, 2012. Anais... Software Quality Group of New England, 2012.

JUNG, J.; KIM, H. MR-CloudSim: designing and implementing mapreduce computing model on cloudsim. In: ICT CONVERGENCE (ICTC), 2012 INTERNATIONAL CONFERENCE ON, 2012. Anais. . . [S.l.: s.n.], 2012. p. 504–509.

KARIN, B.; ANDRé, C. Computação na Nuvem. In: ATUALIZAçõES EM INFORMáTICA, RIO DE JANEIRO: PUC RIO., 2010, Rio de Janeiro, Brasil. Anais. . . PUC Rio, 2010. p. 331 –386. In Portuguese.

KELLER, A.; LUDWIG, H. The WSLA Framework: specifying and monitoring service level agreements for web services. J. Netw. Syst. Manage., New York, NY, USA, v. 11, n. 1, p. 57– 81, Mar. 2003.

KERTESZ, A.; KECSKEMETI, G.; BRANDIC, I. An SLA-based resource virtualization ap- proach for on-demand service provision. In: VIRTUALIZATION TECHNOLOGIES IN DIS- TRIBUTED COMPUTING, 3., 2009, Barcelona, Spain. Proceedings. . . ACM, 2009. p. 27–34. (VTDC '09).

LAW, A. M.; KELTON, D. W. Simulation Modelling and Analysis. [S.l.]: McGraw-Hill Edu- cation - Europe, 2000.

LENK, A.; KLEMS, M.; NIMIS, J.; TAI, S.; SANDHOLM, T. What's inside the Cloud? An architectural map of the Cloud landscape. In: ICSE WORKSHOP ON SOFTWARE ENGI- NEERING CHALLENGES OF CLOUD COMPUTING, 2009., 2009, Washington, DC, USA. Proceedings. . . IEEE Computer Society, 2009. p. 23–31. (CLOUD '09).

LI, K.; XU, G.; ZHAO, G.; DONG, Y.; WANG, D. Cloud Task Scheduling Based on Load Balancing Ant Colony Optimization. In: CHINAGRID CONFERENCE (CHINAGRID), 2011 SIXTH ANNUAL, 2011, Dalian, China. Anais. . . IEEE Computer Society, 2011. p. 3–9.

LLORENTE, I. M.; VOZMEDIANO, M. Open Nebula Cloud Computing Framework. 2009. MALCZEWSKI, J. GIS and Multicriteria Decision Analysis. [S.l.]: Wiley, 1999.

MARTHALER, V. Function Point Counting Practices Manual Release 4.2 Handbook. New Jersey, USA: International Function Point Users Group, 2004.

MELL, P.; GRANCE, T. The NIST Definition of Cloud Computing. Gaithersburg, MD: Na- tional Institute of Standards and Technology (NIST), 2011. (800-145).

MEYERS, L.; GAMST, G.; GUARINO, A. Applied Multivariate Research: design and inter- pretation. [S.l.]: SAGE Publications, 2006.

NETO, L. B.; COELHO, P. H. G. Minicurso de sistema especialista nebuloso. 2006. NIEHöRSTER, O.; BRINKMANN, A.; FELS, G.; 0002, J. K.; SIMON, J. Enforcing SLAs in

Scientific Clouds. In: CLUSTER, 2010. Anais. . . IEEE Computer Society, 2010. p. 178–187.

NURMI, D.; WOLSKI, R.; GRZEGORCZYK, C.; OBERTELLI, G.; SOMAN, S.; YOUSEFF, L.; ZAGORODNOV, D. Eucalyptus : a technical report on an elastic utility computing archi- tecture linking your programs to useful systems. 2008.

OGUNMEFUN, T. Cloud computing service models. [S.l.: s.n.], 2011.

PREEZ, D. du. Companies struggling with cloud performance. [S.l.: s.n.], 2010.

RAWAT, P. S.; SAROHA, G. P.; BARTHWAL, V. Article: quality of service evaluation of saas modeler (cloudlet) running on virtual cloud computing environment using cloudsim. In- ternational Journal of Computer Applications, New York, USA, v. 53, n. 13, p. 35–38, September 2012.

REDKAR, T. Windows Azure Platform. Berkely, CA, USA: Apress, 2010.

RIMAL, B. P.; CHOI, E.; LUMB, I. A Taxonomy, Survery, and Issues of Cloud Compu- ting Ecosystems. In: cloud computing: principles, systems and applications. 1st. ed. [S.l.]: Springer, 2010.

ROSS, J. W.; WESTERMAN, G. Preparing for utility computing: the role of it architecture and

relationship management. IBM Syst. J., Riverton, NJ, USA, v. 43, n. 1, p. 5–19, Jan. 2004. SAATY, T. The Analytic Hierarchy Process: planning, priority setting, resource allocation.

[S.l.]: McGraw-Hill, 1980. (Advanced book program).

SAP. SAP's Standardized Technical Architecture Modeling. [S.l.]: SAP, 2007.

SEMPOLINSKI, P.; THAIN, D. A Comparison and Critique of Eucalyptus, OpenNebula and Nimbus. In: CLOUD COMPUTING TECHNOLOGY AND SCIENCE (CLOUDCOM), 2010 IEEE SECOND INTERNATIONAL CONFERENCE ON, 2010, Indianapolis, Indiana, USA. Anais. . . IEEE Computer Society, 2010. p. 417 –426.

SEVERANCE, C. Using Google App Engine. 1st. ed. [S.l.]: O'Reilly Media, Inc., 2009.

SHI, Y.; JIANG, X.; YE, K. An Energy-Efficient Scheme for Cloud Resource Provisioning Based on CloudSim. In: CLUSTER COMPUTING (CLUSTER), 2011 IEEE INTERNATIO- NAL CONFERENCE ON, 2011, Austin, Texas, USA. Anais. . . IEEE Computer Society, 2011. p. 595–599.

SMIT, M.; STROULIA, E. Maintaining and evolving Service Level Agreements: motivation and case study. In: MESOCA, 2011. Anais. . . IEEE Computer Society, 2011. p. 1–9.

SOSINSKY, B. Cloud Computing Bible. 1st. ed. [S.l.]: Wiley Publishing, 2011.

SOUSA, F. R. C.; MOREIRA, L. O.; MACHADO, J. C. Gerenciamento de Dados em Nu- vem: conceitos, sistemas e desafios. in: xxv simpósio brasileiro de banco de dados, 2010, belo horizonte. sbbd. Belo Horizonte, Minas Gerais, Brasil: [s.n.], 2010.

STEFAN RIED, H. K. Sizing the Cloud, Understanding and Quantifying The Future of Cloud Computing. Forrester Research, [S.l.], 2011.

STOEGERER, C.; NOVAK, T.; KASTNER, W.; KRAMMER, L. Procedure-based availability SLAs for Traffic Management Systems. In: ETFA, 2011. Anais. . . IEEE Computer Society, 2011. p. 1–8.

SUMTER, L. Cloud computing: security risk. In: ANNUAL SOUTHEAST REGIONAL CONFERENCE, 48., 2010, Oxford, Mississippi. Proceedings. . . ACM, 2010. p. 112:1–112:4. (ACM SE ’10).

Sá, T. T.; SOARES, J. M.; GOMES, D. G. CloudReports: uma ferramenta gráfica para a simu- lação de ambientes computacionais em nuvem baseada no framework cloudsim. Mato Grosso do Sul, Brasil: SBRC, 2011.

TABOADA, G. L.; TOURIñO, J.; DOALLO, R. Java for high performance computing: asses- sment of current research and practice. In: INTERNATIONAL CONFERENCE ON PRINCI- PLES AND PRACTICE OF PROGRAMMING IN JAVA, 7., 2009, Calgary, Alberta, Canada. Proceedings. . . ACM, 2009. p. 30–39. (PPPJ ’09).

UNDHEIM, A.; CHILWAN, A.; HEEGAARD, P. Differentiated Availability in Cloud Com- puting SLAs. In: IEEE/ACM 12TH INTERNATIONAL CONFERENCE ON GRID COMPU- TING, 2011., 2011, Washington, DC, USA. Proceedings... IEEE Computer Society, 2011. p. 129–136. (GRID ’11).

VAQUERO, L. M.; RODERO-MERINO, L.; CACERES, J.; LINDNER, M. A break in the clouds: towards a cloud definition. SIGCOMM Comput. Commun. Rev., New York, NY, USA, v. 39, n. 1, p. 50–55, Dec. 2008.

VELTE, T.; VELTE, A.; ELSENPETER, R. Cloud Computing, A Practical Approach. 1st. ed. [S.l.]: McGraw-Hill Osborne Media, 2009.

VOORSLUYS, W.; BROBERG, J.; VENUGOPAL, S.; BUYYA, R. Cost of Virtual Machine Live Migration in Clouds: a performance evaluation. CoRR, [S.l.], v. abs/1109.4974, 2011.

WU, L.; BUYYA, R. Service Level Agreement (SLA) in Utility Computing Systems. CoRR, Melbourne, Australia, v. abs/1010.2881, 2010.

YOUSEFF, L.; BUTRICO, M.; DA SILVA, D. Toward a Unified Ontology of Cloud Computing. In: GRID COMPUTING ENVIRONMENTS WORKSHOP, 2008. GCE ’08, 2008, Austin, Texas. Anais. . . IEEE Computer Society, 2008. p. 1–10.

ZADEH, L. A. Fuzzy sets. Information and Control, [S.l.], v. 8, p. 338–353, 1965.