Modelos de negócio para ambientes de computação
em nuvem que consideram atributos de qos
relacionados a desempenho e a segurança

Bruno Guazzelli Batista

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Bruno Guazzelli Batista

Modelos de negócio para ambientes de computação em nuvem que consideram atributos de qos relacionados a desempenho e a segurança

Tese apresentada ao Instituto de Ciências
Matemáticas e de Computação – ICMC-USP,
como parte dos requisitos para obtenção do título
de Doutor em Ciências – Ciências de Computação e
Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e
Matemática Computacional

Orientadora: Profa. Dra. Regina Helena
Carlucci Santana

USP – São Carlos
Janeiro de 2016

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

B898m Batista, Bruno Guazzelli
Modelos de negócio para ambientes de computação em
nuvem que consideram atributos de qos relacionados a
desempenho e a segurança / Bruno Guazzelli Batista;
orientadora Regina Helena Carlucci Santana. - São
Carlos - SP, 2016.
142 p.

Tese (Doutorado - Programa de Pós-Graduação em
Ciências de Computação e Matemática Computacional)
- Instituto de Ciências Matemáticas e de Computação,
Universidade de São Paulo, 2016.

1. Computação em Nuvem. 2. Qualidade de Serviço.
3. Modelo de Negócio. 4. Provisionamento de Recursos.
5. Mecanismos de Segurança. I. Santana, Regina Helena
Carlucci, orient. II. Título.

Bruno Guazzelli Batista

Business models for cloud computing environments that
consider qos attributes related to performance and security

Doctoral dissertation submitted to the Instituto de
Ciências Matemáticas e de Computação – ICMC-USP,
in partial fulfillment of the requirements for the degree
of the Doctorate Program in Computer Science and
Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and
Computational Mathematics

Advisor: Profa. Dra. Regina Helena Carlucci Santana

USP – São Carlos
January 2016

AGRADECIMENTOS

Agradeço primeiramente à Deus pelas oportunidades que me são dadas, pela minha família e pelos desafios que aparecem no meu caminho, pois são eles que me fazem crescer e me motivam a alcançar meus sonhos.

À minha família, especialmente à minha mãe Marita por sempre estar ao meu lado em todos os momentos, sejam eles de alegria ou de tristeza, pela estrutura e apoio dado ao longo de minha vida e por sempre acreditar em mim.

Aos meus irmãos Camila e Ângelo, ao meu pai Geraldo e à minha avó Hilda, que juntos à minha mãe, são a motivação de meu viver.

À minha grande parceira Nathália, pela paciência, compreensão, companheirismo e ótimos momentos juntos.

À todos os amigos e integrantes do LASDPC pelos momentos de descontração e pelas contribuições ao longo dessa jornada.

Ao pessoal da Universidade de Leicester na Inglaterra, em especial ao Professor Stephan Reiff-Marganiec, por todo o apoio e estrutura dados durante o período em que estive lá.

Agradeço também ao programa Ciências sem Fronteiras pela bolsa concedida para desenvolvimento do estágio de doutorado sanduíche na Universidade de Leicester.

Agradeço à todo o corpo docente que integra o LASDPC, em especial a minha orientadora Regina Helena Carlucci Santana pela orientação, dedicação e pela oportunidade de trabalharmos juntos na realização deste trabalho.

Agradeço à FAPESP, CNPq e CAPES pelo apoio financeiro que permitiu ao aluno dedicação exclusiva ao desenvolvimento do projeto.

Ao Instituto de Ciências Matemáticas e de Computação (ICMC), formado por todos os funcionários que sempre se mostraram disponíveis nos momentos em que os solicitei durante o desenvolvimento deste trabalho.

Enfim, agradeço a todos que direta ou indiretamente contribuíram para a minha formação pessoal e profissional.

RESUMO

BATISTA, B. G.. **Modelos de negócio para ambientes de computação em nuvem que consideram atributos de qos relacionados a desempenho e a segurança**. 2016. 142 f. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

Este projeto de doutorado tem como objetivo definir modelos de negócio para ambientes de computação em nuvem que consideram desempenho e segurança como atributos de qualidade de serviço durante a definição do contrato. Para isso, foi necessário quantificar o impacto causado no desempenho de um ambiente em nuvem quando diferentes mecanismos de segurança foram utilizados. Para a quantificação da sobrecarga foram utilizadas técnicas e metodologias disponíveis na literatura que visam garantir a integridade, disponibilidade e confidencialidade dos dados, abordando desafios que envolvem o acesso, armazenamento e manipulação de dados em serviços oferecidos por meio de máquinas virtuais. Experimentos executados possibilitaram analisar o comportamento das variáveis de resposta na utilização de cenários com diferentes mecanismos de segurança e cargas. Dessa forma, foi possível confrontar a sobrecarga imposta pelos mecanismos de segurança com a alteração da quantidade de recursos aplicada por um módulo proposto, chamado ReMM. De acordo com os resultados, o ReMM alterou a quantidade de recursos virtuais alocados utilizando dois algoritmos de escalabilidade, garantindo as exigências definidas no contrato de níveis de serviço. No entanto, a alteração dos recursos computacionais para contrapor a sobrecarga imposta pelos mecanismos de segurança impactou nos custos finais dos serviços. Dessa forma, a sobrecarga de segurança, desempenho e custo foram considerados na definição dos modelos de negócios em diferentes ambientes de computação em nuvem.

Palavras-chave: Computação em Nuvem, Qualidade de Serviço, Modelo de Negócio, Provisão de Recursos, Mecanismos de Segurança.

ABSTRACT

BATISTA, B. G.. **Modelos de negócio para ambientes de computação em nuvem que consideram atributos de qos relacionados a desempenho e a segurança.** 2016. 142 f. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

This PhD thesis has as main goal defining business models for cloud computing environments that consider performance and security as quality of service attributes during the service level agreement definition. For this, it was necessary quantifying the impact on the performance of a cloud environment in which different security mechanisms were applied. Techniques and methodologies available in the literature that aim ensuring the integrity, availability and confidentiality of data were used to quantify the overhead, addressing challenges related to access, storage and manipulation of data in services offered through virtual machines. Experiments were executed, in which the response variable behaviors were analyzed, using scenarios with different security mechanisms and workloads. In this way, it was possible to compare the overhead imposed by the security mechanisms with the changes in the quantity of resources applied by a module proposed, called ReMM. According to the results, the ReMM changed the amount of allocated virtual resources using two scalability algorithms, ensuring the requirements defined in service level agreement. However, the changes in the computational resources to face the overhead imposed by the security mechanisms influenced the final costs of the service. Therefore, security overhead, performance and cost were considered in the definition of business models in different cloud computing environments.

Key-words: Cloud Computing, Quality of Service, Business Model, Resources Provisioning, Security Mechanisms.

LISTA DE ILUSTRAÇÕES

Figura 1 – Serviços oferecidos em computação em nuvem, adaptado de (RIMAL; CHOI; LUMB, 2009)	11
Figura 2 – Tipos de nuvem	12
Figura 3 – Criptografia de chave simétrica, adaptado de (ROSENBERG; REMY, 2004)	35
Figura 4 – Criptografia de chave assimétrica, adaptado de (ROSENBERG; REMY, 2004)	36
Figura 5 – Assinatura digital utilizando criptografia de chave pública e função <i>hash</i> , adaptado de (ROSENBERG; REMY, 2004)	39
Figura 6 – Verificação da assinatura digital, adaptado de (ROSENBERG; REMY, 2004)	40
Figura 7 – Funcionamento do ReMM	73
Figura 8 – Pseudocódigos dos algoritmos de escalabilidade disponíveis no ReMM	75
Figura 9 – Média de requisições atendidas por cada VM em um ambiente com 8GB de disco	78
Figura 10 – Uso do processador	79
Figura 11 – Média de requisições atendidas por cada VM em um ambiente com 16GB de disco	80
Figura 12 – Influência dos fatores	81
Figura 13 – Tempo médio de execução em segundos	82
Figura 14 – Tempo de execução vs número de vCPUs	83
Figura 15 – Margem de variação do SLA	85
Figura 16 – Tempos Médios de Execuções	86
Figura 17 – Custo por hora pago por um cliente	87
Figura 18 – Diagrama de sequência para o processamento de uma requisição de serviço (CENTURION, 2015)	96
Figura 19 – Operações realizadas pelo ReMM no CloudSim-BEQoS	96
Figura 20 – Relação entre provedores, clientes e usuários	98
Figura 21 – Tempos médios de execuções de requisições sem <i>overhead</i> de segurança – Cenário 1	103
Figura 22 – Custos Médios Finais por Hora sem <i>overhead</i> de segurança – Cenário 1	105
Figura 23 – Número de requisições atendidas sem a presença de mecanismos de segurança – Cenário 1	106
Figura 24 – Tempos médios de execuções das requisições com <i>overhead</i> de segurança – Cenário 1	106
Figura 25 – Custos Médios Finais por Hora com <i>overhead</i> de segurança – Cenário 1	107

Figura 26 – Número de requisições atendidas com a presença de mecanismos de segurança – Cenário 1	109
Figura 27 – Tempos médios de execuções de requisições sem <i>overhead</i> de segurança – Cenário 2	112
Figura 28 – Custos Médios Finais por Hora sem <i>overhead</i> de segurança – Cenário 2 . .	113
Figura 29 – Número de requisições atendidas sem a presença de mecanismos de segurança – Cenário 2	114
Figura 30 – Tempos médios de execuções de requisições com <i>overhead</i> de segurança – Cenário 2	115
Figura 31 – Número de requisições atendidas com a presença de mecanismos de segurança – Cenário 2	116
Figura 32 – Custos Médios Finais por Hora com <i>overhead</i> de segurança – Cenário 2 . .	117

LISTA DE TABELAS

Tabela 1 – Algoritmos criptográficos de chave simétrica, adaptado de (TANENBAUM, 2003)	36
Tabela 2 – Sobrecarga imposta por diferentes mecanismos de segurança sobre o desempenho dos ambientes	62
Tabela 3 – Fatores e níveis para os experimentos com o <i>benchmark Apache</i>	77
Tabela 4 – Fatores e níveis para os experimentos com o <i>benchmark Smallpt</i>	77
Tabela 5 – Especificação do ambiente	77
Tabela 6 – Planejamento dos experimentos	84
Tabela 7 – Resultados dos experimentos	88
Tabela 8 – Quantidade de recursos utilizados	89
Tabela 9 – Comparativo dos percentuais de aumento e de redução em relação ao ambiente Comum	90
Tabela 10 – Especificações dos cenários analisados	100
Tabela 11 – Especificação das máquinas físicas presentes nos <i>data centers</i>	100
Tabela 12 – Especificação das instâncias	100
Tabela 13 – Fatores e níveis	101
Tabela 14 – Resultados obtidos em ambientes sem <i>overhead</i> de segurança – Cenário 1	103
Tabela 15 – Variáveis de resposta em um ambiente Comum, sem <i>overhead</i> de segurança – Cenário 1	104
Tabela 16 – Variáveis de resposta em um ambiente Vertical, sem <i>overhead</i> de segurança – Cenário 1	104
Tabela 17 – Variáveis de resposta em um ambiente Horizontal, sem <i>overhead</i> de segurança – Cenário 1	105
Tabela 18 – Quantidade de recursos utilizados e impacto sobre o número de requisições atendidas no Cenário 1 sem segurança	105
Tabela 19 – Resultados obtidos em ambientes com <i>overhead</i> de segurança	107
Tabela 20 – Variáveis de resposta em um ambiente com escalabilidade vertical e <i>overhead</i> de segurança – Cenário 1	108
Tabela 21 – Variáveis de resposta em um ambiente com escalabilidade horizontal e <i>overhead</i> de segurança – Cenário 1	108
Tabela 22 – Quantidade de recursos utilizados e impacto sobre o número de requisições atendidas no Cenário 1 com segurança	109

Tabela 23 – Percentuais de influência dos fatores nos ambientes Comum e Vertical – Cenário 1	110
Tabela 24 – Percentuais de influência dos fatores nos ambientes Comum e Horizontal – Cenário 1	110
Tabela 25 – Percentuais de influência dos fatores nos ambientes Vertical e Horizontal – Cenário 1	111
Tabela 26 – Resultados obtidos em ambientes sem <i>overhead</i> de segurança – Cenário 2 .	113
Tabela 27 – Variáveis de resposta em um ambiente com escalabilidade vertical, sem <i>overhead</i> de segurança – Cenário 2	113
Tabela 28 – Variáveis de resposta em um ambiente com escalabilidade horizontal sem <i>overhead</i> de segurança – Cenário 2	114
Tabela 29 – Quantidade de recursos utilizados e impacto sobre o número de requisições atendidas no Cenário 2 sem segurança	114
Tabela 30 – Resultados obtidos em ambientes com <i>overhead</i> de segurança – Cenário 2 .	115
Tabela 31 – Quantidade de recursos utilizados e impacto sobre o número de requisições atendidas no Cenário 2 com segurança	116
Tabela 32 – Variáveis de resposta em um ambiente com escalabilidade vertical e <i>overhead</i> de segurança – Cenário 2	117
Tabela 33 – Variáveis de resposta em um ambiente com escalabilidade horizontal e <i>overhead</i> de segurança – Cenário 2	117
Tabela 34 – Percentuais de influência dos fatores nos ambientes Comum e Vertical – Cenário 2	118
Tabela 35 – Percentuais de influência dos fatores nos ambientes Comum e Horizontal – Cenário 2	119
Tabela 36 – Percentuais de influência dos fatores nos ambientes Vertical e Horizontal – Cenário 2	119
Tabela 37 – Tabela comparativa dos TMEs (em segundos) em um ambiente Comum . .	120
Tabela 38 – Percentuais aproximados de aumento (+) e de redução (-) exercidos pelo ReMM sobre as variáveis de resposta na tentativa de contrapor a sobrecarga de segurança e garantir o SLA no Cenário 1	121
Tabela 39 – Percentuais aproximados de aumento (+) e de redução (-) exercidos pelo ReMM sobre as variáveis de resposta na tentativa de contrapor o <i>overhead</i> e garantir o SLA no Cenário 2	122

LISTA DE ABREVIATURAS E SIGLAS

ABE		<i>Attribute-Based Encryption</i>
ACID		Atomicidade, Consistência, Isolamento, Durabilidade
ACL		<i>Access Control List</i>
ACPS		<i>Access Control mechanism for P2P storage Cloud</i>
ACPS		<i>Advanced Cloud Protection System</i>
AISE		<i>Address-Independent Seed Encryption</i>
Amazon S3		<i>Amazon Simple Storage Service</i>
API		<i>Application Programming Interface</i>
ASD		<i>Australian Signals Directorate</i>
AWS		<i>Amazon Web Services</i>
CA		<i>Certification Authority</i>
CAPTCHA		<i>Completely Automated Public Turing test to tell Computers and Humans Apart</i>
CP-ABE	..	<i>Ciphertext-Policy Attribute-Based Encryption</i>
CSA		<i>Cloud Security Alliance</i>
CSMIC	...	<i>Cloud Service Measurement Index Consortium</i>
CTB		<i>Cloud Trace Back</i>
D-H		<i>Diffie-Hellman</i>
DDoS		<i>Distributed Denial of Service</i>
DoS		<i>Denial of Service</i>
DPM		<i>Deterministic Packet Marking</i>
ECC		<i>Elliptic Curve Cryptosystem</i>
ENISA	...	<i>European Network and Information Security Agency</i>
FIFO		<i>First In First Out</i>
GSI		<i>Grid Security Infrastructure</i>
GT4		<i>Globus Toolkit 4</i>
HMAC	...	<i>keyed-Hashing for Message Authentication Code</i>
HPC		<i>High Performance Computing</i>
HTTP		<i>HyperText Transfer Protocol</i>
HTTPS	...	<i>HyperText Transfer Protocol Secure</i>
IaaS		<i>Infrastructure as a Service</i>
IAM		<i>Identity and Access Management</i>

IdM *Identity Management*
 IDS *Intrusion Detection System*
 IPS *Intrusion Prevention System*
 IPSec *IP Security Protocol*
 IS *Infinite Server*
 ISM *Information Security Manual*
 ISO *International Organization for Standardization*
 KDC *Key Distribution Center*
 KVM *Kernel-based Virtual Machine*
 LASDPC . *Laboratório de Sistemas Distribuídos e Programação Concorrente*
 LDAP *Lightweight Directory Access Protocol*
 LIFO *Last In Last Out*
 MAC *Message Authentication Code*
 MAP_{IAT} .. *MAP for Inter-Arrival Time*
 MAP_{SD} ... *MAP for Service Demand*
 MAP_{sinc} .. *MAP Synchronized*
 MD5 *Message Digest Algorithm 5*
 MFA *Multi-Factor Authentication*
 MI *Milhões de Instruções*
 MIPS *Milhões de Instruções Por Segundo*
 MTCS SS . *Multi-Tier Cloud Security Standard*
 NIST *National Institute of Standards and Technology*
 NSA *National Security Agency*
 NSCC *Network Security for Cloud Computing*
 OpenLDAP *Open Lightweight Directory Access Protocol*
 P2P *Peer-to-Peer*
 PaaS *Platform as a Service*
 PCI DSS .. *Payment Card Industry Data Security Standard*
 PDA *Personal Digital Assistant*
 PDP *Policy Decision Point*
 PEP *Policy Enforcement Point*
 PKI *Public Key Infrastructure*
 PRE *Proxy Re-Encryption*
 PS *Processor Sharing*
 QoS *Quality of Service*
 RDP *Remote Desktop Protocol*
 REM *Reference Evaluation Methodology*

ReMM ... *Resource Management Module*
ROIA *Real-time Online Interactive Applications*
RR *Round Robin*
RSA *Rivest, Shamir, Adleman*
SaaS *Software as a Service*
SAML ... *Security Assertions Markup Language*
SD *Security Descriptor*
SecaaS ... *Security as a Service*
SecureME *Secure My Execution*
SHA-1 ... *Secure Hash Algorithm 1*
SIC *Security Inspection Chains*
SIRO *Service In Random Order*
SLA *Service Level Agreement*
SMD *Security Management Domains*
SMG *Security Meta-Group*
SMI *Service Measurement Index*
SOA *Service-Oriented Architecture*
SOC *Security Operations Center*
SOTA *Service-Oriented Traceback Architecture*
SPAD *Service Provider Attack Detection*
SSH *Secure Shell*
SSL *Secure Sockets Layer*
SSO *Single Sign-On*
SVM *Support Vector Machine*
TI *Tecnologia da Informação*
TLS *Transport Layer Security*
TME *Tempo Médio de Execução*
TMR *Tempo Médio de Resposta*
TPM *Trusted Platform Module*
TSAD *Tenant Specific Attack Detection*
TTP *Trusted Third Party*
vCPU *virtual CPU*
VM *Virtual Machine*
VMM *Virtual Machine Monitor*
VPN *Virtual Private Network*
XACML .. *eXtensible Access Control Markup Language*
XML *eXtensible Markup Language*

SUMÁRIO

1	INTRODUÇÃO	1
1.1	Contextualização	1
1.2	Motivação	3
1.3	Hipótese	4
1.4	Objetivos	4
1.5	Metodologia	5
1.6	Descrição dos Capítulos	6
2	COMPUTAÇÃO EM NUVEM	9
2.1	Considerações Iniciais	9
2.2	Arquitetura	10
2.3	Virtualização	13
2.3.1	Hipervisor	14
2.3.2	Estratégias de Virtualização	16
2.4	QoS em Computação em Nuvem	17
2.5	Modelos de Negócio em Computação em Nuvem	21
2.6	Considerações Finais	24
3	SEGURANÇA NA COMPUTAÇÃO EM NUVEM	27
3.1	Considerações Iniciais	27
3.2	Princípios para uma Nuvem Segura	28
3.3	Aspectos Críticos de Segurança em Nuvem	30
3.4	Mecanismos de Segurança	34
3.4.1	Criptografia	35
3.4.1.1	Chave simétrica	35
3.4.1.2	Chave pública	36
3.4.2	Assinatura Digital e Entidades Intermediárias	37
3.4.2.1	Assinaturas com chave simétrica	38
3.4.2.2	Assinaturas com chave assimétrica	38
3.4.2.3	Sumário de mensagens	39
3.4.3	Outros Mecanismos	41
3.4.3.1	Firewall	41
3.4.3.2	Deteção e prevenção de intrusão	42

3.4.3.3	<i>Protocolos que oferecem segurança</i>	42
3.5	Estado da Arte	44
3.5.1	Surveys	44
3.5.2	Segurança nos Provedores	46
3.5.2.1	<i>Amazon Web Services</i>	46
3.5.2.2	<i>Google Apps</i>	47
3.5.2.3	<i>Force.com</i>	48
3.5.2.4	<i>SmartCloud Enterprise</i>	49
3.5.3	Mecanismos Disponíveis na Literatura	51
3.6	Considerações Finais	53
4	CARACTERIZAÇÃO DA SOBRECARGA IMPOSTA POR MECANISMOS DE SEGURANÇA	55
4.1	Considerações Iniciais	55
4.2	Aspectos Críticos Abordados	56
4.3	Análise dos Mecanismos	58
4.3.1	<i>PerfCloud</i>	62
4.3.2	<i>CloudProof</i>	64
4.4	Considerações Finais	65
5	MÓDULO DE GERENCIAMENTO DE RECURSOS	67
5.1	Considerações Iniciais	67
5.2	Provisionamento de Recursos	68
5.3	Trabalhos Relacionados	70
5.4	ReMM – Resource Management Module	72
5.5	Análise de Desempenho no Provisionamento de Recursos em Nuvem 75	
5.5.1	Primeiro Conjunto de Experimentos – Definição da Taxa de Alteração dos Recursos	76
5.5.1.1	<i>Resultados com o benchmark Apache</i>	78
5.5.1.2	<i>Influência dos Fatores</i>	80
5.5.1.3	<i>Resultados com o benchmark Smallpt</i>	82
5.5.2	Segundo Conjunto de Experimentos – Ambiente Simulado com o ReMM	84
5.5.2.1	<i>Variáveis de Resposta</i>	85
5.5.2.2	<i>Análise dos Resultados</i>	86
5.6	Considerações Finais	90
6	AVALIAÇÃO DE DESEMPENHO E DEFINIÇÃO DE MODELOS DE NEGÓCIO PARA AMBIENTES EM NUVEM	93
6.1	Considerações Iniciais	93

6.2	Inserção do ReMM na Arquitetura CloudSim-BEQoS	94
6.3	Planejamento dos Experimentos	97
6.4	Análise dos Resultados	102
6.4.1	Cenário 1	102
6.4.1.1	<i>Experimentos sem sobrecarga de segurança – Cenário 1</i>	102
6.4.1.2	<i>Experimentos com sobrecarga de segurança – Cenário 1</i>	106
6.4.1.3	<i>Influência dos Fatores no Cenário 1</i>	109
6.4.2	Cenário 2	112
6.4.2.1	<i>Experimentos sem sobrecarga de segurança – Cenário 2</i>	112
6.4.2.2	<i>Experimentos com sobrecarga de segurança – Cenário 2</i>	115
6.4.2.3	<i>Influência dos Fatores no Cenário 2</i>	117
6.4.3	Modelos de Negócio	119
6.5	Considerações Finais	123
7	CONCLUSÕES	125
7.1	Considerações Iniciais	125
7.2	Conclusões Finais	126
7.3	Contribuições	127
7.4	Trabalhos Futuros	128
	Referências	131

INTRODUÇÃO

1.1 Contextualização

Nos últimos anos um dos tópicos mais discutidos na área de Tecnologia de Informação (TI) tem sido computação em nuvem. Segundo o NIST (*National Institute of Standards and Technology*), computação em nuvem é um modelo que permite ubiquidade, conveniência e acesso sob demanda para um conjunto de recursos compartilhados que são configuráveis e que podem ser rapidamente entregues com um esforço mínimo de gestão por parte dos usuários (MELL; GRANCE, 2011).

Subashini e Kavitha (2011) definem nuvem como um modelo de computação no qual os recursos de TI são fornecidos como um serviço aos clientes externos por meio da Internet. Os provedores desses recursos oferecem serviços sob demanda de forma transparente, uma vez que os clientes não têm ciência de onde e como esses serviços são executados.

A utilização de computação em nuvem permite aumentar dinamicamente a capacidade de prestação de serviços de uma empresa ou de atender às solicitações de serviços dos clientes, sem que estes precisem investir em infraestrutura como compra de *hardware*, de licenças de *software* ou treinamento de funcionários.

No entanto, por se tratar de um sistema distribuído que provê serviços, supõe-se que o sistema computacional que compõe uma nuvem opere apropriadamente, oferecendo desempenho tanto em termos de rapidez na resposta, quanto em termos de disponibilidade (minimizando a interrupção no oferecimento de serviço) e segurança (evitando perda de dados ou mensagens), a fim de conquistar a confiança e satisfação dos seus clientes. Para isso, os provedores de serviços devem garantir diferentes atributos de Qualidade de Serviço (*Quality of Service – QoS*).

O conceito de qualidade de serviço refere-se ao efeito provocado pelas características de desempenho de um serviço, ou seja, o conjunto de características de um sistema necessário

para atingir uma determinada funcionalidade (ESTRELLA; SANTANA, 2011) (TANENBAUM, 2011). QoS pode ser descrito ainda como um conjunto de atributos que mensuram a qualidade de um fluxo de dados específico, como por exemplo, a largura de banda ou a prioridade atribuída a um determinado cliente. Esses atributos de um sistema podem ser medidos quantitativamente por meio de métricas e utilizados na definição de níveis de QoS (WEBER, 2002). Entre as métricas pode-se citar o tempo médio de resposta, *throughput*, o tempo médio de execução, utilização de recursos, perda de pacotes, *jitter*, latência de rede, segurança, entre outros.

Garantir QoS em um ambiente de nuvem não é uma tarefa trivial, pois existem diversos perfis de clientes com as mais variadas exigências de prestação de serviços (CHARD, 2011) (ARDAGNA *et al.*, 2014). O processo de definição de QoS começa com o estabelecimento dos parâmetros exigidos pelos clientes. Esses parâmetros são mapeados e negociados entre os componentes do sistema, assegurando que todos devem atingir um nível de QoS aceitável, definindo assim um acordo de nível de serviço (*Service Level Agreement – SLA*). Após a definição das cláusulas do SLA, os recursos são alocados e monitorados, havendo a possibilidade de renegociação caso as condições do sistema se alterem (ARDAGNA *et al.*, 2014). Dessa forma, a qualidade de um serviço e o cumprimento dos SLAs têm sido tópicos de grande interesse nos últimos anos tanto no âmbito acadêmico como industrial.

Outro fator importante é a segurança de ambientes em nuvem. De acordo com Subashini e Kavitha (2011), pequenas e médias empresas têm investido cada vez mais em computação em nuvem, pois por meio dela pode-se obter acesso rápido a aplicações em qualquer lugar do mundo, exigindo apenas um terminal com conexão com a Internet. Com essa estratégia é possível aumentar consideravelmente os recursos de infraestrutura da empresa a um custo mais atrativo. Porém, preocupações relacionadas à segurança do ambiente podem impedir que novos clientes utilizem os serviços de uma nuvem (LOMBARDI; PIETRO, 2011) (VAQUERO; RODERO-MERINO; MORÁN, 2011) (BREIVOLD *et al.*, 2014) (HASHEM *et al.*, 2015) (SHAMSOLMOALI; ALAM, 2015).

À medida que as informações sobre os clientes e as empresas são migradas para a nuvem, as preocupações sobre o quão seguro esse ambiente é começam a crescer. Os provedores fornecem vários tipos de serviços, os quais apresentam problemas e desafios de segurança em níveis distintos, uma vez que cada um desses serviços, seja ele SaaS (*Software as a Service*), PaaS (*Platform as a Service*) ou IaaS (*Infrastructure as a Service*), é implementado em camadas de serviços diferentes e com isso cada camada tem que ser abordada de forma diferente (KANDUKURI; PATURI; RAKSHIT, 2009).

Dessa forma, definir como a segurança é aplicada nas camadas de serviços de um ambiente de computação em nuvem e a sua interferência na qualidade e no custo do serviço oferecido é uma tarefa que carece de mais estudos, uma vez que os principais provedores disponíveis no mercado não apresentam os dados relacionados a essa interferência. Mecanismos de segurança devem ser considerados em conjunto com atributos de QoS relacionados a desempenho, criando

assim uma relação entre a garantia de segurança, aliada a garantia de desempenho e o custo do serviço. Um fator importante a ser ressaltado nesta relação é que, em sistemas computacionais, segurança e desempenho geralmente são grandezas inversamente proporcionais.

1.2 Motivação

Em computação em nuvem, muitos aplicativos e dados são movidos para *data centers*, o que gera alguns desafios de segurança no gerenciamento dos dados e serviços (WANG *et al.*, 2010). Esses desafios incluem, mas não se limitam, a vulnerabilidades na acessibilidade, em aplicações *Web*, de acesso físico, privacidade, gerenciamento de identidades e de credenciais, verificação de dados, manipulação, integridade, confidencialidade, perdas ou roubos de dados, autenticação de dispositivos e na virtualização (VELEV; ZLATEVA, 2011).

Os clientes contratam serviços os quais são executados nas máquinas virtuais (VM – *Virtual Machines*)¹. Embora os clientes possam utilizar mecanismos como antivírus e sistemas de detecção de intrusão para proteger as suas VMs, as ameaças de segurança ainda são presentes. Um atacante em posse da máquina virtual pode desativar ou burlar as contramedidas de segurança e causar danos ao sistema.

Nos últimos anos alguns incidentes contribuíram para que as preocupações com o acesso, armazenamento e manipulação dos dados de forma segura na nuvem aumentassem. Por exemplo, em junho de 2011, uma atualização no sistema Dropbox afetou o seu mecanismo de autenticação, de forma que os arquivos dos clientes pudessem ser acessados utilizando senhas incorretas².

Outro fato ocorreu com a Amazon S3 em 2008, em que a depuração de uma falha no balanceador de carga dos *data centers* resultou na corrupção dos dados dos clientes, o que acarretou em falhas no processo de verificação da integridade dos arquivos³.

Em 2014, uma falha na biblioteca OpenSSL permitiu que conexões criptografadas pudessem ser expostas. Isso afetou a maioria dos sistemas operacionais Linux e BSD e, consequentemente, grandes fornecedores como Facebook, Tweeter, Amazon e aplicativos da Oracle, IBM, VMware, entre muitos outros. Cogitou-se que a NSA (*National Security Agency*) já conhecia essa vulnerabilidade e a utilizava para espionar conexões criptografadas e supostamente seguras⁴.

Há na literatura diversos trabalhos que apresentam soluções para as mais variadas ameaças presentes em um ambiente de computação em nuvem. No entanto, não foram encontrados trabalhos que definem modelos de negócio que consideram a avaliação do comportamento do

¹ O procedimento de virtualização é apresentado no Capítulo 2

² <http://www1.folha.uol.com.br/tec/932976-falha-no-dropbox-permite-acessar-sem-senha-algumas-contas.shtml>

³ <https://gigaom.com/2008/02/15/amazon-s3-service-goes-down/>

⁴ <http://www.wired.com/2014/04/nsa-heartbleed/>

desempenho de um serviço na aplicação de diferentes mecanismos de segurança.

Essa relação entre desempenho e segurança é importante, pois a cada dia novos dispositivos, acessórios e serviços surgem no mercado, os quais, além de facilitar a interatividade e o acesso a informação de maneira prática e rápida (exigindo assim um desempenho aceitável), requisitam aos clientes/usuários informações e acessos a dados muitas vezes confidenciais. Dessa forma, há uma carência por estudos que apresentem a relação entre desempenho e segurança de forma que ambos possam ser garantidos em conformidade com as exigências dos clientes.

1.3 Hipótese

A hipótese desta tese é formulada a partir da seguinte premissa: é possível compensar a sobrecarga imposta pelos mecanismos de segurança sobre o desempenho de um serviço por meio de mecanismos de provisionamento de recursos e renegociações dos custos. Sabe-se que, conforme níveis mais rígidos de segurança são necessários, a interferência dessas contramedidas, necessárias para manter a segurança, sobre o desempenho do sistema também cresce. No caso de computação em nuvem, pode-se compensar a sobrecarga gerada pelos mecanismos de segurança alterando a quantidade de recursos disponíveis durante o tempo de execução para executar determinado serviço. No entanto, a alteração da quantidade de recursos irá interferir no custo a ser pago pelo cliente. Desta forma, em se tratando de computação em nuvem, a relação entre desempenho e segurança deve considerar um novo parâmetro, que é o custo dos recursos alocados para a execução do serviço.

1.4 Objetivos

Para comprovar a hipótese desta tese, o principal objetivo a ser alcançado é definir modelos de negócio para ambientes em nuvem baseado em avaliações que consideram a sobrecarga imposta por mecanismos de segurança sobre o desempenho do serviço. Isto é, o objetivo do trabalho não é propor novas políticas de segurança para nuvem e contribuir para melhorar a segurança do sistema em algum aspecto, e sim contribuir na relação entre a obtenção de segurança e de desempenho, e verificar como essa relação interfere no modelo de negócio do ambiente. Para tanto, alguns objetivos específicos devem ser alcançados:

- Caracterização e quantificação da sobrecarga imposta por diferentes mecanismos de segurança sobre o desempenho de um ambiente em nuvem;
- Proposta e validação de um mecanismo para o gerenciamento dos recursos computacionais da nuvem;

- Definição de um limite em que um determinado mecanismo de segurança influencia negativamente no desempenho ou que infrinja o SLA;
- Avaliação de desempenho considerando diferentes mecanismos de segurança, analisando o impacto gerado no sistema com a alteração dos recursos computacionais alocados;
- Definição de modelos de negócio para ambientes em nuvem que relacionem a sobrecarga de segurança, desempenho e custo.

1.5 Metodologia

Para atingir o objetivo proposto, deve-se realizar uma avaliação considerando o impacto da sobrecarga imposta por diferentes mecanismos de segurança no desempenho de um serviço em nuvem. Com os resultados pode-se definir limites nos quais, por exemplo, um determinado mecanismo de segurança influencia negativamente no desempenho do serviço. A partir desse limite, pode-se verificar qual o impacto gerado no sistema com a alteração da quantidade de recursos computacionais a fim de atingir o desempenho contratado por um cliente, influenciando no custo a ser pago. Dessa forma, ao final deste estudo, serão definidos modelos de negócio para ambientes em nuvem que relacionem segurança, desempenho e custo.

Deste modo, a metodologia utilizada para o desenvolvimento desta tese possui como base a investigação da literatura e a utilização de ferramentas e APIs (*Application Programming Interface*) para a construção e análises de ambientes em nuvem por meio de técnicas de aferição e simulação. Assim, a metodologia proposta está organizada da seguinte forma:

- Estudo de trabalhos disponíveis na literatura que realizam uma avaliação de desempenho considerando diferentes mecanismos de segurança. Com os resultados pode-se caracterizar e quantificar a sobrecarga imposta pelos mecanismos analisados sobre o desempenho dos ambientes.
- Análise de diferentes mecanismos de provisionamento de recursos em nuvem, a qual permitirá a proposta de um novo módulo. Sabe-se que para um provisionamento eficiente, os recursos devem ser alocados e desalocados de acordo com a demanda de serviços, garantindo conformidade com as exigências feitas pelos clientes e o uso eficiente dos recursos. Dessa forma, serão realizadas análises por meio de técnicas de aferição para a definição de quais recursos computacionais e em que proporção eles devem ser alterados para atender a demanda de serviços. Com base nos resultados será possível definir dois algoritmos, um que aplica a escalabilidade vertical dos recursos e outro que aplica a horizontal. O primeiro algoritmo deverá alterar a quantidade de recursos computacionais que compõem uma máquina virtual, como núcleos virtuais, memória e/ou disco. O segundo deverá alterar o número de máquinas virtuais disponíveis no ambiente. Em seguida será

realizada a validação do módulo proposto utilizando um ambiente simulado utilizando o simulador CloudSim (CALHEIROS *et al.*, 2011).

- Após proposta e validação do módulo de gerenciamento de recursos e a caracterização e quantificação da sobrecarga imposta pelas contramedidas de segurança, será possível analisar ambientes confrontando a sobrecarga de segurança e o desempenho do ambiente. Por meio do monitoramento das execuções das requisições, o módulo de gerenciamento proposto poderá, quando necessário, alterar a quantidade de recursos disponíveis durante o tempo de execução, utilizando os seus algoritmos na tentativa de contrapor a sobrecarga imposta sobre o desempenho do serviço e garantir o SLA. No entanto, é importante considerar que a alteração dos recursos exerce impacto sobre os custos do serviço.
- Com os resultados das análises será possível definir modelos de negócio para os ambientes analisados que relacionem a sobrecarga de segurança, o desempenho e o custo.

1.6 Descrição dos Capítulos

Todo o conteúdo abordado neste projeto é apresentado em sete capítulos. No Capítulo 2 é descrita toda a fundamentação teórica sobre computação em nuvem que compõe a base para o desenvolvimento do projeto de doutorado, incluindo modelos de prestação de serviços, tipos de nuvens, virtualização, qualidade de serviço e modelos de negócio. No Capítulo 3 são apresentados os conceitos de segurança, bem como uma visão ampla das ameaças e contramedidas existentes na literatura para um ambiente em nuvem.

No Capítulo 4 é realizado um estudo sobre as ameaças de segurança abordadas neste projeto, bem como uma análise de trabalhos disponíveis na literatura que visam propor contramedidas para os aspectos críticos abordados. Baseado nessa análise, foi possível caracterizar e quantificar a sobrecarga imposta por diferentes mecanismos de segurança sobre o desempenho de um sistema em nuvem.

Visando contrapor essa sobrecarga, é proposto no Capítulo 5 um módulo de gerenciamento de recursos, chamado ReMM (*Resource Management Module*), que altera a quantidade de recursos disponíveis durante o tempo de execução para tentar garantir o SLA e o uso eficiente dos recursos, com impacto sobre o custo.

A metodologia utilizada no planejamento dos experimentos, bem como os resultados obtidos e os modelos de negócio definidos são apresentados no Capítulo 6. Por fim, no Capítulo 7 são apresentadas as conclusões e contribuições deste projeto de doutorado, bem como os direcionamentos para trabalhos futuros.

COMPUTAÇÃO EM NUVEM

2.1 Considerações Iniciais

Computação em nuvem é um modelo de fornecimento de serviços sob demanda baseado em técnicas já consolidadas como virtualização, computação distribuída, computação utilitária e computação autônoma. Esse modelo reduz a sobrecarga de tecnologia da informação para o cliente final, permitindo grande flexibilidade, além da redução do custo total na aquisição, gerenciamento e manutenção de toda a infraestrutura pertencente a uma empresa (REESE, 2009) (RITTINGHOUSE, 2009).

De acordo com Fox *et al.* (2009) e Armbrust *et al.* (2010), a nuvem pode ser considerada como um conjunto de serviços acessíveis pela Internet que visam fornecer os mesmos serviços disponíveis em um sistema de computação. A tecnologia envolvida consiste em compartilhar ferramentas computacionais pela interligação dos sistemas, semelhantes às nuvens, ao invés de ter essas ferramentas localmente.

O uso desse ambiente torna-se mais viável do que o uso de unidades físicas, pois as empresas não precisam se preocupar com o gerenciamento ou aquisição de toda a infraestrutura de *hardware* e *software* necessária para o provimento de serviços e, conseqüentemente, tem-se uma redução nos gastos. Assim, os provedores de computação em nuvem fornecem serviços *online* que são acessados por meio de outros serviços *Web* ou de *software*, como um *browser* por exemplo. Com isso, tem-se a abstração da rede e de toda a infraestrutura necessária para o armazenamento e processamento dos serviços fornecidos, e a infraestrutura passa a ser um serviço, no qual seus componentes devem estar disponíveis e acessíveis aos clientes.

Computação em nuvem introduz um novo nível de flexibilidade e escalabilidade nas organizações de TI. Esta flexibilidade ajuda a solucionar desafios que os clientes e provedores de serviços enfrentam, que incluem a rápida alteração de cenários de TI, redução do custo e tempo com gerenciamento de infraestrutura (CHONKA *et al.*, 2011) (BENSLIMANE *et al.*, 2014).

A arquitetura de uma nuvem é muito mais que um conjunto de computadores. Ela deve fornecer uma infraestrutura para gerenciamento que inclui características como: escalabilidade de recursos, balanceamento dinâmico de carga, monitoramento de desempenho, escalonamento do sistema e segurança. Ela oferece diversas vantagens aos seus clientes como (VELEV; ZLATEVA, 2011) (HASHEM *et al.*, 2015):

- Redução dos custos, pois os serviços são fornecidos sob demanda ao cliente com sistema de faturamento como, por exemplo, o *pay-as-you-use*, muito utilizado na computação utilitária;
- Alta abstração dos recursos;
- Escalabilidade e flexibilidade instantâneas;
- Provisionamento de recursos instantâneos;
- Compartilhamento de recursos como *hardware* e base de dados;
- Gerenciamento programado por meio de APIs de serviços *Web*;
- Aumento da mobilidade, pois as informações são acessadas de qualquer lugar;
- Ubiquidade, onde os serviços podem ser acessados por diversos dispositivos dotados de recursos computacionais como celulares, *smartphones* ou *netbooks*.

2.2 Arquitetura

Uma característica peculiar de computação em nuvem é o fato de ser baseada em serviços, os quais são fornecidos sob demanda por meio da Internet (CHIEU *et al.*, 2009). A nuvem em si é uma abstração de toda a infraestrutura lógica (*software*, plataformas de *middleware* ou *frameworks*) e física (*hardware*) de um provedor que oferece seus serviços cobrando por eles ou não.

Muitos estudos na literatura dividem os serviços prestados pela nuvem em três categorias (FOSTER *et al.*, 2008) (WANG *et al.*, 2008) (CHIEU *et al.*, 2009) (LEAVITT, 2009) (PENG *et al.*, 2009) (MELL; GRANCE, 2011) (VELEV; ZLATEVA, 2011) (WANG *et al.*, 2011) (BREIVOLD *et al.*, 2014) (DURAO *et al.*, 2014). A Figura 1 apresenta essas categorias com algumas opções de serviços disponíveis no mercado.

- **Software como Serviço (SaaS – Software as a Service):** é o *software* oferecido por um provedor de serviços, disponível sob demanda, geralmente por meio de um navegador *Web*. Os SaaS estão se tornando comuns e seu sucesso é devido principalmente à maior largura de banda que tornou possível a evolução da *Web* (conhecida como *Web 2.0*). Esta

Figura 1 – Serviços oferecidos em computação em nuvem, adaptado de (RIMAL; CHOI; LUMB, 2009)

evolução também oferece aplicações mais dinâmicas, tais como multimídia e serviços *Web*. Assim, um único *software* pode ser fornecido para vários clientes ao mesmo tempo, e esses clientes podem compartilhar informações e interagir com os outros sem a necessidade de instalar novos *softwares* em suas máquinas;

- **Plataforma como Serviço (PaaS – Platform as a Service):** permite que os clientes desenvolvam novas aplicações utilizando APIs, implementando e operando remotamente. A PaaS está em uma camada acima da camada de IaaS e adiciona uma camada de integração com os *frameworks* de desenvolvimento e funcionalidades como armazenamento de dados, comunicação e consultas que permitem que os desenvolvedores construam aplicações com as linguagens de programação desejadas;
- **Infraestrutura como Serviço (IaaS – Infrastructure as a Service):** fornece recursos computacionais por meio de máquinas virtuais, e outras abstrações de *hardware* e sistemas operacionais que podem ser controlados por meio de uma API. A IaaS incorpora a capacidade de abstração de recursos assim como toda a conectividade física e lógica desses recursos. Além disso, fornece um conjunto de APIs que permitem o gerenciamento e outras formas de interação com as infraestruturas desenvolvidas pelos clientes.

Uma nuvem pode ainda ser classificada conforme a sua localização e finalidade (RITTINGHOUSE, 2009) (VELEV; ZLATEVA, 2011). Na Figura 2 é possível ver as diferentes configurações que uma nuvem pode ter.

- **Nuvem Privada:** as nuvens privadas são aquelas construídas exclusivamente para um único cliente (uma empresa, por exemplo) sobre um *data center* privado. Diferentemente

Figura 2 – Tipos de nuvem

de um *data center* privado virtual, a infraestrutura física utilizada pertence ao cliente e, portanto, ele possui total controle sobre como os serviços são implementados;

- **Nuvem Pública:** a infraestrutura computacional é hospedada pelo provedor de serviços e é compartilhada entre todas as organizações (empresas) contratantes. O cliente não tem visibilidade e controle sobre onde essa infraestrutura computacional está hospedada;
- **Nuvem Comunitária:** a infraestrutura de uma nuvem é compartilhada por diversas organizações e suporta uma comunidade específica que partilha as mesmas preocupações como, por exemplo, a finalidade, os requisitos de segurança, políticas e considerações sobre o cumprimento dos serviços. Pode ser administrada por organizações ou por um provedor de serviços terceirizado, e pode existir tanto localmente quanto remotamente. Na Figura 2 tem-se um exemplo de nuvem comunitária composta por nuvens públicas;
- **Nuvem Híbrida:** considera a composição dos modelos de nuvens públicas e privadas. Essa composição permite que uma nuvem privada possa ter seus recursos ampliados a partir de uma reserva de recursos em uma nuvem pública. Essa característica possui a vantagem de manter os níveis de serviço mesmo que haja variações rápidas na necessidade dos recursos no atendimento das requisições.

Um fator a ser considerado, tanto na implementação das categorias de serviços apresentadas quanto na classificação das nuvens, é a virtualização. Em uma nuvem os recursos são virtualizados independente da localização ou do serviço fornecido.

2.3 Virtualização

A virtualização é uma técnica que possibilita a emulação de uma ou mais estações de trabalho/servidores em um único computador físico, que assume o papel de vários computadores lógicos, também conhecidos como máquinas virtuais. Cada máquina virtual oferece um ambiente completo similar a uma máquina física. Com isso, cada VM pode ter seu próprio sistema operacional, aplicativos e serviços de rede (CARISSIMI, 2008) (ALHAMAZANI *et al.*, 2014).

A virtualização é a tecnologia chave na computação em nuvem. Nesse contexto, ela refere-se principalmente à abstração dos recursos físicos de TI aos clientes e aplicativos que os usam. Além disso, permite que os servidores, dispositivos de armazenamento, *hardware* e outros recursos sejam tratados em conjunto ao invés de sistemas distintos, de modo que esse conjunto de recursos possa ser alocado por demanda (WEI; BLAKE, 2014). Sendo assim, a virtualização é adaptada a uma infraestrutura de nuvem dinâmica, pois oferece vantagens importantes no compartilhamento, gerenciamento e isolamento dos recursos (RIMAL; CHOI; LUMB, 2009).

Os principais objetivos e benefícios obtidos com o uso da virtualização são apresentados a seguir (MENKEN; BLOKDIJK, 2010):

- **Uso eficiente dos recursos:** com a melhoria na tecnologia, os recursos de *hardware* não são totalmente utilizados permanecendo ociosos na maior parte do tempo. Um dos objetivos da virtualização é mitigar esse problema, tornando a utilização dos recursos mais eficiente e reduzindo os custos operacionais e de gerenciamento;
- **Redução dos custos com gerenciamento de recursos:** a maioria das organizações lida com questões de espaço, energia, refrigeração, ou seja, toda a infraestrutura necessária para o seu funcionamento, o que despende muitos recursos e gera muitos gastos para a empresa. Utilizando uma infraestrutura virtualizada, as empresas podem economizar dinheiro e aproveitar melhor os recursos físicos disponíveis;
- **Maior flexibilidade e portabilidade:** o processo de ampliação das estações de trabalho e dos servidores é longo e custoso para as empresas, pois exige muito tempo na instalação e configuração das máquinas físicas, além da questão da ociosidade desses equipamentos. As máquinas virtuais podem ser facilmente instaladas e não despendem gastos adicionais com *hardware*, além de não requisitarem espaço físico extra. Além disso, a configuração das máquinas virtuais e o controle de acesso aos recursos tornam-se tarefas fáceis para os gerenciadores do sistema. Isso porque uma máquina virtual pode ser entendida como um contêiner de *software* que reúne ou encapsula um conjunto completo de recursos virtuais de *hardware*, um sistema operacional e todos os aplicativos dentro de um pacote de *software*. O encapsulamento torna as máquinas virtuais compactas e fáceis de gerenciar;

- **Isolamento:** embora as máquinas virtuais possam compartilhar os recursos físicos de um único computador, elas ficam isoladas dos outros *softwares* da máquina física, incluindo outras máquinas virtuais, como se fossem máquinas físicas separadas;
- **Eliminação de problemas com compatibilidade:** em sistemas operacionais que são instalados diretamente no *hardware*, problemas com incompatibilidade de versões e periféricos são muito comuns. Com a virtualização, vários sistemas operacionais podem ser instalados sobre uma camada de *software*, a qual faz uma interface com o *hardware* e elimina esses problemas de compatibilidade, sem que um sistema interfira nas configurações do outro.

Um ambiente virtualizado consiste de três partes básicas. A primeira consiste do sistema real, nativo ou hospedeiro (*host system*), que contém os recursos reais de *hardware* e *software* do sistema. A segunda, o sistema virtual, também denominado sistema convidado (*guest system*), que opera sobre o sistema real. Em alguns casos, vários sistemas virtuais podem coexistir, executando simultaneamente sobre o mesmo sistema real. Por fim, a camada de virtualização, conhecida como hipervisor ou monitor VMM (*Virtual Machine Monitor*), que constrói as interfaces virtuais a partir da interface real.

2.3.1 Hipervisor

O hipervisor, também conhecido como virtualizador, é um componente de *software* que hospeda as VMs e é responsável pela virtualização e controle dos recursos compartilhados pelas máquinas virtuais, tais como, processadores, dispositivos de entrada e saída e memória (ROSE, 2004) (CARISSIMI, 2008). Além disso, ele deve escalonar qual máquina virtual executará a cada momento, semelhante ao escalonador de processos do sistema operacional (MENASCÉ, 2005).

O VMM possui controle total sobre a CPU e executa instruções privilegiadas, enquanto que as máquinas virtuais são executadas em modo cliente, modo no qual as aplicações normalmente são executadas por meio de instruções simples. Assim, quando uma máquina virtual tenta executar uma instrução privilegiada, é gerada uma interrupção e o VMM se encarrega de emular a execução dessa instrução.

De acordo com Laureano e Maziero (2008), um hipervisor deve satisfazer as seguintes propriedades:

- **Equivalência:** um hipervisor provê um ambiente de execução quase idêntico ao da máquina real original. Dessa forma, todo programa executado em uma máquina virtual deve se comportar da mesma forma que o faria em uma máquina real. Exceções podem resultar oriundas de diferenças nos recursos disponíveis (como memória, disco, processador),

dependências de temporização e a existência dos dispositivos de entrada/saída necessários à aplicação;

- **Controle de recursos:** o hipervisor deve possuir o controle completo dos recursos da máquina real, sendo que nenhum programa executado na máquina virtual deve possuir acesso a recursos que não tenham sido alocados a ele pelo hipervisor, que deve intermediar todos os acessos. Além disso, a qualquer instante o hipervisor pode resgatar recursos previamente alocados;
- **Eficiência:** grande parte das instruções do processador virtual (processador provido pelo hipervisor) deve ser executada diretamente pelo processador da máquina real, sem intervenção do hipervisor. As instruções da máquina virtual que não puderem ser executadas pelo processador real devem ser interpretadas pelo hipervisor e traduzidas em ações equivalentes no processador real. Instruções simples, que não afetam outras máquinas virtuais ou aplicações, podem ser executadas diretamente no *hardware*.

Além dessas três propriedades básicas, propriedades derivadas são frequentemente associadas à hipervisores e podem ser utilizadas na segurança do sistema computacional (LAUREANO; MAZIERO, 2008):

- **Encapsulamento:** o encapsulamento torna as máquinas virtuais compactas e fáceis de gerenciar. Como o hipervisor tem acesso e controle sobre o estado interno de cada máquina virtual em execução, ele pode salvar *checkpoints* periodicamente ou em situações especiais, como por exemplo, antes de uma atualização de sistema operacional. Esses *checkpoints* são úteis para retornar a máquina virtual a estados anteriores (*rollback*) para análises em caso de falhas, ou para permitir a migração da máquina virtual entre hipervisores executando em computadores distintos;
- **Gerenciabilidade:** como cada máquina virtual é uma entidade independente das demais, a administração de diversas instâncias sobre um mesmo hipervisor é simplificada e centralizada. O hipervisor deve ter mecanismos para gerenciar o uso dos recursos existentes entre os sistemas convidados;
- **Inspeção:** o hipervisor tem acesso e controle sobre todas as informações do estado interno da máquina virtual, como registradores do processador, conteúdo de memória, eventos, entre outras;
- **Isolamento:** garante que um *software* em execução em uma máquina virtual não possa influenciar ou modificar outro *software* em execução no hipervisor ou em outra instância. Esta propriedade pode ser utilizada para garantir que erros de *software* ou aplicações maliciosas possam ser contidos em um ambiente controlado, sem afetar outras partes do sistema;

- **Recursividade:** alguns sistemas de máquinas virtuais exibem também esta propriedade, na qual deve ser possível executar um hipervisor dentro de uma VM, produzindo um novo nível de máquinas virtuais.

Diferentes abordagens consideradas na construção de hipervisores implicam na definição de estratégias para a virtualização, sendo que as mais utilizadas são a Virtualização Total (*Full Virtualization*) e a Paravirtualização (*Paravirtualization*). Essas estratégias são detalhadas na próxima Seção.

2.3.2 Estratégias de Virtualização

Na virtualização total tem-se uma réplica do *hardware* e o sistema operacional, sem alterações, executa sobre esse *hardware*. Por outro lado, na paravirtualização o sistema operacional é modificado para interceptar as chamadas ao sistema e executá-las na máquina física. A utilização de uma técnica ou outra normalmente fica a cargo do domínio das máquinas virtuais e do sistema que será implementado nessas máquinas (LI; LI; JIANG, 2010) (REDDY; RAJAMANI, 2014).

A virtualização total provê uma réplica do *hardware* subjacente, de forma que o sistema operacional e as aplicações possam ser executadas como se fossem executadas diretamente sobre o *hardware* (CARISSIMI, 2008). Dessa forma, o sistema operacional hóspede é instalado, sem modificações, sobre o VMM. A utilização do hipervisor sem necessidade de alteração do sistema operacional é a grande vantagem dessa estratégia. Essa é a abordagem utilizada na maioria dos hipervisores de sistemas clássicos, como QEMU, VMWare e KVM (*Kernel-based Virtual Machine*) (ZHOU; JIANG, 2014).

A desvantagem da virtualização total é que o sistema convidado executa mais lentamente, uma vez que todos os acessos ao *hardware* são intermediados pelo hipervisor (LI; LI; JIANG, 2010). Dessa forma, o hipervisor terá que interceptar e emular todas as instruções sensíveis (instrução que altera o estado do sistema) executadas pelos sistemas convidados, o que gera um custo elevado em plataformas de *hardware* sem suporte adequado à virtualização.

Além disso, alguns problemas técnicos gerados pela forma que os sistemas operacionais são implementados devem ser contornados, já que esses foram implementados para serem executados como uma instância única em uma máquina física, não disputando recursos com outros sistemas operacionais. Por exemplo, um sistema operacional convencional implementa memória virtual por meio de paginação. Todo o procedimento de gerência de alocação, liberação e controle de acesso às páginas deve ser respeitado. Para isso, o espaço de endereçamento do sistema hóspede deve ser convertido para um real gerando uma disputa de recursos, o que acarreta em queda de desempenho (MATTOS, 2008).

Na paravirtualização o sistema operacional é modificado, de forma que a chamada de uma instrução sensível é substituída pela chamada a um tratador de interrupção de *software*

(*Virtual Monitor Calls*) o que permite um melhor acoplamento entre os sistemas convidados e o hipervisor (CARISSIMI, 2008). Com isso o VMM não precisa testar instrução por instrução, o que representa um ganho significativo de desempenho das máquinas virtuais. Outro ponto positivo da paravirtualização é que os dispositivos de *hardware* são acessados por *drivers* da própria máquina virtual, não sendo necessário o uso de *drivers* genéricos que inibem o uso da capacidade total do dispositivo (LI; LI; JIANG, 2010). Os primeiros hipervisores a adotar a paravirtualização foram o Denali (WHITAKER; SHAW; GRIBBLE, 2002) e o Xen (BARHAM *et al.*, 2003).

Embora exija que o sistema convidado seja adaptado ao hipervisor, o que diminui sua portabilidade, a paravirtualização permite que o sistema convidado acesse alguns recursos do *hardware* diretamente, sem a intermediação ativa do hipervisor. Nesses casos, o acesso ao *hardware* é apenas monitorado pelo hipervisor, que informa ao sistema convidado seus limites, como as áreas de memória e de disco disponíveis. Para o acesso aos demais dispositivos como *mouse* e teclado, o hipervisor apenas gerencia a ordem de acesso no caso de múltiplos sistemas convidados em execução simultânea.

A computação em nuvem utiliza a virtualização como parte fundamental para alcançar os seus objetivos de negócios onde os provedores utilizam um hipervisor para virtualizar as máquinas que são fornecidas aos seus clientes (RIMAL; CHOI; LUMB, 2009). Dessa forma, o gerenciamento de máquinas virtuais, assim como de outros recursos que compõem uma nuvem, exige mecanismos eficientes de forma que os recursos físicos possam ser utilizados de forma eficiente e que as variadas requisições possam ser atendidas sob demanda (MANVI; SHYAM, 2014).

Dessa forma, no Capítulo 5 é proposto um módulo de gerenciamento de recursos que foi utilizado neste projeto para a alteração das configurações dos recursos providos a um cliente, com a finalidade de garantir o contrato firmado entre cliente e provedor e o uso eficiente desses recursos. Esse gerenciamento está diretamente relacionado com a qualidade de serviço que é oferecida por um provedor. Por essa razão, na próxima Seção é abordada a importância de se prover um serviço com qualidade para que uma boa relação entre provedores e clientes seja estabelecida.

2.4 QoS em Computação em Nuvem

Qualidade de Serviço (QoS) em nuvem corresponde à percepção do cliente e/ou do provedor em relação à eficiência que um serviço é prestado. Seu objetivo é fornecer algum tipo de garantia sobre o serviço como, por exemplo, mínimo de perdas e de atrasos na recepção de dados pelos clientes, bom desempenho, menor tempo de resposta, integridade e consistência dos dados (VEGESNA, 2001) (LIN; CHEN; LIN, 2014).

QoS pode ser definida em termos absolutos ou relativos. Por exemplo, em termos absolutos pode-se especificar que a perda de pacotes não excederá 3% ou nenhum pacote atrasará mais que 100 milissegundos. Já em termos relativos garante-se apenas que a classe mais prioritária de clientes receberá um serviço melhor (ou no mínimo não pior) que qualquer classe inferior (VASILOU, 2000) (PEIXOTO, 2012).

Segundo Liu *et al.* (2013), uma das formas de manter os níveis de QoS é estabelecer acordos, como um contrato de serviço (SLA) entre clientes e provedores. Nesses acordos ocorrem a escolha e o mapeamento dos melhores atributos de QoS, o que não é uma atividade simples, pois depende dos objetivos dos clientes, assim como da capacidade de fornecer qualidade do provedor.

Dessa forma, garantir a qualidade do serviço não é uma tarefa trivial, especialmente no contexto da nuvem, devido à heterogeneidade dos clientes e do ambiente da *Web* que possui ainda uma natureza imprevisível (ARDAGNA *et al.*, 2014). A nuvem precisa ser monitorada, avaliada e adaptada continuamente a fim de assegurar o nível adequado de QoS (STANTCHEV; SCHRÖPFER, 2009). Para isso, o sistema responsável pelo gerenciamento deve incluir requisitos como: mapeamento e alocação de recursos, negociação e monitoramento de QoS, e estabelecimento de um SLA (GUO *et al.*, 2007) (KRISHNA, 2013).

Outro fator importante é que não há uma especificação padrão disponível para prover QoS em um ambiente de nuvem, pois há diversos tipos de provedores, cada um com um modo de prestar seus serviços (CHARD, 2011). Portanto, uma vez que os clientes e os provedores podem possuir objetivos diferentes com relação a um determinado serviço, a correta identificação dos atributos de QoS oferece um dos elementos necessários para atingir a adequada relação de custo/benefício (CHANG; WALTERS; WILLS, 2013) (ZAMAN; GROSU, 2013).

A qualidade de serviço de um provedor está diretamente relacionada à capacidade de tolerância a falhas do seu sistema. O objetivo de tolerância a falhas é alcançar dependabilidade, a qual indica a qualidade do serviço fornecido por um dado sistema e a confiança depositada no serviço fornecido. Segundo Weber (2002), há diversos atributos de dependabilidade que podem ser considerados, dentre os quais podem-se citar:

- **Confiabilidade:** capacidade de atender as solicitações de serviços dentro das condições definidas, durante certo período de funcionamento e condicionado a estar operacional no início do período;
- **Disponibilidade:** está relacionada à probabilidade de um sistema estar em funcionamento em um determinado momento, considerando as pausas necessárias para a execução de reparos;
- **Segurança de funcionamento:** está relacionada à probabilidade de um serviço que está funcionando corretamente parar de funcionar sem aviso prévio. Nesse caso é necessário

parar a operação para evitar possíveis danos;

- **Segurança:** este atributo considera problemas como a falta de proteção contra acessos maliciosos, fraudes, erros, etc. Questões de segurança como, por exemplo, autenticidade, privacidade, não repúdio e integridade dos dados devem ser garantidas.

Um dos grandes desafios em QoS é descobrir quais são os provedores que realmente podem satisfazer as exigências feitas pelos clientes, dada a diversidade de ofertas de serviços em nuvem. Muitas vezes há diferenças entre as exigências feitas por um mesmo cliente a provedores distintos, ou no modo que um provedor presta um mesmo serviço para diferentes perfis de clientes. Isso faz com que seja muito difícil avaliar os níveis de serviços de diferentes provedores de nuvem de uma forma objetiva de tal modo que, a qualidade requerida, confiabilidade e segurança de um serviço possam ser asseguradas. Portanto, não é suficiente apenas descobrir serviços em múltiplos provedores, mas também avaliar qual é a nuvem mais adequada.

Neste contexto, o CSMIC (*Cloud Service Measurement Index Consortium*) tem trabalhado para identificar um conjunto de atributos que compõem um índice chamado SMI (*Service Measurement Index*). Esse índice oferece uma análise comparativa dos serviços de diferentes nuvens (GARG; VERSTEEG; BUYYA, 2011) (GARG; VERSTEEG; BUYYA, 2012). Os atributos SMI são projetados com base na Organização Internacional para Padronização (ISO – *International Organization for Standardization*). O índice SMI proporciona uma visão holística de QoS necessária pelos clientes para a seleção de um provedor de serviços em nuvem. Esses atributos são descritos a seguir (GARG; VERSTEEG; BUYYA, 2011) (GARG; VERSTEEG; BUYYA, 2012):

- **Responsabilidade:** esse grupo de métricas de QoS é usado para medir várias características específicas dos provedores. Entre essas métricas estão: auditabilidade, cumprimento/conformidade com o contrato, direitos da propriedade dos dados, ética e sustentabilidade. A responsabilidade é importante para conquistar a confiança de um cliente em qualquer provedor. Nenhum cliente deseja implantar seus serviços e armazenar seus dados críticos em um lugar onde não há responsabilidade com a segurança, com a exposição dos dados e com o cumprimento dos SLAs;
- **Agilidade:** uma das mais importantes vantagens da computação em nuvem é que ela contribui para a agilidade de uma organização. A organização pode se expandir e mudar rapidamente a um custo reduzido. No SMI, a agilidade é medida como uma taxa de variação, mostrando quão rapidamente novas capacidades são integradas na TI conforme a necessidade do cliente. Ao considerar a agilidade de um serviço na nuvem, os clientes querem analisar se o serviço é elástico, portátil, adaptável e flexível;
- **Custo:** o primeiro questionamento de um cliente antes de adotar a computação em nuvem é se é rentável ou não. Portanto, o custo é claramente um dos atributos vitais para a TI e os

negócios. Ele tende a ser a métrica mais quantificável, mas é importante expressar o custo das características que são relevantes para um cliente específico, visto que cada cliente possui características diferentes;

- **Desempenho:** há muitas soluções diferentes oferecidas por provedores de nuvem que abordam as necessidades de TI de diferentes clientes. Cada solução tem um desempenho diferente em termos de funcionalidade, de tempo de resposta e precisão. Os clientes precisam entender como as suas aplicações irão se comportar nas diferentes nuvens e se essas implantações satisfazem as suas expectativas;
- **Cumprimento do serviço:** essa característica indica a probabilidade de um serviço da nuvem ser cumprido dentro do prazo ou como prometido no contrato SLA. Toda organização visa expandir seus negócios e oferecer melhores serviços aos seus clientes, seja ela um provedor de serviços ou um cliente que contrata um determinado serviço da nuvem e presta outros serviços para outros usuários. Portanto, a tolerância a falhas, confiabilidade e estabilidade do serviço são fatores importantes na escolha de serviços em nuvem.
- **Segurança e privacidade:** por incluírem muitos atributos como confidencialidade, integridade e disponibilidade dos dados, a proteção e a privacidade dos dados são preocupações importantes em todas as organizações. O armazenamento de dados sob controle de outra organização é sempre uma questão crítica que requer políticas rigorosas de segurança empregadas pelos provedores de nuvem. Por exemplo, as organizações financeiras em geral exigem conformidade com os regulamentos que envolvem a integridade e privacidade dos dados;
- **Usabilidade:** para a adoção rápida de serviços em nuvem, a usabilidade desempenha um papel importante. Quanto mais fácil de usar e de aprender é um serviço, maiores são as chances de um cliente contratá-lo. A usabilidade de um serviço pode depender de vários fatores, tais como acessibilidade, instalabilidade, capacidade de aprendizado e operabilidade.

Outra característica que torna a garantia de QoS mais complexa e que merece destaque é a prestação de serviço sob demanda. Por exemplo, em determinado momento o cliente pode requisitar uma maior capacidade de processamento para executar a sua aplicação. Nesse caso, a nuvem deverá fornecer mais recursos de processamento por meio da alocação dinâmica. No entanto, é necessário definir quantos recursos são necessários para atender essa solicitação de aumento de processamento sem comprometer o nível de QoS estabelecido no SLA. Para isso, é necessário realizar um monitoramento automático da QoS. Com isso, aplica-se a Computação Autônoma (capacidade do sistema se autogerenciar) em nuvem ([HASSAN; AL-JUMEILY; HUSSAIN, 2009](#)). Porém, o estabelecimento de SLAs dinâmicos é uma tarefa complexa e depende do modelo de negócio implantado.

2.5 Modelos de Negócio em Computação em Nuvem

O conceito de manufaturamento em computação em nuvem está emergindo como um novo modelo de negócio que está remodelando a forma de negociar em sistemas orientados a serviços (REN *et al.*, 2015). Em um ambiente de computação em nuvem, existem os provedores (donos de recursos) e os consumidores (usuários/clientes de recursos), e ambos possuem seus próprios objetivos.

Quando um cliente deseja contratar um recurso da nuvem, normalmente, no mínimo três parâmetros devem ser especificados: as características do recurso a ser utilizado, o preço máximo a ser pago, e o prazo para execução do pedido (WEINGÄRTNER; BRÄSCHER; WESTPHALL, 2015). Essas informações são utilizadas para localizar e selecionar os recursos que possam atender os critérios especificados e para definir o contrato SLA. Dessa forma, modelos de negócio são implementados com os mais variados objetivos. As abordagens mais comuns para o gerenciamento de um ambiente complexo como o de uma nuvem podem utilizar políticas orientadas ao sistema (provedor) ou orientadas ao cliente (BUYYA; ABRAMSON; VENUGOPAL, 2005).

As abordagens orientadas ao sistema visam otimizar o desempenho do sistema e são comumente usadas em gerenciamento de recursos em um único domínio administrativo. Elas adotam uma estratégia convencional, cujo objetivo é melhorar o desempenho do sistema, por exemplo, *throughput*, utilização e taxa de execução, na qual o escalonamento decide quais serviços são executados em qual configuração de recurso (KARUNAKARAN; KRISHNASWAMY; SUNDARRAJ, 2014).

As abordagens orientadas ao cliente, por outro lado, concentram esforços para as questões que envolvem os clientes e contribuem com o cumprimento de requisitos de QoS. Um exemplo dessa abordagem é a garantia de certos níveis de desempenho baseados em atributos de clientes, tais como o *deadline* e o custo. Isso significa que os clientes podem negociar um preço particular para processamento de sua requisição baseado na demanda, no valor absoluto, na prioridade, ou no orçamento previsto, culminando em um modelo baseado em economia (BUYYA *et al.*, 2003).

As características de um escalonamento baseado em economia levam em consideração a dinamicidade do ambiente, e são orientadas e direcionadas especificamente aos clientes finais. O modelo baseado em economia preocupa-se com os serviços prestados a um cliente final, visando garantir que eles paguem pelo acesso a um recurso de acordo com o valor negociado. As políticas baseadas em preço funcionam de acordo com a demanda dos clientes e as ofertas dos recursos, o que leva a uma competição nesse modelo. Assim, um cliente compete com outros clientes, e os proprietários de recursos competem com outros proprietários de recursos estabelecendo um mecanismo dinâmico de cobrança em nuvem baseado, por exemplo, em leilão (JAVED *et al.*, 2015).

Modelos baseados em economia competitiva fornecem políticas/algoritmos e ferramentas para o compartilhamento de recursos em sistemas de computação em nuvem. Esses modelos podem ser baseados em escambo ou por preço. No modelo baseado em escambo, todos os participantes devem possuir um recurso para trocá-lo por outro, por exemplo, trocar uma unidade de armazenamento de dados por ciclos de processamento. Por outro lado, no modelo baseado em preço, todos os recursos têm um preço, que é baseado na demanda, oferta, valor, e a riqueza em um sistema econômico. Adicionalmente, pode-se assegurar a estabilidade, ou seja, o equilíbrio financeiro da nuvem (BERMAN; FOX; HEY, 2003).

Existem na literatura diversos trabalhos que propõem e avaliam modelos de negócio para computação em nuvem. Inicialmente têm-se os trabalhos que apresentam estudos e análises realizadas sobre essa linha de pesquisa.

Alhamad, Dillon e Chang (2011) analisam os desafios relacionados aos conceitos de confiança, gerenciamento de SLA e computação em nuvem. Os autores se concentram na definição de SLA para apresentar uma estrutura clara de negociação para clientes de nuvem e melhorar a forma de construção da relação de confiança entre o provedor de serviço e o cliente. Eles discorrem sobre os SLAs existentes em diferentes domínios, tais como serviços *Web* e grades computacionais, e sobre as vantagens e limitações dos modelos de medição de desempenho em SOA (*Service-Oriented Architecture*), sistemas distribuídos, computação em grade e serviços de nuvem.

Samimi e Patel (2011) fazem uma análise comparativa entre grades/nuvens econômicas e os modelos de preços disponíveis na literatura, a partir da qual as tarifas e os modelos de cobrança podem ser escolhidos para atender as negociações entre provedores e clientes. A escolha do modelo de negócio apropriado é muito importante e complexa, pois depende de muitos fatores, tais como os regulamentos de empresas, leis fiscais, SLAs e retorno sobre investimentos. Dessa forma, os autores apresentam os princípios fundamentais básicos e uma avaliação comparativa dos modelos econômicos recentes e dos preços aplicáveis à computação em grade/nuvem, a fim de propor modelos melhores.

Um modelo de negócio bastante conhecido e utilizado é o *pay-as-you-go*. Nele, os clientes pagam pelos recursos que utilizam. No entanto, muitas vezes os preços cobrados por esse modelo incluem momentos em que o cliente não utilizou o recurso contratado.

Em Ibrahim, He e Jin (2011) são demonstradas variações significativas dos custos cobrados, indicando injustiça entre o que é pago e o que realmente é utilizado. Estudos adicionais revelaram que o motivo de tais variações é a interferência entre as máquinas virtuais que executam simultaneamente em uma mesma máquina física. O custo adicional gerado pela interferência depende de vários fatores, incluindo as características de carga de trabalho, o número de VMs simultâneas e as políticas de agendamento de serviços na nuvem. Dessa forma, os autores adotaram o conceito de preços justos. Para resolver a injustiça causada pela interferência, eles propuseram um modelo de negócio chamado *pay-as-you-consume*, que cobra dos clientes aquilo

que eles realmente utilizaram excluindo interferências de outras VMs. A ideia chave por trás desse modelo é um mecanismo de aprendizagem baseado no modelo de previsão do custo relativo de interferência. Os resultados preliminares utilizando o Xen demonstraram a precisão do modelo de predição e a justiça do método proposto.

Há também trabalhos voltados para a economia de energia. A redução do consumo de energia e do custo operacional de servidores tem gerado uma grande preocupação em todas as áreas. O conceito de Computação Verde (*Green Computing*) e a busca por tecnologias que aplicam essa ideologia crescem a cada dia. O trabalho apresentado por [Mani e Rao \(2011\)](#) mostra um modelo de negócio para a economia de energia e de custo operacional dos servidores. Eles consideram a grande variação dos custos de energia causada pela localização geográfica dos recursos e pelo tempo de uso. Dessa forma, eles exploram a natureza dinâmica dos preços de energia elétrica para tomar decisões de escalonamento que consideram os custos operacionais (como a energia utilizada, por exemplo), além das cargas impostas a cada provedor e a prioridade de cada solicitação.

Além da economia de energia, são encontrados trabalhos sobre a utilização de nuvens colaborativas. Em [Yang et al. \(2012\)](#) foi desenvolvido um modelo orientado a negócio para nuvens federadas no qual vários provedores de infraestrutura independentes podem cooperar entre si de forma transparente para fornecer uma infraestrutura de TI escalável e serviços de hospedagem que garantam QoS para ROIA (*Real-time Online Interactive Applications*).

Outro trabalho que utiliza o ambiente de nuvens federadas é apresentado em ([VILLEGAS et al., 2012](#)). Nele os autores apresentam uma abordagem inicial para o problema de nuvem federada, considerando um modelo de camadas onde a negociação é restrita a um conjunto de parâmetros definidos. Eles discorrem sobre os benefícios de dissociar as diferentes camadas – SaaS, Paas e IaaS – para que a execução de um aplicativo possa ser realizada por diferentes provedores. Além disso, eles explicam como as políticas dos usuários e dos provedores podem ser usadas nas negociações entre vários provedores que compõem uma nuvem federada ou nas negociações entre diferentes camadas de serviço de um mesmo provedor.

O trabalho apresentado por [Hwang et al. \(2011\)](#) faz uma analogia ao uso de entidades federadas para serviços de criptografia e descriptografia. Este estudo propõe um modelo de negócio para a computação em nuvem baseada no conceito de separar os serviços de criptografia e descriptografia utilizados no armazenamento de dados em diferentes provedores que compõem a nuvem federada. Isso dificultaria o acesso de usuários mal-intencionados às chaves de criptografia e descriptografia e aos dados armazenados.

Existem também *frameworks* para a negociação de SLAs e aplicação de modelos de negócio. Em ([MAHBUB; SPANOUDAKIS, 2010](#)) é apresentado um *framework*, chamado PROSDIN, que integra a negociação SLA proativa com a descoberta de serviços dinâmicos. A descoberta de serviços no PROSDIN se baseia em características de vários serviços, incluindo características estruturais, comportamentais e de QoS. O objetivo da negociação SLA proativa

no PROSDIN é garantir, por meio de acordos preestabelecidos, que um determinado serviço possa ser usado por uma aplicação de algum cliente a qualquer momento sem a necessidade de interromper a execução da aplicação para realizar a negociação. A descoberta do serviço é feita de acordo com as necessidades da aplicação e, em seguida, o SLA é negociado. Esse SLA só é ativado quando existe a necessidade de utilização do serviço negociado. Os autores argumentam que se os contratos forem renegociados, reativamente os clientes poderão sofrer perdas. Por outro lado, se a negociação for feita de forma proativa, a manutenção dos SLAs se torna mais ágil fazendo com que as perdas dos clientes sejam mínimas. Para essa negociação, os autores utilizaram a reputação dos serviços a serem contratados e as características de qualidade de serviço dos mesmos.

Dos trabalhos analisados não foram encontradas definições de modelos de negócio que consideram atributos de QoS relacionados à segurança e ao desempenho do serviço prestado. Por esse motivo, o trabalho apresentado nesta tese tem o objetivo de definir modelos de negócio para computação em nuvem que levem em consideração os atributos de QoS relacionados a desempenho e ao *overhead* imposto por mecanismos de segurança. Para isso, será analisado o impacto causado no desempenho de um serviço quando diferentes mecanismos de segurança são considerados, além do comportamento do desempenho na alteração da quantidade de recursos computacionais como uma tentativa de contrapor o *overhead* imposto pelos mecanismos de segurança. Esses mecanismos serão selecionados por meio de um estudo de técnicas e metodologias que visam garantir a integridade, disponibilidade e confidencialidade, abordando desafios referentes ao acesso, armazenamento e manipulação dos dados no provimento de um serviço por meio de máquinas virtuais. Dessa forma, o custo gerado pela alteração dos recursos computacionais também deve ser considerado.

2.6 Considerações Finais

A computação em nuvem pode ser considerada uma extensão natural de tecnologias como a virtualização, computação utilitária, distribuída e autônoma, que permitem o gerenciamento escalável de recursos computacionais para atender a demanda de serviços dos clientes (MANVI; SHYAM, 2014).

A utilização de máquinas virtuais tornou-se uma alternativa concreta para várias soluções domésticas e corporativas, na qual se pode utilizar os recursos computacionais de forma mais eficiente na prestação de serviços, e consequentemente, economizar gastos. Dessa forma, a qualidade com que o serviço é prestado está diretamente relacionada aos recursos alocados, sendo um fator importante e de destaque na comunidade científica e industrial.

Dos atributos de QoS mencionados neste Capítulo, a segurança de um ambiente de computação em nuvem é um fator de destaque neste projeto. É importante verificar quanto um

determinado mecanismo de segurança influencia na qualidade do serviço prestado em termos de desempenho, uma vez que, em um ambiente de computação em nuvem, os serviços são prestados sob demanda utilizando VMs e os contratos firmados devem ser cumpridos. Dessa forma, desempenho, segurança e custo estão diretamente relacionados e devem ser considerados nas definições dos modelos de negócio. Por isso, o próximo Capítulo apresenta toda a fundamentação teórica sobre segurança em computação em nuvem, bem como a importância da aplicação de contramedidas eficientes em ambientes em nuvem.

SEGURANÇA NA COMPUTAÇÃO EM NUVEM

3.1 Considerações Iniciais

Antigamente, as redes de computadores eram usadas principalmente por pesquisadores com a finalidade de enviar mensagens como as de correio eletrônico, e por funcionários de empresas para compartilhar impressoras e outros recursos. Mudanças fundamentais ocorreram no decorrer das últimas décadas. Nos últimos anos, milhões de usuários usam as redes para transações bancárias, compras, declaração de impostos e outras funcionalidades, tornando a segurança das redes um dos temas mais importantes e discutidos pelas áreas acadêmica e industrial e um tópico essencial em sistemas distribuídos (KUROSE; ROSS, 2013).

Problemas de segurança são causados intencionalmente por agentes maliciosos que tentam obter algum benefício de um sistema e/ou prejudicar alguém. Segundo Tanenbaum (2003), os problemas podem ser divididos nas seguintes áreas interligadas: sigilo (também chamado de confidencialidade), autenticação, não repúdio e controle de integridade, e podem ser evitados por meio da aplicação de serviços de segurança.

De acordo com a RFC 2828¹, um serviço de segurança deve prover um tipo específico de proteção aos recursos a fim de assegurar o funcionamento do sistema e evitar a ocorrência de violações. Dessa forma, ele deve garantir as seguintes características (ZISSIS; LEKKAS, 2012):

- Assegurar a disponibilidade e a integridade das informações e dos serviços prestados entre os sistemas participantes evitando a perda ou alteração de informações devido ao acesso não autorizado, falha de componentes ou outros erros;

¹ <http://tools.ietf.org/html/rfc2828>

- Fornecer o controle sobre o acesso aos serviços ou aos seus componentes apenas aos usuários autorizados;
- Autenticar a identidade das entidades envolvidas na comunicação e, se necessário, garantir o não-repúdio das operações;
- Sempre que necessário, fornecer segurança na interação com outros sistemas externos;
- Garantir o isolamento dos dados e processos em nível virtual da nuvem, de forma que nenhum cliente tenha acesso aos dados de outros clientes;
- Manter o mesmo nível de segurança ao adicionar ou remover recursos no nível físico.

Em computação em nuvem, quando uma determinada empresa migra toda a sua infraestrutura lógica para um provedor de serviços, ela não se preocupa com questões penosas e custosas como, por exemplo, gerenciamento de infraestrutura física e lógica, compra de *hardware* e *software*, refrigeração, espaço físico, treinamento de profissionais e ociosidade de recursos. Suas principais preocupações estão relacionadas à segurança de seus dados e operações, o tempo gasto na execução das operações e o valor cobrado pelo provedor no provimento de um determinado serviço ([CHAKRAVARTHY; KANNAN, 2014](#)).

Portanto, quando se trata de segurança, um amplo campo de pesquisas, desafios e soluções se abre. Por esse fato, este Capítulo apresenta os aspectos críticos de segurança em nuvem bem como mecanismos disponíveis na literatura que visam combatê-los. Sabe-se que as ameaças enfrentadas por um provedor de serviços e/ou por um cliente podem ser categorizadas com base nos objetivos e propósitos dos ataques. Portanto, o conhecimento funcional dessas categorias de ameaças pode ajudar a organizar uma estratégia de segurança para a aplicação de contramedidas eficientes.

3.2 Princípios para uma Nuvem Segura

Quando as empresas migram seus ambientes de computação com suas identidades, informações e infraestrutura para a nuvem, elas devem estar dispostas a se submeter a algum nível de controle. Dessa forma, os clientes devem ser capazes de confiar em sistemas de uma nuvem e em seus provedores. Essa relação de confiança é adquirida com o controle de acesso aos recursos, segurança de dados, cumprimento dos prazos e gerenciamento de eventos/operações ([VELEV; ZLATEVA, 2011](#)).

Segundo [Lekkas \(2003\)](#), o conceito de confiança na sociedade é construído por várias razões diferentes, com base em cálculos, em conhecimento ou em razões sociais. Em uma nuvem a confiança pode ser definida como a certeza do cliente de que o provedor é capaz de fornecer os serviços exigidos com precisão e sem falhas. Dessa forma, a confiança pode ser expressa

como a crença do cliente na integridade moral, na solidez do funcionamento, na experiência e no cumprimento de todos os regulamentos e leis por parte de um provedor de serviços. Além disso, a relação de confiança entre cliente e provedor pode ser construída por meio de mecanismos de segurança que visam eliminar todos os possíveis riscos de danos/perdas em um sistema ou leva-los a um mínimo absoluto. Para isso, as propriedades descritas a seguir devem ser garantidas (STALLINGS; VIEIRA, 2008) (CASOLA *et al.*, 2011) (ZISSIS; LEKKAS, 2012):

- **Confidencialidade:** garante que apenas as entidades autorizadas têm acesso aos dados protegidos. No entanto, quanto maior o número de participantes, dispositivos e aplicações envolvidas, maior é a ameaça de corromper os dados de um sistema, pois o número de pontos de acesso torna-se maior. Dessa forma, várias questões de segurança surgem com relação à multilocação (*multi-tenancy*) dos recursos, armazenamento dos dados, segurança de aplicativos e privacidade;
- **Privacidade:** a privacidade está relacionada ao modo que uma pessoa controla e divulga suas informações pessoais. As organizações que lidam com dados pessoais são obrigadas a obedecer ao regime jurídico de um determinado país que visa garantir a proteção à privacidade e à confidencialidade. A nuvem apresenta uma série de desafios legais em relação às questões de privacidade envolvidas no armazenamento de dados em diferentes locais, além do aumento do risco de confidencialidade e violações de privacidade. Em vez dos dados serem armazenados em servidores da empresa, os dados dos clientes são armazenados em servidores do provedor de serviço, que podem estar na Europa, Ásia ou em qualquer outro lugar. Dessa forma, vários conflitos relacionados às diferentes legislações podem surgir;
- **Multilocação:** refere-se à característica da nuvem de compartilhar recursos como memória, aplicativos, redes e dados entre vários clientes. Embora esses clientes estejam isolados a um nível virtual, o *hardware* não é separado. Dessa forma, os recursos compartilhados devem ser cuidadosamente controlados a fim de evitar vulnerabilidades na privacidade, integridade e confidencialidade dos dados, como por exemplo, a quebra de sigilo;
- **Integridade:** garante que a informação não poderá ser modificada, de forma acidental ou intencional, por entidades que não possuem autorização. O controle de admissão e os direitos a recursos específicos de uma empresa garantem que os dados e serviços não sejam violados, desviados ou roubados. Ao impedir o acesso não autorizado, as organizações (clientes ou provedores) podem alcançar uma maior confiança na integridade dos dados e do sistema;
- **Disponibilidade:** garante o acesso e utilização sob demanda de um recurso do sistema por uma entidade autorizada de acordo com especificações de desempenho. Os serviços prestados por uma nuvem apresentam uma forte dependência das infraestruturas de recursos e disponibilidade da rede em todos os momentos. Dessa forma, para que a relação de

confiança entre cliente e provedor seja mantida, é crucial que os recursos, dados e serviços prestados estejam disponíveis na maior parte do tempo, com o mínimo de falhas possível.

A fim de conscientizar as empresas e as pessoas interessadas em aderir à computação em nuvem dos riscos associados a essa decisão, algumas organizações de segurança e instituições acadêmicas disponibilizaram alguns resultados de suas pesquisas. Esses trabalhos são abordados na próxima Seção.

3.3 Aspectos Críticos de Segurança em Nuvem

O estudo apresentado por [Zissis e Lekkas \(2012\)](#) descreve os requisitos de segurança exigidos para combater as ameaças presentes nos diferentes níveis de serviço oferecidos em uma nuvem. Compreender os relacionamentos e dependências entre as diferentes camadas da computação em nuvem é fundamental para entender os riscos de segurança relacionados aos três principais modelos de serviço (SaaS, PaaS e IaaS).

SaaS provê o conjunto mais integrado de funcionalidades de computação em nuvem, construídas de acordo com as necessidades dos clientes, possuindo o mínimo de extensibilidade para eles, pois o serviço já está definido, e com um nível relativamente alto de segurança integrado. As principais preocupações de segurança estão relacionadas ao controle de acesso, privacidade, proteção e exposição dos dados e informações dos clientes.

PaaS destina-se a permitir que os desenvolvedores construam suas próprias aplicações sobre a plataforma. Dessa forma, é mais flexível que o modelo SaaS, pois oferece ao cliente funcionalidades pré-configuradas para que este possa escolher e usar. Como o conjunto de funcionalidades não é completo ou plenamente integrado, existe uma maior flexibilidade para se inserir camadas de segurança adicionais como de controle de acesso e mecanismos de gerenciamento, detecção e prevenção de ataques como os de invasão, modificação ou interrupção de um serviço.

IaaS provê poucas funcionalidades integradas de segurança além da proteção da própria infraestrutura virtualizada. Dessa forma, IaaS necessita que o sistema operacional, aplicações e conteúdo sejam gerenciados e protegidos pelo cliente da nuvem ([FERNANDES *et al.*, 2014](#)). Há também preocupações quanto à segurança dos *data centers* físicos como roubos, sabotagens e desastres naturais.

Provedores de IaaS oferecem para seus consumidores a ilusão de uma capacidade computacional ilimitada de processamento, largura de banda e armazenamento. O processo de registro para compra desse serviço é simples e pode ser feito por qualquer portador de um cartão de crédito válido. Como os registros podem ser anônimos por meio de *logins* que identificam um usuário do sistema e não entidades no mundo real, e o uso do serviço é imediato após a

contratação, usuários mal intencionados como *spammers* e programadores de *botnets*, chamados aqui de atacantes, são capazes de realizar atividades maliciosas com relativa impunidade (ZHAO *et al.*, 2009).

As soluções possíveis para mitigar os casos citados devem envolver, por exemplo, processos de registro e validação de usuários mais rigorosos, aprimoramento da monitoração de fraudes no uso de cartão de crédito e monitoração do tráfego de rede do cliente sem violar sua privacidade, além do monitoramento do tráfego de informações de origem e destino públicos.

Por exemplo, o sequestro de contas ou serviços é um problema comum, no qual ocorrem *phishing*, fraudes, explorações de vulnerabilidades de sistemas e aplicações. Uma vez obtidas as credenciais de seu alvo, um atacante pode acompanhar as atividades e transações efetuadas pela conta de acesso, gerando uma diversidade de problemas, como por exemplo, leitura, alteração e inserção de dados, redirecionamento de clientes para domínios falsos, subversão de instâncias de serviços legítimos, entre outros. O compartilhamento ou a delegação de credenciais entre entidades deve ser evitado ou bem monitorado. O sistema deve implantar técnicas robustas de autenticação, preferencialmente baseadas em vários fatores como a combinação de técnicas de impressão digital, senhas, monitoramento para detectar atividades não autorizadas ou suspeitas, como por exemplo, um sistema de detecção de intrusão (PROVOS; RAJAB; MAVROMMATIS, 2009).

Diferente das abordagens tradicionais de tecnologia da informação, a computação em nuvem oferece grande flexibilidade para os usuários visto que estes não precisam se preocupar com a complexidade de gerenciamento inerente a cada sistema (por exemplo, os bancos de dados podem ser transferidos para *data centers* de grandes empresas especializadas). Porém, o gerenciamento dos dados em ambientes terceirizados nem sempre é confiável. Os usuários ficam a mercê da disponibilidade e integridade provida pelos provedores de serviço de armazenamento como a *Amazon Simple Storage Service* (Amazon S3). Dessa forma, é necessária a utilização de modelos de armazenamento de dados seguros visando garantir a integridade dos dados dos clientes (WANG *et al.*, 2009).

Os dados de usuários e consumidores podem ser comprometidos de várias maneiras, por exemplo, informações que não possuem cópia de segurança podem ser eliminadas ou alteradas, os registros de um contexto podem ser desvinculados, o armazenamento pode ser feito em mídias não confiáveis ou a chave de codificação pode ser perdida. O risco de comprometimento de dados aumenta na nuvem devido ao grande número de desafios inerentes as características arquiteturais e operacionais desse ambiente, que envolvem desde a implementação de controles de autenticação, autorização, auditoria e cifragem, até falhas operacionais, problemas de jurisdição, confiabilidade do *data center*, entre outros.

Ambientes terceirizados ou externos como os presentes em computação em nuvem necessitam de práticas e mecanismos de segurança rigorosos para garantir a segurança dos dados em trânsito, proteção dos dados utilizados em processos, gerenciamento do ciclo de vida de

chaves, estabelecimento de regras contratuais com os provedores (exigindo a correta destruição de dados que estão armazenados em mídias antes desta ser liberada para uso) e estratégias para efetuar a cópia de segurança (FERNANDES *et al.*, 2014).

Preocupadas com essas ameaças, algumas organizações publicaram os seus estudos sobre os riscos na adesão e migração de infraestruturas empresariais para a nuvem. Em junho de 2008, a empresa de consultoria Gartner publicou um relatório intitulado *Assessing the Security Risks of Cloud Computing*, no qual eles identificaram algumas questões específicas de segurança que os clientes devem analisar antes de escolher um provedor (HEISER; NICOLETT, 2008). Segundo os autores, quando os dados de um cliente são processados em uma nuvem, o provedor de serviço deve garantir as cláusulas definidas no SLA, além de garantir o controle de acesso aos usuários credenciados, disponibilidade, suporte, conformidade com a jurisdição do país onde os dados são armazenados e recuperação desses dados em caso de falhas e desastres naturais. No entanto, muitos dos provedores disponíveis atualmente não garantem essas especificações.

Em novembro de 2009, a ENISA (*European Network and Information Security Agency*) publicou outro documento intitulado *Cloud Computing – Benefits, Risks and Recommendations for Information Security* (CATTEDDU, 2010). O documento provê uma visão de gerenciamento de risco em computação em nuvem e uma série de recomendações. Em dezembro de 2009, a CSA (*Cloud Security Alliance*)² lançou a versão 2.1 do seu documento *Security Guidance for Critical Areas of Focus in Cloud Computing*, no qual identificou várias áreas de risco em computação em nuvem (BRUNETTE; MOGULL, 2009). Logo depois, em março de 2010, a CSA publicou o documento *Top Threats to Cloud Computing v1.0* com os resultados de suas pesquisas sobre as principais ameaças para a computação em nuvem (ARCHER *et al.*, 2010). O objetivo é proteger os provedores de nuvem, bem como os seus potenciais clientes e ajudar a identificar os principais riscos na adesão de uma infraestrutura de nuvem.

Outros trabalhos como os apresentados em (SUBASHINI; KAVITHA, 2011), (LOMBARDI; PIETRO, 2011), (VAQUERO; RODERO-MERINO; MORÁN, 2011), (VELEV; ZLATEVA, 2011) e (SHAMSOLMOALI; ALAM, 2015) vão ao encontro com o conteúdo publicado pelas empresas mencionadas. Os autores apresentam vários aspectos críticos de segurança em um ambiente de nuvem, dos quais se destacam:

- **Segurança de Identidade:** o gerenciamento de identidade e os serviços de autenticação devem ser um elemento chave da segurança em nuvem, pois é por meio deles que as empresas definem quem tem acesso aos seus dados (FELICIANO *et al.*, 2011). Esses mecanismos podem ser implementados pelas próprias empresas contratantes ou pelo provedor de serviço. Por meio da segurança da identidade mantém-se a integridade e confidencialidade dos dados e das aplicações ao mesmo tempo em que tornam o acesso disponível aos clientes apropriados (VELEV; ZLATEVA, 2011). Alguns dos principais

² <https://cloudsecurityalliance.org/>

padrões e soluções utilizadas para a implantação de gerência de identidades são: SAML (*Security Assertions Markup Language*) (HUGHES; MALER, 2005), OpenID³, Shibboleth⁴, Higgins⁵, OpenAM⁶, XACML (*Extensible Access Control Markup Language*)⁷ e OAuth⁸.

- **Segurança da Informação:** controle de acesso físico, de acesso ao *hardware* e *software* e de identidade combinam-se para proteger os dados nos *data centers*. Na nuvem, a barreira de proteção da infraestrutura é difusa, pois envolve uma série de fatores. Assim, a segurança da informação exigirá (BRUNETTE; MOGULL, 2009):
 - **Isolamento dos dados:** na multilocação de recursos os dados do ambiente devem ser armazenados de forma segura, a fim de protegê-los quando vários clientes utilizam os recursos compartilhados. Em uma nuvem a intrusão de dados de um cliente por outro é possível. Um cliente malicioso pode inserir dados mascarados em uma aplicação, e essa aplicação quando executada, pode gerar danos ao sistema. Virtualização, criptografia e controle de acesso são mecanismos que permitem vários graus de isolamento de dados entre empresas e clientes diferentes.
 - **Segurança dos dados:** os provedores de serviços devem fornecer mecanismos de segurança para proteger os dados de seus clientes. Isso envolve o uso de técnicas de criptografia e controle de acesso aos dados. A Seção 3.5.2 descreve como os provedores aplicam os mecanismos de segurança.
 - **Segurança de rede:** em uma nuvem os dados são obtidos das empresas, processados e armazenados. Todo o fluxo de dados da rede deve estar seguro para evitar a perda e manipulação de informações. Para isso, podem-se utilizar técnicas de criptografia ou outras técnicas que garantam a segurança e o monitoramento do tráfego de rede.
 - **Integridade dos dados:** a integridade dos dados é um dos elementos mais críticos em qualquer sistema. Qualquer tipo de transação deve seguir as propriedades de ACID (Atomicidade, Consistência, Isolamento e Durabilidade).
 - **Vulnerabilidade na virtualização:** uma máquina virtual oferece um ambiente completo similar a uma máquina física. Dessa forma, algumas vulnerabilidades encontradas em uma máquina física são também encontradas nos *softwares* de virtualização. Essas vulnerabilidades podem ser utilizadas por atacantes para adquirirem determinados privilégios e violar outras restrições de segurança (SUBASHINI; KAVITHA, 2011) (VAQUERO; RODERO-MERINO; MORÁN, 2011).

³ <http://openid.net/get-an-openid/what-is-openid/>

⁴ <http://shibboleth.net/>

⁵ <http://www.eclipse.org/higgins/>

⁶ <http://www.forgerock.com/openam.html>

⁷ http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf

⁸ <http://hueniverse.com/oauth/guide/intro/>

- **Segurança de infraestrutura:** provedores de aplicativos IaaS tratam as solicitações de serviços dentro de uma instância virtual de um cliente como se fossem uma caixa preta. Por exemplo, quando uma empresa contrata um serviço de IaaS e presta serviços a outros clientes, o provedor de IaaS é indiferente quanto às operações e ao gerenciamento de pedidos das empresas contratantes do seu serviço. Assim, é importante que o contratante assuma total responsabilidade por assegurar a sua infraestrutura, implantando, por exemplo, mecanismos de controle de acesso, de criptografia dos dados e/ou ferramentas de monitoramento da rede (BRUNETTE; MOGULL, 2009) (MATHER; KUMARASWAMY; LATIF, 2009) (VELEV; ZLATEVA, 2011). No entanto, nada impede que alguns provedores disponibilizem mecanismos de segurança, como por exemplo, a Amazon que fornece um mecanismo de controle de acesso chamado *AWS Identity and Access Management*⁹ (IAM). Além disso, tem-se a segurança de toda a infraestrutura física pertencente a um provedor a fim de se evitar, por exemplo, o roubo de equipamentos, sabotagens e vazamento de informações sigilosas.

Embora alguns provedores forneçam mecanismos de segurança aos seus clientes, eles não apresentam o impacto gerado por esses mecanismos sobre o desempenho, e consequentemente, sobre o custo do serviço. O trabalho apresentado nesta tese visa em um primeiro momento analisar o comportamento do desempenho de um ambiente em nuvem na utilização de diferentes mecanismos de segurança. A partir dessa análise, modelos de negócio que relacionam segurança, desempenho e custo foram definidos. Dessa forma, dada a ampla gama de desafios relacionados à segurança em nuvem, os aspectos críticos que foram utilizados são definidos e apresentados em maiores detalhes no próximo Capítulo. Por meio desses aspectos críticos, mecanismos de segurança relacionados ao acesso, armazenamento e manipulação dos dados no provimento de um serviço por meio de máquinas virtuais foram selecionados para compor a análise mencionada. No entanto, antes de apresentá-los, uma descrição mais abrangente dos mecanismos de segurança disponíveis na literatura, bem como do estado da arte de trabalhos que os utilizam, é necessária.

3.4 Mecanismos de Segurança

Existem na literatura diversos mecanismos de segurança criados para combater as mais variadas ameaças presentes em um ambiente em nuvem. Embora a utilização da criptografia esteja diretamente relacionada com a confidencialidade dos dados, observa-se que ela também é essencial para o provimento de integridade, não repúdio e controle de acesso, o que faz da criptografia uma peça chave na segurança de sistemas computacionais (KUROSE; ROSS, 2013).

⁹ <http://aws.amazon.com/iam/>

3.4.1 Criptografia

Criptografia consiste em um conjunto de técnicas para converter um texto original, denominado texto aberto, em um texto ilegível (texto cifrado) por meio de algum algoritmo de criptografia parametrizado por uma chave. A criptografia é um dos principais métodos para proteger informações valiosas e o aspecto mais importante da segurança de comunicações (STALLINGS; VIEIRA, 2008).

Técnicas criptográficas permitem que um remetente disfarce os dados de modo que um atacante não consiga obter nenhuma informação que possa ser manipulada ou utilizada para danificar o sistema. No entanto, é necessário que o destinatário esteja habilitado a recuperar os dados originais a partir dos dados disfarçados. Para realizar essas operações o algoritmo deve ter como parâmetro a chave correta. Essa chave é formada por um conjunto de *bits* que determinam o seu tamanho (ABD *et al.*, 2014).

De acordo com o princípio de *Kerckhoff*, os algoritmos criptográficos devem ser públicos enquanto que as chaves devem ser secretas. Dessa forma, o sigilo das informações é assegurado desde que o tamanho da chave seja adequado, visto que quanto maior o seu comprimento, mais segura é a criptografia (TANENBAUM, 2003). De acordo com o tipo da chave, a criptografia pode ser classificada como Criptografia de Chave Simétrica ou Criptografia de Chave Pública.

3.4.1.1 Chave simétrica

Na criptografia simétrica ou privada, os processos de criptografia e descryptografia são realizados por uma mesma chave compartilhada entre o emissor e o receptor. Nessa criptografia o texto original é transformado em texto cifrado por meio da chave secreta e de um algoritmo de criptografia. Usando a mesma chave e um algoritmo de descryptografia, o texto original é recuperado a partir do texto cifrado. A Figura 3 mostra esse procedimento.

Figura 3 – Criptografia de chave simétrica, adaptado de (ROSENBERG; REMY, 2004)

A vantagem da criptografia de chave simétrica é que os seus algoritmos são rápidos e podem operar em diferentes tamanhos de mensagens. Por outro lado, a dificuldade em gerenciar

a distribuição da chave compartilhada é a sua principal desvantagem, visto que apenas os usuários autorizados devem possuir essa chave antes de trocarem as mensagens (MORENO; PEREIRA; CHIARAMONTE, 2005). A Tabela 1 mostra as chaves simétricas mais comuns segundo Tanenbaum (2003).

Tabela 1 – Algoritmos criptográficos de chave simétrica, adaptado de (TANENBAUM, 2003)

Cifra	Autor	Comprimento
Blowfish	Bruce Schneier	1 a 448 bits
DES	IBM	56 bits
IDEA	Massey e Xuejia	128 bits
RC4	Ronald Rivest	1 a 2048 bits
RC5	Ronald Rivest	128 a 256 bits
AES	Daemen e Rijmen	128 a 256 bits
Serpent	Anderson, Biham, Knudsen	128 a 256 bits
DES triplo	IBM	168 bits
Twofish	Bruce Schneier	128 a 256 bits

3.4.1.2 Chave pública

A criptografia de chave pública ou assimétrica é um método que utiliza um par de chaves, sendo uma chave pública e uma chave privada. A chave pública é distribuída livremente para todos os correspondentes, enquanto que a chave privada deve ser conhecida apenas pelo seu proprietário. Em um algoritmo de criptografia assimétrica, uma mensagem cifrada com a chave pública somente pode ser decifrada pela sua chave privada correspondente. A Figura 4 mostra esse procedimento.

Figura 4 – Criptografia de chave assimétrica, adaptado de (ROSENBERG; REMY, 2004)

O algoritmo mais utilizado para criptografia de chave pública é o RSA (*Rivest, Shamir, Adleman*) (KUROSE; ROSS, 2013). Esse algoritmo apesar de ser considerado forte, é lento devido à exigência por chaves de tamanho mínimo de 1024 bits a fim de garantir um nível

satisfatório de segurança semelhante ao nível de segurança dos algoritmos de chave simétrica de 128 *bits*. Por esse motivo, o uso do RSA é pouco viável para criptografar uma grande quantidade de dados (KOCHER *et al.*, 2004).

Há também o ECC (*Elliptic Curve Cryptosystem*), um método para gerar chaves pública e privada com base nas propriedades das curvas elípticas. As chaves são derivadas de um ramo diferente da matemática e, ao contrário do RSA, sua segurança não depende da dificuldade de decompor grandes números em fatores. As chaves obtidas são mais curtas, seguras e as demandas de processamento para cifrar e decifrar são menores do que para o RSA (COULOURIS; DOLLIMORE; KINDBERG, 2007). Dessa forma, os algoritmos de criptografia de curva elíptica podem ser utilizados em sistemas como aqueles que incorporam dispositivos móveis, os quais possuem recursos de processamento limitados (VIJAYALAKSHMI; RAJA, 2012).

3.4.2 Assinatura Digital e Entidades Intermediárias

A assinatura digital é um mecanismo de autenticação que permite que o criador de uma mensagem anexe um código que atue como uma assinatura (STALLINGS; VIEIRA, 2008). A utilização da assinatura digital providencia a prova inegável de que uma mensagem veio do emissor. Para verificar este requisito, as seguintes propriedades são necessárias (TANENBAUM, 2003):

- **Autenticidade:** o receptor deve poder confirmar que a assinatura foi feita pelo emissor;
- **Integridade:** qualquer alteração da mensagem faz com que a assinatura não corresponda mais ao documento;
- **Não repúdio:** o emissor não pode negar a autenticidade da mensagem.

A assinatura digital consiste na criação de um código, por meio da utilização de uma chave, de modo que a pessoa ou entidade que receber uma mensagem contendo este código possa verificar se o remetente é mesmo quem diz ser e identificar qualquer mensagem que possa ter sido modificada.

Para garantir a integridade de uma determinada mensagem ou a autenticidade de um determinado emissor podem-se utilizar entidades intermediárias. Em um ambiente de computação em nuvem, as entidades intermediárias tem um papel fundamental para permitir que organizações associadas se autenticuem. Para isso, as propriedades dos métodos de assinatura digital e das criptografias de chave simétrica e pública são combinadas para a geração de um certificado digital.

3.4.2.1 Assinaturas com chave simétrica

Na assinatura com chave simétrica é utilizado um intermediário conhecido como central de distribuição de chave (KDC – *Key Distribution Center*), no qual todos os envolvidos nas trocas de mensagens confiam (KUROSE; ROSS, 2013).

O usuário escolhe uma chave secreta e a leva para o KDC. Quando o emissor envia uma mensagem, esta mensagem passa pelo intermediário que verifica a sua origem, onde ele só aceitará a mensagem se ela for criptografada com a chave secreta daquele emissor, descriptografa e a envia ao receptor. O receptor recebe a mensagem com o texto simples e a mensagem assinada pelo KDC. Ou seja, o KDC gera um certificado digital que atesta a identidade do emissor da mensagem. Um exemplo de sistema projetado para isso é o Kerberos.

O Kerberos é um serviço de autenticação projetado para uso em um sistema distribuído, onde ele utiliza um serviço de autenticação de terceiros confiável, que permite que clientes e servidores estabeleçam uma comunicação autenticada (STALLINGS; VIEIRA, 2008).

3.4.2.2 Assinaturas com chave assimétrica

Na criptografia de chave assimétrica é utilizada a autoridade certificadora (CA – *Certification Authority*) que possui a responsabilidade de verificar se determinada chave pertence a uma entidade. Se a chave pública for certificada e se a autoridade certificadora que a certificou for confiável, então a veracidade sobre a quem essa chave pertence é garantida.

Um exemplo é o serviço de autenticação X.509 que define uma estrutura para a provisão de serviços de autenticação pelo diretório X.500 aos seus usuários. Esse diretório pode ser usado como um repositório de certificados de chave pública, no qual cada certificado contém a chave pública de um usuário e é assinado com a chave privada de uma autoridade certificadora confiável (STALLINGS; VIEIRA, 2008). O OpenLDAP (*Open Lightweight Directory Access Protocol*) é um exemplo de serviço de diretório com as características descritas.

No entanto, fazer uma única CA emitir todos os certificados do mundo não funcionaria, pois ela seria um ponto central de falha, além de ficar sobrecarregada. Por essas razões foi desenvolvida a PKI (*Public Key Infrastructure*).

Uma PKI possui vários componentes, como usuários, autoridades certificadoras, certificados e diretórios e é um órgão que tem como objetivo manter uma estrutura de emissão de chaves públicas baseada no princípio da terceira parte confiável, oferecendo uma mediação de credibilidade e confiança em transações entre partes que utilizam certificados digitais. A sua principal função é definir um conjunto de técnicas, práticas e procedimentos a serem adotados pelas entidades a fim de estabelecer um sistema de certificação digital baseado em chave pública (TANENBAUM, 2003).

3.4.2.3 Sumário de mensagens

Apesar da utilização de uma entidade intermediária ser uma estratégia válida para a validação de identidades, distribuição de chaves e emissão de certificados, a exigência de que todos os envolvidos confiem nela pode ser um fator crítico, visto que atacantes podem utilizar desse artifício para interceptar e manipular dados. Dessa forma, pode ser interessante que o ato de assinatura de documentos não exija a presença de uma autoridade intermediária.

A assinatura digital com criptografia assimétrica traz uma solução para esse problema. Devido à lentidão dos algoritmos assimétricos, os métodos de assinatura digital assinam um sumário da mensagem, também conhecido como resumo da mensagem, ao invés da mensagem toda. Esse sumário é obtido por meio do processamento da mensagem em uma função *hash*, a qual extrai um trecho qualquer do texto e, a partir dele, gera uma cadeia de *bits* (MORENO; PEREIRA; CHIARAMONTE, 2005). Dessa forma, tem-se a combinação da criptografia assimétrica com a função *hash* na geração da assinatura digital. Essa combinação é apresentada na Figura 5.

Figura 5 – Assinatura digital utilizando criptografia de chave pública e função *hash*, adaptado de (ROSENBERG; REMY, 2004)

Inicialmente aplica-se uma função *hash*, como por exemplo, o SHA-1 (*Secure Hash Algorithm 1*), sobre o texto original gerando um sumário de tamanho fixo. Em seguida, ocorre a criptografia do sumário utilizando a chave privada do emissor e o envio do documento original juntamente com o sumário criptografado para o receptor. O receptor recebe o documento original e o sumário criptografado, aplica a mesma função *hash* sobre o documento original e compara *bit a bit* o resultado obtido com o sumário descriptografado a partir da chave pública do emissor. Caso eles sejam exatamente iguais, a assinatura é considerada válida (Figura 6).

Figura 6 – Verificação da assinatura digital, adaptado de (ROSENBERG; REMY, 2004)

Para atingir o objetivo especificado, é necessário que a função *hash* tenha as seguintes características (BURNETT; PAINE, 2001):

- Ser eficiente e rápida para gerar o sumário da mensagem;
- Não permitir que o documento original seja gerado a partir do sumário da mensagem;
- Não gerar o mesmo sumário a partir de dois documentos originais distintos.

Os algoritmos de função *hash* mais utilizados são o MD5 (*Message Digest Algorithm* 5) e o SHA-1. O MD5 é um algoritmo de *hash* de 128 *bits* unidirecional desenvolvido pela *RSA Data Security*, descrito na RFC 1321, e muito utilizado por *softwares* com o protocolo P2P (*Peer-to-Peer*) na verificação de integridade de arquivos e *logins*. Foi desenvolvido em 1991 por Ronald Rivest para suceder ao MD4 que tinha alguns problemas de segurança. Por ser um algoritmo unidirecional, uma *hash* MD5 não pode ser transformada novamente no texto que lhe deu origem. Dessa forma, o método de verificação é feito pela comparação das duas *hash*, uma da mensagem original confiável e outra da mensagem recebida (TANENBAUM, 2003).

O SHA-1 é um algoritmo que produz um sumário de 160 *bits*. Ele é baseado no MD4, que é semelhante ao MD5, com algumas operações adicionais. Ele é significativamente mais lento do que o MD5, mas o sumário de 160 *bits* apresenta uma segurança maior contra ataques

estilo força bruta. Podem-se gerar sumários maiores como os de 224, 256 e 512 *bits*, porém o seu comprimento adicional acarreta em maiores custos para a geração, armazenamento e comunicação de assinaturas digitais (COULOURIS; DOLLIMORE; KINDBERG, 2007).

3.4.3 Outros Mecanismos

Além de todas as ferramentas criptográficas descritas, existem outras técnicas que podem ser usadas para proteger a integridade dos dados e o tráfego de rede como *firewalls*, sistemas de detecção e prevenção de intrusão, ferramentas de gerenciamento e monitoramento de redes (como CACTI¹⁰, Nagios¹¹, ZenOSS¹² e OpManager¹³), VPNs (*Virtual Private Networks*), protocolos como o IPsec (*IP Security Protocol*) (camada de rede), SSL (*Secure Sockets Layer*) e TLS (*Transport Layer Security*) (camada de transporte), HTTPS (*Hyper Text Transfer Protocol Secure*) (camada de aplicação), entre outros (TANENBAUM, 2003) (STALLINGS; VIEIRA, 2008) (KUROSE; ROSS, 2013).

3.4.3.1 Firewall

Um *firewall* é uma combinação de *hardware* e *software* que isola a rede interna de uma organização da Internet em geral, controlando o tráfego de entrada e saída da rede. Comumente usados para a prevenção de ataques, os *firewalls* são a primeira linha de defesa, mas não devem ser a única. Ele pode ser classificado em duas categorias: filtros de pacote e *gateways* de aplicação (TANENBAUM, 2003).

Filtros de pacote tomam decisões baseadas em regras específicas definidas pelo administrador da rede como porta, endereço de origem/destino, estado da conexão e outros parâmetros do pacote. Como exemplo pode-se citar o *iptables* do Linux.

Um *gateway* de aplicação é um servidor específico de aplicação através do qual todos os dados da aplicação devem passar. Ele analisa o conteúdo do pacote para tomar suas decisões de filtragem, e dessa forma, é mais intrusivo, permitindo um controle relacionado com o conteúdo do tráfego. No entanto, para cada aplicação diferente é necessário um *gateway* de aplicação diferente. O *software* cliente deve saber entrar em contato com o *gateway* e deve saber como dizer ao *gateway* a qual servidor se conectar. Além disso, o desempenho é comprometido visto que todos os dados passam pelo *gateway*.

¹⁰ <http://www.cacti.net/>

¹¹ <http://www.nagios.org/>

¹² <http://www.zenoss.com/>

¹³ <http://www.manageengine.com/network-monitoring/>

3.4.3.2 Detecção e prevenção de intrusão

Além do *firewall*, há outros dispositivos que podem ser utilizados na detecção e prevenção de ataques. Um sistema de detecção de intrusão (*Intrusion Detection System – IDS*) é composto por um *hardware* ou *software* que monitora um sistema ou uma rede contra atividades não autorizadas. Suas funções incluem monitorar, detectar a presença de atividade não autorizada e gerar alertas.

Há também o sistema de prevenção de intrusão (*Intrusion Prevention System – IPS*). O IPS é composto por um *hardware*, *software* ou combinação de ambos e é responsável por realizar alguma ação quando o IDS detecta alguma atividade não autorizada e gera um alerta.

Esses sistemas podem ser utilizados para detectar uma série de tipos de ataques incluindo mapeamento de rede, escaneamento de portas, escaneamento de pilha TCP, ataques de inundação de banda larga, *worms* e vírus, ataques à vulnerabilidades de sistemas operacionais e de aplicações. Eles podem ser classificados em sistemas baseados em assinatura ou sistemas baseados em anomalias (KUROSE; ROSS, 2013).

Um sistema baseado em assinatura mantém um banco de dados extenso com assinaturas que podem prejudicar o sistema. Cada pacote é analisado comparando as assinaturas. Se elas forem compatíveis, um alerta é gerado. No entanto, esse sistema exige um conhecimento prévio do ataque para gerar uma assinatura. Outro fator importante é, pelo fato de cada pacote ser comparado com um amplo conjunto de assinaturas, o IDS fica ocupado com o processamento das comparações e pode falhar na detecção de muitos pacotes malignos.

Um sistema baseado em anomalias procura por cadeias de pacote que são estatisticamente incomuns, não recorrendo a conhecimentos prévios de outros ataques. O desafio é distinguir o tráfego normal de tráfegos estatisticamente incomuns.

3.4.3.3 Protocolos que oferecem segurança

Considerando que a Internet é a base para diversos sistemas distribuídos como, por exemplo, ambientes em nuvem, é possível oferecer serviços seguros por meio de protocolos que atuam em diferentes camadas. Embora a segurança da camada de rede possa oferecer cobertura total cifrando todos os dados nos datagramas, ou seja, todos os segmentos da camada de transporte, e autenticando todos os endereços IPs dos destinatários, ela não pode prover segurança no nível do usuário. Por exemplo, uma página de comércio não pode confiar na segurança da camada IP para autenticar um cliente que vai comprar mercadorias nessa página. Assim, existe a necessidade de uma funcionalidade da segurança em camadas superiores bem como cobertura total em canais inferiores.

Quando a segurança é oferecida para um protocolo específico da camada de aplicação, a aplicação que usa o protocolo (por exemplo, email) utilizará um ou mais serviços de segurança,

como sigilo, autenticação ou integridade. Entre os protocolos, pode-se citar o HTTPS. O HTTPS é uma implementação do protocolo HTTP (*Hyper Text Transfer Protocol*) sobre uma camada adicional de segurança que utiliza o protocolo SSL/TLS. Essa camada adicional permite que os dados sejam transmitidos por meio de uma conexão criptografada e que se verifique a autenticidade do servidor e do cliente por meio de certificados digitais (KUROSE; ROSS, 2013). O protocolo HTTPS é utilizado, em regra, quando se deseja evitar que a informação transmitida entre o cliente e o servidor seja visualizada por terceiros, como por exemplo, no caso de compras online.

Quando a segurança é oferecida por um protocolo da camada de transporte, todas as aplicações que usam esse protocolo aproveitam os serviços de segurança do protocolo de transporte. Os protocolos SSL e TLS são amplamente utilizados e são responsáveis por prover segurança na camada de transporte. Eles são dois protocolos de comunicação diferentes que permitem que os aplicativos se comuniquem de forma segura na Internet utilizando a criptografia dos dados. O SSL e TLS são protocolos que fornecem comunicações seguras na Internet para atividades como navegação na *Web*, email, mensagens instantâneas e outras transferências de dados.

As diferenças entre o TLS 1.0 e SSL 3.0 são relativamente pequenas, mas suficientes para que eles não interoperem. O TLS tem a capacidade de trabalhar em portas diferentes e usa algoritmos de criptografia mais fortes como o HMAC (*keyed-Hashing for Message Authentication Code*), enquanto que o SSL usa apenas o MAC (*Message Authentication Code*).

O protocolo TLS foi criado como o sucessor do SSL e é frequentemente usado como uma configuração nos programas de email. Assim como o SSL, o TLS pode ter um papel em qualquer transação cliente-servidor.

O protocolo SSL provê a privacidade e a integridade de dados entre dois sistemas que se comunicam pela Internet. Isto ocorre através da autenticação das partes envolvidas e da cifra dos dados transmitidos entre as partes. Esse protocolo ajuda a prevenir que intermediários entre as duas pontas da comunicação tenham acesso indevido ou falsifiquem os dados transmitidos (RESCORLA, 2001). O protocolo SSL suporta diferentes algoritmos criptográficos para uso em operações de autenticação, transmissão de certificados e estabelecimento de sessões. Como exemplo, pode-se citar o seu protocolo de *handshake* (TANENBAUM, 2003).

Quando a segurança é oferecida na camada de rede em base *host* para *host*, todos os segmentos da camada de transporte e, portanto, todos os dados da camada de aplicação, aproveitam os serviços de segurança da camada de rede. O IPsec é responsável por prover segurança na camada de rede. Segundo Kurose e Ross (2013), ele é uma extensão do protocolo IP que visa ser o método padrão para o fornecimento de:

- Privacidade do usuário, aumentando a confiabilidade das informações fornecidas pelo usuário para uma localidade da Internet, como bancos;

- Integridade dos dados, garantindo que o mesmo conteúdo que chegou ao seu destino seja o mesmo da origem, e;
- Autenticidade das informações ou prevenção de falsificação de identidade, garantindo que uma pessoa é quem diz ser.

O IPsec pode ser usado de dois modos. No modo de transporte o cabeçalho IPsec é inserido logo após o cabeçalho IP e somente a mensagem é criptografada. O roteamento permanece intacto, desde que o cabeçalho do IP não seja modificado e nem cifrado. No modo de tunelamento, todo o pacote IP, incluindo o cabeçalho, é encapsulado no corpo de um novo pacote IP. O tunelamento é usado para comunicações entre redes, por meio de túneis seguros entre os roteadores, ou comunicações de *host-a-rede* e de *host-a-host* sobre a Internet (STALLINGS; VIEIRA, 2008). Esse procedimento é utilizado em uma VPN (*Virtual Private Network*).

Uma VPN é uma conexão estabelecida sobre uma infraestrutura pública ou compartilhada, usando tecnologias de tunelamento e criptografia para manter seguros os dados trafegados. VPNs seguras usam protocolos de criptografia por tunelamento que fornecem a confidencialidade, autenticação e integridade necessárias para garantir a privacidade das comunicações requeridas. Quando adequadamente implementados, estes protocolos podem assegurar comunicações seguras através de redes inseguras (KUROSE; ROSS, 2013).

Após a descrição de mecanismos utilizados para combater as diversas ameaças em computação em nuvem, a próxima Seção apresenta alguns trabalhos que os aplicam na prática.

3.5 Estado da Arte

Nesta seção são abordados diversos trabalhos disponíveis na literatura que apresentam desde abordagens relacionadas à segurança em computação em nuvem, como ameaças, princípios de segurança e contramedidas, até o modo que a segurança é aplicada pelos provedores de serviços e pela comunidade acadêmica.

3.5.1 Surveys

Um estudo sobre a segurança nos diferentes modelos de serviço é apresentado por Subashini e Kavitha (2011). Após discutir sobre os principais aspectos críticos em cada modelo, os autores apresentam uma visão geral das soluções de segurança disponíveis na literatura. Choubey, Dubey e Bhattacharjee (2011) identificam as principais vantagens, desvantagens e compensações entre custo e segurança em um ambiente em nuvem. Srinivasamurthy e Liu (2010) apresentam estudos sobre as vantagens de uma arquitetura segura e as diferentes ameaças à segurança com algumas contramedidas existentes para minimiza-las.

No trabalho apresentado em (ZHOU *et al.*, 2010), os autores elaboraram um estudo sobre as preocupações de segurança e privacidade de muitos provedores de computação em nuvem. Foram consideradas questões relacionadas à disponibilidade, confidencialidade, integridade, controle e auditoria. Além disso, também foram discutidos alguns problemas relacionados ao armazenamento de dados.

Vaquero, Rodero-Merino e Morán (2011) fornecem uma visão abrangente de segurança em IaaS. O estudo foca nas questões de segurança relacionados a multialocação de recursos, analisando-as e categorizando-as de acordo com as ameaças à computação em nuvem publicadas pela CSA em 2010. Além disso, os autores descrevem a segurança da rede, da virtualização e física em uma infraestrutura fornecida como serviço.

Ahuja e Komathukattil (2012) analisam algumas ameaças comuns e os riscos associados às nuvens. Eles apresentam abordagens para lidar com essas ameaças e riscos, além de modelos de segurança dos principais provedores de nuvem.

Rodero-Merino *et al.* (2012) fazem um levantamento sobre o estado da arte de segurança em PaaS. Eles se concentraram em plataformas baseadas em compartilhamento, focando em plataformas .NET e Java com ênfase em isolamento, contabilidade dos recursos e propriedades de segurança das plataformas.

Em Xiao e Xiao (2013), os autores apresentam uma revisão sistemática das questões de segurança nas nuvens com base em uma metodologia orientada para o atributo. Os atributos utilizados foram confidencialidade, integridade, disponibilidade, responsabilidade e preservação da privacidade. Para cada atributo, algumas ameaças foram analisadas juntamente com as soluções de defesa correspondentes.

Pearce, Zeadally e Hunt (2013) elaboraram uma extensa pesquisa sobre virtualização de forma independente de plataforma, abordando os problemas de segurança em torno desse tópico. O trabalho discorre primeiramente sobre os conceitos básicos de virtualização para depois descrever uma arquitetura para virtualização de sistemas, com ênfase na rede. O estudo discute as incorretas suposições de segurança sobre o isolamento, monitoramento e duplicação do sistema por meio de VMs e apresenta ameaças resultantes de propriedades fortes de virtualização e de fracas implementações dos requisitos dessa tecnologia. Recomendações para implementações mais seguras também são apresentadas.

Aguiar, Zhang e Blanton (2014) apresentam um estudo sobre vários tópicos relacionados ao armazenamento e à segurança no processamento de dados. Tais tópicos incluem a autenticação e autorização, virtualização, serviços *Web*, responsabilidade e disponibilidade. Em seguida, os autores apresentam técnicas e mecanismos para a garantia de privacidade no armazenamento e autenticação de dados e processamentos terceirizados.

No trabalho apresentado em Fernandes *et al.* (2014), os autores realizam uma ampla revisão da literatura sobre segurança em computação em nuvem. Eles abordam vários temas-

chave como vulnerabilidades, ameaças e ataques, e propõem uma taxonomia de classificação na qual os trabalhos analisados são divididos.

Por fim, em [Ali, Khan e Vasilakos \(2015\)](#) os autores detalham as questões de segurança que surgem devido à própria natureza da nuvem e apresentam as soluções recentes disponíveis na literatura para combater esses problemas. As vulnerabilidades de segurança na computação em nuvem móvel também são destacadas.

Nos trabalhos analisados são apresentados estudos que demonstram a importância e a necessidade de contramedidas eficientes para as diversas ameaças de segurança presentes em um ambiente em nuvem. No entanto, esses estudos não fornecem aos pesquisadores e demais leitores um embasamento teórico, sendo necessária uma análise da forma que os provedores, clientes e a comunidade científica lidam na prática com as questões de segurança em nuvem.

3.5.2 *Segurança nos Provedores*

Nessa Seção são analisadas algumas práticas de segurança aplicadas por alguns provedores de nuvem. Entre os provedores analisados estão: *Amazon Web Services*, *Google Apps*, *Force.com* do Salesforce e *SmartCloud Enterprise* da IBM.

3.5.2.1 *Amazon Web Services*

A Amazon oferece a *Amazon Web Services* (AWS), que consiste em um conjunto de serviços *Web* para computação em nuvem, no qual os clientes podem optar por um ambiente de segurança com *firewalls* e sistemas de detecção de intrusão, por exemplo. A infraestrutura da AWS é protegida por extensos sistemas de monitoramento de segurança. Esses sistemas oferecem medidas de segurança importantes, como proteção contra negação de serviço (DDoS) e detecção da senha por força bruta nas contas da AWS.

Os componentes de infraestrutura da AWS são continuamente verificados e testados. Enquanto algumas organizações realizam verificações de vulnerabilidades em seus recursos uma vez por trimestre ou uma vez por mês, os responsáveis pela segurança na AWS realizam verificações várias vezes ao dia.

O *hardware* que está no fim da vida útil é substituído pelos mais recentes processadores, que não só melhoram o desempenho, mas também incluem tecnologias de segurança como as últimas instruções para acelerar operações de criptografia (por exemplo, a instrução Intel AES-NI para o algoritmo AES, e a Intel RDRAND para a geração aleatória de números) e o chip *Trusted Platform Module* para habilitar recursos de segurança baseados em *hardware*, como armazenamento seguro e verificação de *software* do *host*. Em caso de alguma falha em um *data center*, processos automatizados desviam o tráfego de dados do cliente da área afetada para outros *data centers* de forma balanceada.

A rede de produção da AWS é segregada da rede corporativa da Amazon e exige um conjunto separado de credenciais de acesso, que consiste em autenticação de chave pública SSH (*Secure Shell*) por meio de uma MFA (*Multi-Factor Authentication*). A MFA é uma prática simples que adiciona uma camada extra de proteção ao nome de usuário e senha. Esse acesso é monitorado e analisado diariamente por gerentes de segurança da AWS.

A AWS também procura maneiras de reduzir o conflito entre serviços e processos de segurança existentes que afetam o desempenho. Por exemplo, em 2014, a *Amazon CloudFront* adicionou os *Tickets* de sessão SSL, que salvam as informações de negociação SSL entre o cliente e o servidor e, portanto, aceleram o processo de negociação quando a conexão precisa ser retomada ou reiniciada. Esses recursos operam de forma transparente e não precisam de configurações por parte do cliente.

Além disso, a AWS fornece relatórios de certificação que descrevem como a infraestrutura de nuvem cumpre os controles exigidos pelas normas de segurança. Ela obteve conformidade com uma extensa lista de normas de segurança globais, incluindo *International Organization for Standardization – ISO 27001*, *Security Operations Center – SOC*, *Payment Card Industry Data Security Standard – PCI DSS*, *Australian Signals Directorate – ASD*, *Information Security Manual – ISM* e *Singapore Multi-Tier Cloud Security Standard – MTCS SS 584*.

Por fim, a Amazon utiliza a *AWS Compliance*, que permite que os clientes compreendam os controles robustos estabelecidos na AWS para manter a segurança e a proteção de dados. Ela garante a segurança da infraestrutura adjacente do provedor e o cliente se responsabiliza pelas iniciativas de conformidade relacionadas a tudo o que for colocado na infraestrutura da AWS. Dessa forma, a AWS permite aos clientes adicionar camadas de segurança para proteger os seus servidores virtuais e para garantir a segurança dos seus usuários.

Caso um cliente se interesse, a AWS oferece o *AWS Identity and Access Management*, que permite que o cliente controle com segurança o acesso aos serviços e recursos da AWS para seus usuários. Por meio do IAM, um cliente pode criar e gerenciar usuários e grupos da AWS e usar permissões para conceder e negar acesso aos recursos¹⁴. No entanto, a utilização de mecanismos de segurança adicionais pode gerar custos adicionais para os clientes.

3.5.2.2 Google Apps

O *Google Apps* consiste de um conjunto de serviços baseado em aplicações *Web* fornecidas como SaaS pelo Google. Ele usa um sistema de arquivos distribuído para armazenar os dados, no qual a replicação é usada para distribuir os dados para múltiplos sistemas a fim de evitar pontos únicos de falha. A arquitetura do aplicativo e da rede executada pelo Google é projetada para ter o máximo de segurança e de disponibilidade. Ela está preparada para enfrentar falhas de *hardware*, onde um *software* de *failover* robusto é utilizado para suportar as perturbações.

¹⁴ <http://d0.awsstatic.com/whitepapers/Security/AWS%20Security%20Whitepaper.pdf>

Todos os sistemas são redundantes em sua própria concepção e cada subsistema é independente de qualquer servidor físico ou lógico específico para seu funcionamento contínuo.

O Google analisa o tráfego interno a procura de comportamentos suspeitos. A conectividade SSL/TLS está disponível para todos os clientes do Google Apps e está ativada por padrão para os novos clientes. Se o cliente ativar as conexões por HTTPS, o Google forçará o uso do HTTPS quando seus usuários acessarem a maioria dos serviços do *Google Apps* como o Gmail, Google Agenda, Google Drive, Google Sites e o Bate-papo do Google.

O Google Apps está integrado nos sistemas padronizados de *log* único da Web que utilizam o padrão SAML 2.0. As próprias organizações podem fazer a integração ou podem trabalhar com um parceiro do Google para realizar essa tarefa. Dessa forma, o SSO (*Single-Sign-On*) pode ser utilizado em conjunto com os sistemas dos clientes, como por exemplo o LDAP (*Lightweight Directory Access Protocol*), permitindo que as empresas mantenham o controle sobre o gerenciamento dos mecanismos de autenticação.

Além disso, o Google fornece o CAPTCHA (*Completely Automated Public Turing test to tell Computers and Humans Apart*), um tipo de medida de segurança conhecido como autenticação por desafio e resposta. Trata-se de um teste de confirmação de *login* que somente um ser humano é capaz de efetuar, protegendo a conta contra *spams*, descryptografia de senhas e outras formas de acesso digital a contas não autorizadas. O Google utiliza o teste CAPTCHA para reforçar a segurança em torno dos pontos mais sensíveis do acesso às contas.

Os *clusters* de computação são projetados para terem resiliência e redundância, eliminando pontos de falhas individuais e minimizando o impacto de riscos ambientais e de falhas comuns nos equipamentos. O acesso aos *data centers* é limitado, onde apenas os funcionários selecionados são autorizados. O Google também monitora os *logs* do sistema para atividades inesperadas, como tentativa de acesso aos dados de clientes. Qualquer acesso administrativo é registrado e revisto periodicamente.

Por fim, o Google protege seus sistemas operacionais utilizando um *software* que os monitoram para modificações binárias. Quaisquer diferenças encontradas entre um sistema operacional e uma imagem padrão ativam um mecanismo de recuperação que restaura automaticamente o sistema operacional para o seu estado padrão¹⁵.

3.5.2.3 Force.com

O Salesforce oferece a Force.com, uma PaaS para o desenvolvimento de aplicativos em nuvem. Nesse ambiente as senhas dos clientes são protegidas e armazenadas em bancos de dados após a aplicação da função *hash* MD5. Há um rigoroso controle de acesso por meio de autenticação usando SAML, permitindo que os serviços Web filiados troquem informações de autenticação e de autorização. Além de oferecer soluções para o gerenciamento de identidades

¹⁵ <https://cloud.google.com/security/whitepaper>

(*Identity Management* – IdM), Force.com também permite que uma organização escolha o seu próprio método de autenticação, como por exemplo o LDAP, em conjunto com SSO.

Quando um usuário solicita uma conexão pela primeira vez usando um novo endereço, a plataforma reconhece esse procedimento, envia um email para o usuário, e solicita que o usuário confirme a sua identidade. Após esse procedimento, o navegador do usuário mantém um *cookie* criptografado para agilizar as solicitações de conexão futuras.

Force.com também pode exigir que os usuários realizem um teste de verificação do usuário (CAPTCHA) para exportar dados. Este teste de entrada de texto simples impede que programas automatizados atacantes acessem dados de uma organização.

A fim de proteger as suas redes, Force.com verifica todos os pacotes que trafegam pela rede usando *firewalls*. Assim, todas as requisições de serviços provenientes de endereços desconhecidos ou suspeitos são negadas. Além disso, os protocolos de segurança TLS e SSL também são usados para criptografar todo o tráfego.

Force.com emprega um número de ferramentas de segurança sofisticadas que monitoram as atividades na plataforma em tempo real para expor diversos tipos de eventos maliciosos, ameaças e tentativas de intrusão. Os ataques externos comuns são detectados utilizando sistemas de detecção de intrusão. Ferramentas de gerenciamento de eventos que correlacionam as ações do usuário com os dados são usadas para monitorar as atividades das aplicações e do banco de dados e gerar alertas apropriados, caso necessário.

Por fim, Force.com é certificada com os mais altos padrões de privacidade, aceitos por diversas indústrias internacionais. Entre essas certificações destacam-se a *TRUSTe Certified Privacy Seal*, *EU Safe Harbor*, SAS 70 Type II, SysTrust e ISO 2700¹⁶.

3.5.2.4 SmartCloud Enterprise

A *IBM SmartCloud Enterprise* (IBM Cloud) consiste de uma IaaS que permite acesso rápido a ambientes de servidores virtuais. Ela emprega um modelo de responsabilidade compartilhada, na qual a IBM fornece uma infraestrutura de gerenciamento, física e logicamente segura, e ambientes de sistema operacional base para os clientes desenvolverem implementações em um ambiente de nuvem. O cliente por sua vez assume a responsabilidade por manter a segurança dos recursos virtuais depois que eles são fornecidos.

O provisionamento de recursos nesse ambiente pode ser feito por meio de um portal de autoatendimento ou por meio de APIs. A infraestrutura usada para implementar esses pontos de entrada é implementada em instalações protegidas da IBM usando uma arquitetura multicamada e por zona. Esses recursos estão sujeitos aos rígidos requisitos e processos de segurança interna da IBM, onde ela usa ofertas e produtos reconhecidos (como o *Rational AppScan*¹⁷) para verificar,

¹⁶ http://www.salesforce.com/assets/pdf/misc/WP_Forcedotcom-Security.pdf

¹⁷ <http://www.ibm.com/developerworks/downloads/r/appscan/>

monitorar e gerenciar o ambiente. Todas as comunicações com os recursos (por exemplo, um navegador da *Web* ou um aplicativo personalizado usando APIs) são protegidas usando SSL sobre HTTP. Além disso, a IBM realiza análises regulares de vulnerabilidade, gerenciamento de correção, verificação automatizada do funcionamento e revisões de código do ambiente, incluindo a infraestrutura de rede, por meio de IDS e IPS.

Como parte do modelo de responsabilidade compartilhado da *IBM SmartCloud Enterprise*, o cliente é responsável por todos os aspectos de segurança dos recursos fornecidos no ambiente da nuvem. Para o gerenciamento de identidade, o sistema padrão de *IBM Web Identity* usado por todos os sistemas da IBM é empregado.

Após um cliente se inscrever para o serviço *IBM Cloud*, o identificador especificado durante o processo de inscrição é designado como um administrador de conta corporativa. Por meio do portal de autoatendimento do *IBM Cloud*, o administrador da conta tem a capacidade de adicionar, excluir e modificar identificadores de usuários adicionais que podem ser usados para fornecer recursos da nuvem, como instâncias, imagens e armazenamento. Ao adicionar outros identificadores ao ambiente, é responsabilidade do cliente gerenciar todos os identificadores de usuário da conta com base em seus próprios requisitos.

Uma oferta opcional da *IBM Cloud* é um serviço de rede privada virtual. Cada VPN opcional fornece ao cliente um túnel baseado em IPsec sobre a Internet entre o *gateway* compatível com IPsec de um cliente e um *data center* da *IBM Cloud*. Com a opção de VPN, o cliente recebe uma rede local virtual (VLAN) privada. Sempre que ele fornecer uma instância, ele poderá escolher entre fornecer a instância na VLAN pública ou na VLAN privada. Dessa forma, tem-se a opção de uma comunicação criptografada de dados sobre a Internet e um nível adicional de isolamento dentro da rede virtual da *IBM Cloud*.

A *IBM Cloud* permite que os clientes configurem regras personalizadas de *firewall* nos níveis do *host* e do sistema operacional convidado. Cada imagem também é configurada por padrão para executar regras do *firewall* do *software* que realizarão a filtragem de porta nominal. Imagens de Linux são configuradas para usar *IPTables* e permitem, por padrão, a porta TCP 22 (SSH). As instâncias do Windows são configuradas para permitir a porta TCP 3389 (*Remote Desktop Protocol – RDP*) por padrão. Por meio do *firewall* do Windows ou do *IPTables* nos sistemas Linux, os clientes sempre podem adicionar outras camadas de proteção à sua instância.

Conforme afirmado anteriormente, o acesso padrão ao sistema operacional da máquina virtual serve para permitir privilégios totais ao cliente. Como resultado, os clientes têm controle total sobre a forma que os dados são manipulados dentro de seus ambientes. Um cliente pode implementar qualquer conjunto de ferramentas de *software* para mover os dados e é responsável pela manutenção desse conjunto de ferramentas e pela administração de quaisquer controles de acesso. Além disso, eles podem considerar medidas de segurança adicionais para seus dados, como a criptografia do sistema de arquivos.

A IBM tem uma política de não mover ou migrar os recursos fornecidos por um cliente (imagens, instâncias, armazenamento persistente) de um *data center* para outro. Quando o cliente fornece um recurso, ele escolhe em qual *data center* esse recurso será fornecido. Essa política pode ser importante para clientes com preocupações de segurança relacionadas à movimentação dos dados fora de determinadas áreas geográficas¹⁸.

Dessa forma, não só a IBM, mas os outros provedores analisados, fornecem contramedidas que garantem a segurança básica dos serviços. Mecanismos adicionais também são ofertados, porém eles podem gerar custos adicionais aos clientes. Além disso, vale destacar que esses provedores não fornecem dados estatísticos que representam o impacto gerado pelos mecanismos de segurança utilizados sobre o desempenho dos serviços ofertados.

A próxima Seção discorre sobre os diferentes meios que a comunidade científica lida com as questões de segurança em um ambiente em nuvem. Para isso, alguns trabalhos disponíveis na literatura foram analisados.

3.5.3 Mecanismos Disponíveis na Literatura

Há diversos trabalhos que visam solucionar alguns dos problemas de segurança em computação em nuvem como os apresentados em (MARTY, 2011) para controle de *logs*; (LOMBARDI; PIETRO, 2011) que utilizam a virtualização para aumentar a segurança em nuvem, protegendo a integridade das máquinas virtuais e os recursos da infraestrutura; (YAN; RONG; ZHAO, 2009) para gerenciamento de identidades; e (HU; AHN; KULKARNI, 2011) para o controle de acesso.

Chonka *et al.* (2011) oferecem uma solução XML (*eXtensible Markup Language*), chamada SOTA (*Service-Oriented Traceback Architecture*), para rastrear a origem de ataques como os de negação de serviço baseados em HTTP. Essa arquitetura foi implementada em um sistema de nuvem chamado CTB (*Cloud Trace Back*) e é baseada no algoritmo DPM (*Deterministic Packet Marking*) (BELENKY; ANSARI, 2003). Os autores introduziram uma rede neural de *back-propagation*, chamada *Protector Cloud*, que foi treinada para detectar e filtrar os ataques. De acordo com os resultados, a ferramenta foi capaz de detectar e filtrar a maior parte das mensagens de ataque, além de identificar a sua origem dentro de um curto período de tempo.

Yang e Jia (2012) abordam o problema de auditoria no armazenamento de dados em uma nuvem apresentando e analisando as possíveis soluções disponíveis na literatura. Em seguida, eles propuseram um conjunto de requisitos necessários para o desenvolvimento de um protocolo de auditoria para o armazenamento de dados baseado na terceira parte confiável e apresentam os principais desafios encontrados durante esse desenvolvimento.

¹⁸ <http://www-935.ibm.com/services/us/en/it-services/security-services/cloud-security-services/index.html>

[Khorshed, Ali e Wasimi \(2012\)](#) realizaram uma pesquisa sobre as ameaças e contramedidas existentes para o ambiente de nuvem, bem como os desafios encontrados para a implementação dessas contramedidas. Eles propuseram um modelo de detecção de ataques que informa ao administrador do sistema a ocorrência e o tipo de ataque que está ocorrendo, baseado em técnicas de aprendizado de máquinas. Para isso, os autores desenvolveram um protótipo de nuvem sobre o qual foram realizados ataques. Para a detecção, eles utilizaram uma base de dados com informações como o número de pacotes enviados, recebidos e perdidos, número de portas abertas, uso da rede e de CPU, entre outras.

[Zissis e Lekkas \(2012\)](#) apresentam algumas ameaças e contramedidas disponíveis para um ambiente em nuvem. Em seguida, os autores propõem a introdução de uma terceira parte confiável encarregada de assegurar as características específicas de segurança. A solução proposta é baseada em criptografia, especificamente em uma PKI que opera em conjunto com SSO e LDAP, para garantir a autenticação, integridade e confidencialidade dos dados e das comunicações entre provedor e cliente. Segundo os autores, uma PKI fornece meios tecnicamente sólidos e juridicamente aceitáveis para a implementação de autenticação forte, autorização, não repúdio, confidencialidade e integridade dos dados. Essa infraestrutura se encaixa perfeitamente com o LDAP, onde a PKI pode acessar os serviços de diretório e as listas de revogação de certificados disponíveis no LDAP para implementar serviços de autenticação. Por fim, uma PKI em conjunto com SSO permite que os usuários naveguem entre fronteiras interorganizacionais sem a necessidade de digitar repetidamente as senhas para acessar recursos por meio de uma rede. Ainda segundo os autores, a solução apresenta um nível de segurança disponível para todas as entidades envolvidas, no qual a confiança entre as partes é mantida.

No trabalho apresentado por [Modi et al. \(2013\)](#) os autores discutem sobre vários tipos de invasões que podem ameaçar a integridade, confidencialidade e disponibilidade dos serviços em nuvem. Eles enfatizam e analisam o uso de técnicas de detecção e de prevenção de intrusão apresentando as suas vantagens e desvantagens. Eles afirmam que a eficiência da detecção e prevenção da intrusão depende de parâmetros como as técnicas utilizadas, o posicionamento no interior da rede, a configuração, entre outros fatores. Entre as técnicas de IDS e IPS os autores abordam: detecção baseada em assinatura, detecção de anomalia, IDS baseado em rede neural, em lógica Fuzzy, em SVM (*Support Vector Machine*), em algoritmos genéticos e em técnicas híbridas.

Com relação à virtualização, tecnologia considerada como um dos pilares de computação em nuvem, várias pesquisas vêm sendo conduzidas não apenas sobre a aplicação de hipervisores na segurança de sistemas, mas também na melhoria da segurança dos próprios hipervisores. Essas pesquisas visam propor soluções para ameaças em máquinas virtuais como: ataques por meio de *system calls*, onde um invasor pode executar uma *system call* a partir de um processo e causar danos ao sistema, ataques de sobrecarga de *buffer*, invasão de memória, violação de identidades, controle de acesso às máquinas virtuais e aos recursos da nuvem, comunicação,

tráfego de dados e migração de máquinas virtuais (KARGER, 2005) (LAUREANO; MAZIERO, 2008) (WENG *et al.*, 2008) (JIN; HUH, 2011) (LI *et al.*, 2012) (WEI *et al.*, 2014).

3.6 Considerações Finais

Em sistemas distribuídos como a computação em nuvem, onde os dados dos clientes são transferidos e manipulados por outras organizações, em geral por provedores de serviços, a segurança é um fator crítico e motivo de pesquisas no âmbito acadêmico e industrial. Considerando a grande demanda de serviços em um ambiente de computação em nuvem, há a necessidade de medidas de segurança que garantam a privacidade, integridade, confidencialidade e disponibilidade dos dados e dos recursos computacionais.

Por isso, este Capítulo apresentou algumas técnicas utilizadas para garantir a segurança como criptografia, assinatura digital e entidades certificadoras, bem como as principais ameaças e contramedidas encontradas em um ambiente de computação em nuvem. Vários trabalhos apresentados comprovam a importância da correta aplicação das técnicas de segurança no combate aos diversos riscos e ameaças apresentados. No entanto, é importante destacar que a aplicação de mecanismos de segurança acarreta em queda de desempenho em um sistema computacional.

Os trabalhos e provedores analisados apresentam diversas contramedidas de segurança para as diversas ameaças em nuvem, mostrando a importância desse tema e o amplo escopo de pesquisa que essa área fornece. Embora as soluções propostas sejam muitas, ainda há questões que precisam ser sanadas, como por exemplo, na relação entre desempenho do serviço e a garantia de segurança.

No trabalho apresentado nesta tese, o objetivo não é propor novas contramedidas de segurança, e sim, analisar soluções disponíveis na literatura que permitam uma caracterização e quantificação da sobrecarga imposta pelos mecanismos de segurança utilizados sobre o desempenho do serviço. Esse estudo é apresentado no próximo Capítulo.

CARACTERIZAÇÃO DA SOBRECARGA IMPOSTA POR MECANISMOS DE SEGURANÇA

4.1 Considerações Iniciais

Nos Capítulos anteriores foi apresentada toda a fundamentação teórica utilizada no desenvolvimento do projeto de doutorado. Para definir modelos de negócio que relacionem segurança, desempenho e custo, análises sobre o impacto gerado por mecanismos de segurança sobre o desempenho em um ambiente em nuvem são necessárias. Por essa razão, considerando que a área de segurança em computação em nuvem é muito ampla, inicialmente é necessário um estudo detalhado sobre quais ameaças de segurança devem ser abordadas, bem como sobre quais mecanismos disponíveis na literatura devem ser utilizados para combater essas ameaças. Dessa forma, este Capítulo apresenta todo o procedimento adotado para a definição dos problemas de segurança abordados e dos mecanismos utilizados.

Na análise realizada são considerados trabalhos disponíveis na literatura que visam combater ameaças relacionadas ao acesso, armazenamento e a manipulação dos dados no provimento de um serviço por meio de máquinas virtuais. Por meio desses trabalhos foi possível aferir a sobrecarga imposta sobre o desempenho de um sistema na utilização de diferentes mecanismos de segurança. Esses dados foram utilizados para a modelagem e simulação de um ambiente em nuvem, o qual foi utilizado para a execução dos experimentos descritos no Capítulo

4.2 Aspectos Críticos Abordados

Todo sistema computacional precisa ser protegido contra as mais variadas ameaças a fim de garantir a confidencialidade, integridade e disponibilidade dos dados e dos recursos que o compõe. Porém, vários fatores precisam ser analisados, como a sensibilidade dos dados e dos recursos que uma aplicação irá manipular, para que a segurança seja dimensionada adequadamente sem comprometer o ambiente. Além disso, no processo de análise em que a segurança e o desempenho são confrontados, busca-se avaliar os impactos gerados por diferentes mecanismos de segurança sobre o desempenho utilizando diferentes métricas de QoS (LANDWEHR, 2001).

O fato de ter a Internet como a sua tecnologia base faz com que a computação em nuvem herde todas as preocupações de segurança que ameaçam a Internet. A infraestrutura em nuvem não diz respeito apenas ao *hardware*, no qual os dados são processados e armazenados, mas também a forma que os dados são transmitidos e acessados. Em uma nuvem os dados são transmitidos da origem para o destino passando por inúmeros dispositivos e infraestruturas de terceiros, o que possibilita a interceptação e alteração desses dados (RISTENPART *et al.*, 2009). Dessa forma, mesmo que vários mecanismos de segurança sejam implementados pelo cliente e pelo provedor, os dados não estão imunes aos ataques.

Os provedores de serviços fornecem recursos computacionais aos clientes para que eles hospedem seus dados ou executem alguma aplicação. Esses clientes também podem usar a infraestrutura contratada para oferecer serviços a outros clientes (no decorrer desta tese, essa segunda classe de clientes será chamada de usuários).

Em geral, os clientes podem utilizar diferentes sistemas operacionais e aplicações em suas máquinas virtuais, que podem proporcionar diversas vulnerabilidades de segurança como as mencionadas no Capítulo 3. As exigências de segurança são tratadas de diferentes formas pelos provedores e variam de um cliente para outro. Alguns clientes podem optar pela segurança básica, enquanto que outros podem exigir mecanismos adicionais. Por exemplo, um cliente que fornece um aplicativo financeiro como o *Internet Banking* necessita de mais segurança do que um outro cliente que fornece uma hospedagem *Web* básica ou o compartilhamento de arquivos.

Vários clientes podem ter os seus recursos virtuais hospedados no mesmo recurso físico gerando vulnerabilidades na segurança. Essas vulnerabilidades podem ser exploradas por um atacante para gerar diferentes tipos de ataques. Esses ataques podem ser direcionados tanto à infraestrutura física do provedor quanto às máquinas virtuais dos clientes.

O VM *Escape*¹, por exemplo, permite que a máquina virtual explore as vulnerabilidades em hipervisores, como por exemplo o Xen, e acesse o domínio de privilégios e o sistema operacional hospedeiro. Uma vez que diferentes clientes podem ter seus recursos hospedados no mesmo recurso físico, um atacante que obtém acesso ao domínio de privilégios pode realizar

¹ <http://itsecurity.telelink.com/vm-escape/>

diferentes tipos de ataques a outras VMs e a outros sistemas. Entre os possíveis ataques estão os de negação de serviço (*Denial of Service*), em que um sistema pode ser bombardeado por diversas requisições até que fique indisponível; o ataque de *starving*, no qual a VM de um cliente que compartilha recursos físicos com outros clientes nunca terá acesso a eles; e o ataque de captura e manipulação do tráfego de outras VMs (*Man-in-the-Middle*). Além disso, os recursos alocados a uma VM podem ser alterados, acarretando em violações de SLAs e cobranças indevidas.

Outro tipo de ataque é relacionado à forma que as máquinas virtuais são alocadas nos recursos físicos. Considere uma situação na qual um cliente possui diversas VMs alocadas em diferentes recursos físicos, os quais são compartilhados com outros clientes. Nesse caso, o cliente deseja garantir o compartilhamento dos seus dados entre as instâncias que ele contratou, ao mesmo tempo em que ele deseja proteger as suas VMs de possíveis ataques oriundos de sistemas que compartilham os mesmos recursos físicos. Caso o provedor não monitore as comunicações entre as VMs, um atacante pode explorar essa vulnerabilidade para gerar ataques às outras. Além disso, um atacante pode se beneficiar da capacidade de elasticidade presente na nuvem, comprometendo a infraestrutura e as VMs presentes nesse ambiente.

Dessa forma, o provedor deve fornecer mecanismos de segurança para o isolamento entre as máquinas virtuais de diferentes clientes e para o controle de acesso a elas. No entanto, como mencionado anteriormente, quando uma empresa (cliente) contrata um serviço de IaaS, por exemplo, e presta serviços a outros usuários, o provedor de IaaS é indiferente quanto às operações e ao gerenciamento de pedidos das empresas contratantes do seu serviço. Assim, caso o provedor não forneça mecanismos de segurança, é importante que o contratante assuma total responsabilidade por assegurar a sua infraestrutura.

Em outras palavras, atualmente nenhum dos provedores de serviços mais populares como Amazon S3, Google Drive e Microsoft Azure, fornece garantias de segurança em seus SLAs. A Amazon S3² e a Microsoft Azure³, por exemplo, apenas garantem a disponibilidade do serviço em seus SLAs. Se a disponibilidade do serviço cai abaixo de 99,9%, os clientes são recompensados de acordo com as cláusulas contratuais. A Amazon afirma que a segurança das máquinas virtuais é de responsabilidade dos clientes, uma vez que eles são livres para executar quaisquer tipos de sistemas operacionais ou aplicações. Dessa forma, os clientes são responsáveis por garantir a segurança de suas máquinas virtuais que são hospedadas na nuvem.

Dada a importância dos fatos mencionados, neste projeto de doutorado são abordados mecanismos que garantam a segurança no acesso, armazenamento e na manipulação dos dados no provimento de um serviço por meio de máquinas virtuais. Esses mecanismos são de suma importância para que a integridade, a disponibilidade e a confidencialidade dos dados e recursos em uma nuvem sejam garantidas.

O controle de acesso é de importância vital, uma vez que é por meio de seus mecanismos

² <http://aws.amazon.com/s3/sla/>

³ <http://azure.microsoft.com/en-us/support/legal/sla/>

que um cliente pode ou não acessar vários recursos e dados da nuvem. O acesso é controlado por um procedimento que estabelece a identidade do cliente com algum grau de confiança (autenticação), e só então concede determinados privilégios (autorização) de acordo com esta identidade (BATES *et al.*, 2013).

Com relação à segurança no armazenamento e manipulação de dados nos provedores, os gerenciadores da nuvem devem se precaver contra problemas acidentais como o vazamento de dados confidenciais, alteração dos dados, ou retorno de dados inconsistentes para diferentes clientes/usuários. Isso pode acontecer devido às falhas de *software* ou *hardware*, ou até mesmo configurações incorretas. Além disso, é importante se defender contra as violações de segurança por parte de atacantes, as quais são mais difíceis de serem detectadas e mais prejudiciais (WEI *et al.*, 2014). Por exemplo, adversários externos podem invadir o gerenciamento dos recursos no provedor ou os próprios funcionários podem cometer um ataque interno, prejudicando a infraestrutura da nuvem e os clientes.

Com base nos aspectos críticos descritos, mecanismos responsáveis pelo controle de acesso aos dados e recursos e que garantam a integridade dos dados, das operações e das VMs por meio da segurança durante o tráfego, armazenamento e processamento dos dados são utilizados nas análises da sobrecarga imposta por eles sobre o desempenho do ambiente. Esses mecanismos foram selecionados a partir de um conjunto de mecanismos apresentados na próxima Seção, o que permitiu uma caracterização da sobrecarga imposta sobre o desempenho do ambiente.

4.3 Análise dos Mecanismos

Há diversos trabalhos na literatura que visam solucionar problemas de segurança em computação em nuvem relacionados ao acesso, armazenamento e manipulação dos dados e recursos no provimento de um serviço por meio de VMs. Esta Seção apresenta uma análise de alguns trabalhos, o que possibilitou uma caracterização da sobrecarga imposta por mecanismos de segurança sobre o desempenho do ambiente. Eles foram selecionados, pois além de abordarem os aspectos críticos definidos, eles apresentam uma avaliação de desempenho que permite a quantificação do *overhead* gerado.

Sanka, Hota e Rajarajan (2010) discutem os desafios relacionados ao acesso seguro aos dados em computação em nuvem. Os autores propõem uma abordagem de segurança, na qual mecanismos de controle de acesso e de criptografia são utilizados na proteção dos dados. Eles modificaram o mecanismo de troca de chaves *Diffie-Hellman* (D-H) (STEINER; TSUDIK; WAIDNER, 1996), de forma que ele pode ser utilizado tanto pelo provedor de serviços quanto pelos clientes para o compartilhamento de chave simétrica e manipulação dos dados. Além disso, foi proposto um sistema de controle de acesso baseado em capacidade com técnicas criptográficas para uma plataforma em nuvem, e uma lista de capacidade que é usada para especificar os

direitos de acesso dos usuários. Um ambiente simulado foi utilizado para a execução e análise dos experimentos no qual a solução proposta gerou um *overhead* de aproximadamente 28% no sistema.

No trabalho apresentado em (CASOLA *et al.*, 2011), os autores propõem uma solução baseada em mecanismos de controle de acesso de granularidade fina, por meio de autorização e autenticação, e em identidade federada que permite a cooperação e interoperabilidade entre recursos de uma nuvem e de uma grade computacional. O *overhead* introduzido pela arquitetura multicamadas e pelos mecanismos de segurança são mensurados por meio de experimentos realizados em um protótipo, chamado PerfCloud, o que permitiu análises comparativas entre segurança e desempenho. De acordo com os resultados, a utilização de mecanismos de autorização como o XACML praticamente não gerou *overhead* sobre o sistema, menos que 0,01%. Com relação ao canal de transporte e o mecanismo de autenticação, ambos influenciaram o desempenho dos recursos virtuais e físicos. De acordo com os autores, ambos os fatores tiveram um grande impacto no tempo de resposta, 30% e 60%, respectivamente. Por fim, a utilização de uma TTP (*Trusted Third Party*) como uma segunda opção para realizar o processo de autenticação exerceu um grande impacto sobre o desempenho do ambiente devido ao protocolo de autenticação (*Secure Message*), ao canal de transporte (HTTP) e aos mecanismos de autorização (MapFile e XACML). O *overhead* gerado foi de aproximadamente 184% sobre o tempo de resposta para todas as combinações.

O objetivo do estudo apresentado por Chhabra *et al.* (2011) é investigar como proporcionar privacidade e integridade para aplicativos que são executados em nós de computação não confiáveis, nos quais o *hardware* e o sistema operacional são vulneráveis à ataques. Para isso, os autores propõem o SecureME (*Secure My Execution*), um mecanismo de *hardware* e *software* que fornece um ambiente de computação seguro. O SecureME visa proteger uma aplicação de ataques de *hardware*, utilizando um processador seguro, e também de ataques do sistema operacional, por meio da camuflagem da memória, permissão de paginação e proteção das chamadas de sistema. Experimentos foram executados em um ambiente simulado utilizando o simulador Simics (MAGNUSSON *et al.*, 2002), o que possibilitou uma avaliação de desempenho usando *microbenchmarks* e diferentes cargas. Os autores verificaram que o SecureME adiciona um *overhead* sobre o tempo de execução de aproximadamente 13,5% em comparação com um sistema totalmente desprotegido.

Lombardi e Pietro (2011) mostram como a virtualização pode aumentar a segurança da computação em nuvem, protegendo tanto a integridade das máquinas virtuais quanto os componentes da infraestrutura da nuvem. Eles propõem uma arquitetura, chamada ACPS (*Advanced Cloud Protection System*), que visa garantir a segurança dos recursos. O ACPS pode ser implantado em várias soluções de nuvem e pode efetivamente monitorar a integridade das VMs e dos componentes da infraestrutura de forma transparente. Para isso, verificações periódicas de *checksum* dos arquivos e bibliotecas são realizadas e, de acordo com os resultados, o ACPS

pode reagir localmente à violações de segurança, bem como notificar o gerenciador do ambiente na ocorrência de tais eventos. Um protótipo foi implementado utilizando o Eucalyptus e o OpenECP. O protótipo foi testado contra ataques conhecidos na literatura e uma avaliação da arquitetura proposta foi realizada com diferentes tipos de carga. Os resultados mostram que ACPS é resiliente contra ataques e que o *overhead* introduzido foi de aproximadamente 6%.

No trabalho apresentado em [Popa et al. \(2011\)](#), os autores realizam uma avaliação de desempenho dos mecanismos de segurança disponíveis no CloudProof. Esse mecanismo foi desenvolvido para garantir a confidencialidade e a integridade dos dados, a escrita serializada de múltiplos usuários e a leitura atualizada da última versão do arquivo. Além disso, o CloudProof garante o controle de acesso e é capaz de detectar violações na integridade dos dados compensando o cliente em caso de alguma ocorrência. De acordo com os autores, a utilização do CloudProof introduz um *overhead* de aproximadamente 17% quando comparado com experimentos nos quais esse mecanismo não foi utilizado. Há também uma redução de aproximadamente 10% no *throughput* durante o armazenamento de dados na nuvem. Os algoritmos criptográficos utilizados são as implementações .NET de SHA-1 para *hashing*, AES para criptografia simétrica e RSA com chave de 1024 bits para assinatura.

[Zhang et al. \(2011\)](#) propõem um abordagem transparente que protege a privacidade e integridade das máquinas virtuais dos clientes em infraestruturas virtualizadas. Segundo os autores, um protótipo chamado CloudVisor foi implementado, capaz de proporcionar privacidade e garantir a integridade, mesmo que o hipervisor e o *software* de gerenciamento das máquinas virtuais estejam sobre o controle de agentes atacantes. Alguns experimentos foram realizados para uma avaliação de desempenho. Nesses experimentos foi avaliado o comportamento do protótipo com diferentes números de clientes, números de núcleos de uma VM e quantidades de VMs. As operações de I/O correspondem ao gargalo do CloudVisor devido as operações de criptografia/descriptografia com o algoritmo AES e operações *hash*. Nos resultados o *overhead* gerado na utilização do CloudVisor com aplicações de I/O varia entre 4,5% e 54,5% quando comparado com os experimentos nos quais o virtualizador Xen foi utilizado. Essa variação está relacionada à quantidade de clientes.

[Bates et al. \(2013\)](#) propõem um modelo de controle de acesso para o problema de armazenamento de dados em computação em nuvem em *data centers* geograficamente distribuídos, nos quais podem haver problemas relacionados à proveniência dos dados. Os autores consideraram o gerenciamento e validação dos metadados de um sistema de proveniência confiável, e introduziram protocolos que permitem a transferência segura de metadados entre hospedeiros finais e entidades da nuvem. Usando esses protocolos, eles desenvolveram uma arquitetura em forma de protótipo para o controle de acesso baseado em proveniência de armazenamento em nuvem, chamado Cumulus, capaz de processar milhares de operações de leitura e escrita por segundo. Por meio de componentes replicados, o sistema gerou um *overhead* de aproximadamente 14%, o que segundo os autores, demonstra que o controle de acesso baseado em proveniência é uma

solução prática e escalável para a nuvem.

No trabalho apresentado em [He et al. \(2014a\)](#), os autores propõem um mecanismo de controle de acesso de granularidade fina para uma nuvem de armazenamento P2P chamado ACPC (*Access Control mechanism for P2P storage Cloud*). Por meio desse mecanismo é possível explorar o espaço de armazenamento em uma nuvem dos usuários participantes de um sistema P2P. No entanto, uma vez que os provedores e usuários estão em domínios diferentes, muitas vezes não confiáveis, alguns desafios surgem relacionados à segurança dos dados e ao controle de acesso. Para resolver esses problemas, os autores desenvolveram técnicas de criptografia CP-ABE (*Ciphertext-Policy Attribute-Based Encryption*) ([BETHENCOURT; SAHAI; WATERS, 2007](#)) e PRE (*Proxy Re-Encryption*) ([BLAZE; BLEUMER; STRAUSS, 1998](#)), chamadas de PCCP-ABE e PCPRE, respectivamente, que compõem o mecanismo ACPC. Dessa forma, segundo os autores, o ACPC pode resistir à ataques de colisão e proteger as informações dos usuários de forma eficaz. Uma análise de desempenho é apresentada, na qual o ACPC é comparado com outros mecanismos de ABE (*Attribute-Based Encryption*), considerando o *overhead* na geração da chave secreta, criptografia e descriptografia dos dados armazenados. Nos resultados, as operações realizadas pelo ACPC apresentaram os melhores resultados, introduzindo um *overhead* de aproximadamente 13,2%.

No trabalho apresentado em [Varadharajan e Tupakula \(2014\)](#), os autores apresentam uma arquitetura na qual o provedor pode fornecer segurança como um serviço aos seus clientes. O modelo de segurança proposto oferece tanto a segurança básica para que o provedor proteja a sua infraestrutura, quanto uma segurança flexível que fornece mecanismos de segurança adicionais. Para a segurança básica os autores propõem um componente chamado SPAD (*Service Provider Attack Detection*). Para a segurança flexível os autores propõem um componente chamado TSAD (*Tenant Specific Attack Detection*). Além disso, eles fornecem mecanismos contra ataques ao hipervisor. Para isso, eles usam a *Security Gateway* que especifica políticas e mecanismos para detectar ataques ao hipervisor. Uma análise de desempenho da arquitetura proposta é apresentada. Nos resultados, as operações realizadas pelo SPAD e TSAD introduziram um *overhead* de aproximadamente 6% com diferentes tipos de ataques.

[He et al. \(2014b\)](#) propõem uma arquitetura de segurança, chamada NSCC (*Network Security for Cloud Computing*), que visa resolver o problema de segurança na rede em uma nuvem. De acordo com os autores, NSCC lida com questões relacionadas à escalabilidade, tolerância à falhas e utilização dos recursos em uma nuvem de forma segura prevenindo o sistema contra ataques internos e externos. No NSCC, o SMD (*Security Management Domains*), o SMG (*Security Meta-Group*) e um *switch* virtual (*vSwitch*) trabalham em conjunto para garantir a segurança da nuvem. O SMD define as regras de encaminhamento do *vSwitch*, e este por sua vez, encaminha o tráfego interno e externo por meio do SMG. Por fim, o SMG realiza a inspeção e filtragem do tráfego de entrada e saída dos serviços dos usuários da nuvem. Alguns experimentos foram executados em um protótipo. Nos experimentos o *overhead* foi mensurado

por meio da comparação entre um ambiente com o NSCC e um ambiente sem segurança. De acordo com os resultados, o NSCC impôs um *overhead* de aproximadamente 9,3% sobre o *throughput* e latência da rede.

Os trabalhos analisados visam propor mecanismos específicos de segurança como contramedidas às ameaças específicas. Por esse motivo, o objetivo da análise realizada não é compará-los, visto que cada um possui as suas próprias características como modelagem, forma de abordar o problema de segurança, planejamento dos experimentos e ambiente de execução. O objetivo da análise é fazer um levantamento de trabalhos disponíveis na literatura que consideram os mesmos aspectos críticos de segurança abordados neste projeto, e que realizam uma avaliação que considera a sobrecarga gerada pelos mecanismos de segurança sobre o desempenho do ambiente.

A Tabela 2 apresenta as características dos trabalhos selecionados para a quantificação da sobrecarga imposta ao desempenho do ambiente. Nessa tabela são apresentados os problemas de segurança abordados por cada trabalho, bem como as soluções e mecanismos utilizados para a análise do *overhead* gerado sobre o ambiente. A partir dos trabalhos descritos nesta Seção, experimentos foram realizados no Capítulo 6, os quais consideram as sobrecargas impostas pelos mecanismos de segurança utilizados por [Casola et al. \(2011\)](#) e [Popa et al. \(2011\)](#). Esses trabalhos foram escolhidos, pois utilizam uma ampla variedade de mecanismos de segurança para lidar com os aspectos críticos abordados neste projeto.

Tabela 2 – Sobrecarga imposta por diferentes mecanismos de segurança sobre o desempenho dos ambientes

Problema	Trabalho	Solução	Mecanismos	Overhead
Acesso Armazenamento	(SANKA; HOTA; RAJARAJAN, 2010)	Controle de acesso Criptografia	Access Control Lists – ACLs baseadas em capacidade, MD5 e D-H	28%
	(CASOLA et al., 2011)	Autorização Autenticação	MapFile ou XACML Canal de Transporte (HTTP ou HTTPS) Protocolo (Conversation ou Message)	0,01% 30% 60%
		Federação de identidade	HTTP, Message, Mecanismo de Autorização	184%
	(POPA et al., 2011)	CloudProof	Algoritmos criptográficos (SHA-1, AES, RSA)	17%
	(ZHANG et al., 2011)	CloudVisor	Boot autenticado por meio de um TPM (Trusted Platform Module) e Criptografia (AES + hash MD5)	4,5% a 54,5%
	(BATES et al., 2013)	Cumulus	ACLs baseadas na proveniência, PEP (Policy Enforcement Point) e PDP (Policy Decision Point)	14%
Infraestrutura	(HE et al., 2014a)	ACPC	Sistema de reputação P2P (File Access Tree) PCCP-ABE e PCPRE	13,2%
	(CHHABRA et al., 2011)	SecureME	Camuflagem de memória, AISE (Address-Independent Seed Encryption) e árvore de Merkle	13,5%
	(LOMBARDI; PIETRO, 2011)	ACPS	Monitoramento e geração de alertas	6%
	(VARADHARAJAN; TUPAKULA, 2014)	SPAD TSAD	Monitoramento do tráfego, Security Gateway Políticas baseadas em anomalia e em assinatura	6%
Rede	(HE et al., 2014b)	NSCC	SMD, SMG, vSwitch e SIC (Security Inspection Chains)	9,3%

4.3.1 PerfCloud

O protótipo implementado em [Casola et al. \(2011\)](#), chamado PerfCloud, é composto por *clusters* virtuais hospedados em uma nuvem e, esta por sua vez, utiliza uma infraestrutura física

de uma grade computacional, compondo uma arquitetura de três camadas. Este protótipo utiliza o Globus Toolkit 4 (GT4)⁴ como *middleware* da grade.

Por padrão, o GT4 utiliza o conceito de descritores de segurança (*Security Descriptor – SD*) como método para configurar os requisitos de segurança e as políticas de clientes e serviços⁵. O descritor de segurança torna possível especificar os protocolos de comunicação e alguns mecanismos para melhorar a segurança em nível de mensagem (*Secure Message*), em nível de sessão e em nível de transporte (*Secure Conversation* e *Secure Transport*), os quais são baseados nas especificações *WS-Security* e *WS-SecureConversation* (ATKINSON *et al.*, 2002).

Com relação aos mecanismos de autorização, por padrão o GT4 oferece apenas mecanismos simples como o *MapFile*. Por esse motivo, os autores estenderam o mecanismo de autorização, a fim de suportar XACML, e forçaram o descritor de segurança a adotar canais de comunicação seguros para todos os serviços e recursos relacionados à nuvem.

Os autores utilizam federações de identidade para resolver problemas de segurança relacionados à interoperabilidade entre recursos físicos e virtuais. Cada *cluster* virtual possui a sua própria autoridade certificadora e seus certificados são validados por meio da REM (*Reference Evaluation Methodology*). Essa metodologia é capaz de analisar se uma CA é verdadeira ou não. A autenticação se baseia em uma PKI e na adoção de certificados digitais X.509.

Experimentos foram executados para avaliar a sobrecarga introduzida pela arquitetura de múltiplas camadas *cloudgrid* por meio da realização de medições no PerfCloud. Os autores desenvolveram um serviço sintético que apenas envia uma mensagem de resposta à requisição. A variável de resposta utilizada foi o tempo de resposta que foi medido na combinação dos diversos fatores definidos. Dessa forma, é possível ter uma visão sobre o *overhead* introduzido pelo PerfCloud.

O protótipo utiliza o *cluster Powercost*, que é composto por 40 nós, onde cada nó possui um Intel Xeon com 2 GB de RAM, interligados através de uma Ethernet Gigabit. Além disso, os autores configuraram um cliente que realiza 50 requisições do serviço sintético e, ao final do teste, armazena os tempos obtidos em um banco de dados. Dessa forma, nenhuma carga real foi utilizada nos experimentos. Com os resultados foi calculada a média, o desvio padrão e o intervalo de confiança para cada experimento.

De acordo com os resultados, a utilização de mecanismos de autorização como o XACML praticamente não gerou *overhead* sobre o sistema quando comparado com os outros dois mecanismos, menos que 0,01%. Isso porque os mecanismos básicos presentes na infraestrutura de segurança da grade *Grid Security Infrastructure – GSI* precisam instanciar objetos Java para a decisão de autorização. Dessa forma, o tempo exigido para invocar um mecanismo de autorização externo como o XACML é ofuscado pelo tempo necessário para carregar os objetos GSI.

⁴ <http://toolkit.globus.org/toolkit/docs/4.0/>

⁵ <http://toolkit.globus.org/toolkit/docs/4.0/security/authzframe/>

Com relação ao canal de transporte e o mecanismo de autenticação, ambos geraram sobrecarga sobre o desempenho dos recursos virtuais e físicos. De acordo com os autores, o *overhead* gerado é oriundo de dois fatores: a escolha entre HTTP e HTTPS e da criptografia utilizada nas mensagens SOAP. Ambos os fatores tiveram um grande impacto no tempo de resposta, 30% e 60%, respectivamente. Com base nisso, os autores implementaram uma outra abordagem de autenticação por meio de uma terceira parte confiável (*Trusted Third Party – TTP*). Dessa forma, foi possível instanciar um simples contêiner de HTTP e impor sobrecargas de segurança apenas nos serviços que precisam de uma infraestrutura segura.

A utilização de uma TTP para realizar o processo de autenticação exerceu um grande impacto sobre o desempenho do ambiente devido ao protocolo de autenticação (*Secure Message*), ao canal de transporte (HTTP) e aos mecanismos de autorização (Nenhum, MapFile e XACML). O *overhead* gerado foi de aproximadamente 184% sobre o tempo de resposta para todas as combinações. Dessa forma, a inserção de um sistema de interoperabilidade prejudicou o desempenho do ambiente através dos protocolos de autenticação de mensagem e de transporte, dos mecanismos de autorização e a adoção de uma federação de identidade.

Portanto, para as experimentações propostas nesta tese foram considerados apenas os mecanismos de autorização (XACML) e de autenticação (*Message*), os quais exercem uma sobrecarga de aproximadamente 60% sobre o desempenho do ambiente.

4.3.2 CloudProof

No trabalho apresentado em [Popa et al. \(2011\)](#), um protótipo chamado CloudProof foi implementado utilizando a plataforma da Microsoft Azure. Existem três partes envolvidas no CloudProof: o proprietário, entidade que adquire o serviço de armazenamento em nuvem (pode ser uma empresa ou um usuário doméstico); provedor que fornece o serviço de armazenamento em nuvem; e o usuário, com permissão de ler e/ou escrever dados na nuvem.

O proprietário dos dados é o único autorizado a dar permissões de acesso para os usuários. Os tipos de acesso são leitura e leitura/escrita. Cada bloco de usuários, chamado *block family*, possui uma lista de controle de acesso (*Access Control List – ACL*) que corresponde a uma lista de usuários, suas prioridades de acesso e as chaves necessárias para criptografia e descriptografia. Dessa forma, o usuário pode acessar e operar apenas sobre os blocos de dados permitidos.

Para garantir o controle de acesso à leitura, todos os dados armazenados na nuvem são criptografados com o algoritmo AES. Apenas os clientes com permissão de leitura possuem a chave para o processo de decodificação. Para garantir o controle de acesso à escrita, os autores utilizam um esquema de assinatura com chave pública, no qual cada *block family* possui uma chave de verificação pública e uma chave de assinatura privada. A chave de verificação pública é conhecida por todos, incluindo o provedor, enquanto que a chave de assinatura é conhecida apenas pelos usuários que possuem permissão para escrita definida na ACL.

Toda vez que um usuário modifica um bloco, ele gera uma assinatura que atesta a integridade do bloco modificado. Para isso, ele utiliza a função *hash* SHA-1 sobre o bloco de dados atualizado e a chave de assinatura privada (algoritmo RSA). Após esse procedimento, ele envia essa assinatura juntamente com seu pedido de escrita ao provedor. Este por sua vez, verifica a assinatura recebida e apenas realiza as alterações junto ao bloco de dados armazenados na nuvem, caso a veracidade e integridade da assinatura sejam comprovadas.

Para todas as operações realizadas uma certificação é gerada, a qual é utilizada para provar que a operação foi realizada por determinada entidade. Dessa forma, no processo de leitura e/ou escrita de dados de um determinado bloco de dados, a integridade das assinaturas das entidades comunicantes e a veracidade das certificações são analisadas. A estrutura da certificação depende do modelo de segurança desejado. Para isso, CloudProof fornece cinco modelos de segurança, onde cada um corresponde à adição de uma nova propriedade de segurança. Essas propriedades são: sem segurança, confidencialidade (C), confidencialidade e integridade (CI), os dois anteriores e escrita serializada (CIW), e por fim, segurança completa (CIWF).

Nos experimentos os autores avaliaram o desempenho do CloudProof, analisando o *overhead* gerado por cada modelo de segurança. Para os experimentos, inicialmente foi utilizada uma máquina *Intel Duo CPU 3.0 GHz* e 4.0 GB de RAM como cliente para gerar as requisições. Os autores realizaram 50 operações de escrita e 50 de leitura utilizando blocos de dados de 4KB e avaliaram o tempo médio gasto por operação. De acordo com os resultados, observou-se que o *overhead* aumentou conforme novas propriedades de segurança foram inseridas, onde a utilização de todas as propriedades gerou uma sobrecarga de aproximadamente 17%.

Portanto, para as experimentações apresentadas no Capítulo 6 foram considerados os mecanismos que garantem a segurança completa do ambiente.

4.4 Considerações Finais

Como visto no Capítulo anterior, os aspectos críticos de segurança em nuvem são tratados de diferentes formas pelos provedores e pela comunidade acadêmica. Considerando os provedores analisados, cada um possui uma forma diferente de oferecer segurança nos serviços prestados. No entanto, uma análise do impacto desses mecanismos sobre o desempenho do sistema é necessária. Pelo fato da segurança ser um fator crítico em nuvem, ela torna-se um importante diferencial entre os provedores de serviços.

Com base nos diversos trabalhos e mecanismos analisados neste Capítulo, a Tabela 2 foi definida. Essa tabela mostra os aspectos críticos abordados neste projeto, as contramedidas e os mecanismos utilizados para combater as ameaças relacionadas ao acesso, armazenamento e manipulação dos dados no provimento de um serviço por meio de máquinas virtuais. Dos trabalhos apresentados, dois foram selecionados para análises que consideram a sobrecarga

imposta pelos mecanismos de segurança sobre o desempenho do serviço oferecido em dois cenários apresentados no Capítulo 6.

Pelo fato do trabalho apresentado nesta tese considerar a hipótese de que é possível contrapor a sobrecarga gerada pelos mecanismos de segurança com a alteração dos recursos computacionais do ambiente, o próximo Capítulo apresenta um módulo de gerenciamento de recursos capaz de alterar a quantidade de recursos virtuais providos durante o tempo de execução, com impactos nos custos, na tentativa de cumprir as exigências feitas pelos clientes nas definições dos SLAs.

MÓDULO DE GERENCIAMENTO DE RECURSOS

5.1 Considerações Iniciais

Nos Capítulos anteriores foram abordadas questões relacionadas à segurança em computação em nuvem. Sabe-se que conforme mecanismos de segurança são aplicados na tentativa de contrapor as diversas ameaças, a sobrecarga imposta por esses mecanismos sobre o desempenho do serviço pode prejudicar o cumprimento das exigências feitas pelos clientes no SLA e o uso eficiente da infraestrutura da nuvem. Uma alternativa para contrapor o *overhead* imposto pelos mecanismos de segurança é alterar a quantidade de recursos computacionais alocados para a execução da demanda de serviços.

A nuvem é um ambiente altamente escalável, no qual a demanda de serviços pode mudar instantaneamente. Dessa forma, a alocação automática de recursos para atender essa demanda torna-se um tópico interessante tanto no âmbito acadêmico quanto no industrial. O correto provisionamento permite um melhor uso dos recursos computacionais disponíveis e, consequentemente, de toda a infraestrutura que compõe a nuvem, pois o mapeamento entre a carga e os recursos é mais eficiente. Além disso, ele ajuda no cumprimento das exigências feitas pelos clientes, sejam elas de desempenho e/ou segurança, e provê um grande dinamismo ao sistema.

A tarefa de prover mais recursos computacionais a um cliente, por exemplo, é altamente viável e de fácil disponibilização, uma vez que os recursos computacionais são virtualizados e podem ser considerados ilimitados na visão dos clientes. No entanto, é importante lembrar que a alocação de mais recursos computacionais influencia no custo final que é repassado ao cliente e requer mecanismos eficientes por parte do provedor (COUTINHO *et al.*, 2015).

O trabalho desenvolvido nesta tese visa, por meio da alteração de recursos em uma

nuvem, contrapor a sobrecarga imposta por mecanismos de segurança e garantir o desempenho contratado. No entanto, a alteração da quantidade de recursos alocados para um serviço influencia nos custos do serviço, e por essa razão, deve ser considerada no modelo de negócio.

Portanto, o provisionamento de recursos deve ser realizado automaticamente e dinamicamente baseado nos requisitos de segurança e de desempenho dos clientes e a um preço justo. Isto não é uma tarefa trivial, pois é necessário considerar alguns parâmetros importantes como orçamento, *deadline*, nível de segurança e/ou a qualidade de serviço desejada. Dessa forma, este Capítulo apresenta um módulo para o gerenciamento de recursos em um ambiente em nuvem que considera a manipulação dos recursos computacionais disponíveis a fim de contrapor o *overhead* imposto pelos mecanismos de segurança durante o tempo de execução e o impacto dessa manipulação no desempenho do sistema e no modelo de negócio.

Para isso, uma revisão e a análise dos mecanismos de alocação de recursos disponíveis na literatura são apresentadas na Seção 5.3. Com base nos trabalhos analisados, um novo módulo, chamado ReMM (*Resource Management Module*), é proposto. O ReMM é um módulo dinâmico e autogerenciável que visa a utilização eficiente dos recursos e o cumprimento do SLA do cliente. Ele usa uma abordagem baseada na diferenciação das capacidades dos recursos das máquinas virtuais, tornando possível aplicar a escalabilidade vertical e horizontal. Na escalabilidade horizontal altera-se a quantidade de máquinas virtuais (também chamadas de instâncias) disponíveis para um cliente. Por outro lado, na escalabilidade vertical altera-se a configuração dos recursos computacionais que compõem uma VM.

5.2 Provisionamento de Recursos

Em um ambiente em nuvem, quando um mecanismo de provisionamento de recursos bem elaborado é aplicado, as aplicações podem operar mais eficientemente, com redução nos custos, melhor utilização da infraestrutura disponível e melhor desempenho em momentos de pico e com variações na demanda de serviços. No entanto, o processo de provisionamento é complexo (JENNINGS; STADLER, 2014). De acordo com Guzek, Bouvry e Talbi (2015), isto requer a definição das melhores configurações de *software* e *hardware* para garantir que os SLAs sejam cumpridos, além de maximizar a eficiência e a utilização do sistema.

Há diversos desafios relacionados ao provisionamento de recursos e ao atendimento das requisições utilizando a infraestrutura de uma nuvem. Esses desafios integram a modelagem de carga, virtualização, modelagem de desempenho, depuração e monitoramento de aplicações nos recursos virtualizados (MUSTAFA *et al.*, 2015). Além disso, há situações imprevisíveis que podem prejudicar o eficiente provisionamento e atendimento das requisições durante o tempo de execução, dentre as quais podem ser citadas (CALHEIROS; RANJAN; BUYYA, 2011):

- **Erro de estimativa:** a combinação errada entre os recursos computacionais e aplicações

pode levar a uma sub ou superestimativa na demanda de clientes, o que exerce um grande impacto na QoS contratada e no custo do serviço;

- **Carga dinâmica:** como mencionado anteriormente, a nuvem é um ambiente com diferentes tipos de clientes que requisitam diferentes tipos de serviços. Por isso, diferentes picos de carga podem ocorrer, dependendo do dia e da época do ano, ou da popularidade de uma aplicação. Esses fatores são responsáveis por sérios problemas durante a estimativa do comportamento da carga e na definição dos recursos;
- **Comportamento inesperado:** disponibilidade, carga, *throughput* de recursos e conexões da rede podem variar de maneira imprevisível em ambientes de computação em grande escala como em uma nuvem. Essas instabilidades do sistema prejudicam a determinação dos recursos de forma eficiente durante o provisionamento.

Provedores como Amazon EC2 e Microsoft Azure utilizam uma metodologia de provisionamento de recursos na qual os clientes são responsáveis por estimar com precisão a quantidade de recursos necessários e selecionar a instância a ser contratada (HUU; MONTAGNAT, 2010). As máquinas virtuais são classificadas de acordo com suas configurações de memória, núcleos virtuais (vCPUs – *virtual CPUs*), e tamanho de disco, e o preço é definido baseado na classe, onde as instâncias mais potentes são as mais caras.

É importante observar que, o aumento da quantidade de recursos não implica necessariamente em um melhor desempenho do sistema. No entanto, esse aumento tem um grande impacto sobre o custo. Considere uma situação na qual uma aplicação necessita de mais poder de processamento e nenhuma adição na quantidade de memória, e a próxima classe de VM fornece um aumento nos dois recursos (CPU e memória). O cliente pode alterar de um tipo de VM para outro com mais recursos, pagar mais e, conseqüentemente, obter um desempenho melhor, mas o aumento do desempenho não está associado com o aumento de todos os recursos que compõem a VM. Talvez apenas um aumento na capacidade de processamento fosse suficiente.

Além disso, a Amazon EC2 utiliza um mecanismo chamado *Auto Scalling*¹ que monitora as condições do sistema e altera o número de instâncias de máquinas virtuais, caso seja necessário. Ele adiciona ou remove instâncias, sem alterar as configurações dos recursos que compõem a VM. Na Microsoft Azure², os clientes podem definir limites a partir dos quais a escalabilidade horizontal é aplicada. Essa pode ser aplicada baseada no percentual médio de uso da CPU ou baseada no número de mensagens em uma fila. Se o percentual médio de uso da CPU ou o número de mensagens na fila estão abaixo ou acima dos limites especificados, novas instâncias são criadas ou excluídas, ou máquinas virtuais são ligadas ou desligadas a partir de um conjunto de máquinas criadas previamente.

¹ <http://aws.amazon.com/autoscaling/>

² <https://msdn.microsoft.com/en-us/library/hh680945%28v=pandp.50%29.aspx>

Dessa forma, os provedores citados não alteram as configurações das VMs durante o tempo de execução. Eles alteram o número de instâncias respeitando as classes definidas por eles, onde eles adicionam ou removem VMs da mesma classe (os provedores chamam esse procedimento de escalabilidade horizontal) ou eles adicionam ou removem VMs de diferentes classes, por exemplo, um cliente contrata uma VM do tipo *Small* e por alguma razão uma nova VM do tipo *Large* é ligada (os provedores chamam esse procedimento de escalabilidade vertical). Dessa forma, é mais fácil para os provedores como a Amazon definir instâncias com configurações fixas ao invés de criar uma instância com configurações específicas para cada cliente. Além disso, o uso da escalabilidade horizontal torna-se mais rentável para os provedores. Na seção 5.5.2 são apresentados alguns experimentos que comprovam essa afirmação.

No trabalho apresentado nesta tese, acredita-se que um eficiente mecanismo de provisionamento de recursos deve aplicar tanto a escalabilidade horizontal quanto a vertical, de forma que os clientes possam definir as instâncias de VMs com configurações diferentes, ao invés de serem limitados pelas classes definidas pelos provedores. Os clientes devem ser capazes de definir a quantidade de núcleos virtuais, memória ou disco, por exemplo, que eles querem contratar e pagar um preço justo.

No entanto, a complexidade de encontrar uma alocação de recursos ótima é exponencial em grandes sistemas como *clusters*, grades e nuvens (ZHOU; HE, 2014). Uma vez que a demanda e atendimento às requisições pode ser dinâmica e incerta, várias estratégias para o gerenciamento da nuvem são disponibilizadas na literatura. A próxima Seção apresenta alguns dos trabalhos analisados. Baseado nesse estudo, um novo módulo para o gerenciamento de recursos é proposto e apresentado na Seção 5.4.

5.3 Trabalhos Relacionados

Há na literatura diversos trabalhos que analisam e propõem mecanismos para o gerenciamento de recursos em um ambiente em nuvem. Em Hussain *et al.* (2013), por exemplo, os autores apresentam um estudo sobre a alocação de recursos entre múltiplos sistemas HPC (*High Performance Computing*) como *cluster*, grade e nuvem. Os autores classificam as soluções existentes para HPC para prover uma análise mais aprofundada sobre as características das estratégias de alocação e de gerenciamento de recursos.

Sharkh *et al.* (2013) discutem vários fatores internos e externos que devem ser considerados na alocação de recursos. Segundo os autores, qualquer modelo de alocação deve considerar primeiramente os recursos computacionais disponíveis, bem como recursos de rede para refletir com precisão as demandas de serviços reais. Outro aspecto que deve ser considerado é o consumo de energia. Esses fatores são discutidos em detalhes e lacunas carentes de pesquisas são apontadas.

[Manvi e Shyam \(2014\)](#) realizaram um estudo que considerou técnicas para o gerenciamento de recursos como provisionamento, alocação, mapeamento e adaptação em uma IaaS. Eles afirmam que há muitas questões a serem abordadas no gerenciamento de recursos relacionados à flexibilidade, escalabilidade, adaptabilidade, personalização e reutilização dos recursos. Além disso, as métricas de desempenho, tais como atraso, *overhead* de largura de banda, *overhead* de computação, confiabilidade e segurança devem ser consideradas ao projetar um módulo de gerenciamento de recursos.

Em [Anuradha e Sumathi \(2014\)](#), várias estratégias de alocação e seus desafios são discutidos, abordando as vantagens e desvantagens de cada uma. O objetivo foi identificar um mecanismo eficiente para o gerenciamento de recursos em nuvem. Dentre as técnicas analisadas, os algoritmos genéticos proveram as melhores soluções.

Em [Iqbal et al. \(2010\)](#), os autores exploram a alocação adaptativa de recursos para aplicações *back-end mashup*. No protótipo desenvolvido, o *back-end* é um sistema de recursos intensivo responsável pela coleta e análise em tempo real de dados de serviços e aplicações externas. Baseado nessa análise, o protótipo adaptativamente aloca recursos para os *back-ends* para atender as requisições de monitoramento dos clientes. Entretanto, nos resultados apresentados, há apenas um tipo de máquina virtual com configurações fixas, a qual é atribuída de acordo com a demanda. Portanto, não há heterogeneidade nas configurações das VMs e nenhuma adaptação nas capacidades das mesmas durante o tempo de execução. Outra limitação é a falta de capacidade do sistema de desalocar instâncias quando o número de requisições reduz.

O trabalho apresentado em [Huu e Montagnat \(2010\)](#) propõe uma abordagem baseada no custo para a alocação de recursos para aplicações reais. Os autores definem quatro estratégias de alocação de infraestruturas virtuais e apresentam experimentos usando um protótipo. No protótipo analisado, uma máquina virtual é alocada em cada máquina física e todas as máquinas físicas possuem a mesma configuração. Dessa forma, considerando a grande heterogeneidade das aplicações em um ambiente em nuvem, os recursos disponíveis podem ser sobrecarregados ou ficarem ociosos e, conseqüentemente, podem prejudicar o cumprimento do SLA. Outra limitação é que o protótipo não é dinâmico, pois não é possível adicionar ou remover instâncias ou mesmo alterar as configurações das máquinas virtuais durante o tempo de execução.

[Inomata et al. \(2011\)](#) propõem uma arquitetura dinâmica para adicionar e/ou remover uma instância de acordo com a carga. Eles desenvolveram um protótipo no qual as informações (número de conexões TCP, disponibilidade de CPU e o tempo de recebimento dos dados) armazenadas no banco de dados são usadas para definir se uma instância será removida ou criada. No entanto, o protótipo utiliza apenas um tipo de instância e não há alterações na quantidade de recursos de uma VM durante o tempo de execução.

[Wu, Garg e Buyya \(2011\)](#) propõem algoritmos para a alocação de recursos para provedores de SaaS com o objetivo de minimizar os custos de infraestrutura e violações de SLA. De acordo com os autores, os algoritmos propostos são capazes de gerenciar a alteração dinâmica de

clientes, mapeando as requisições para os parâmetros de nível de infraestrutura, e manipular a heterogeneidade por meio da utilização de diferentes classes de VMs. Eles simularam um ambiente com três tipos de serviços e três tipos de VMs, no qual as VMs são atribuídas de acordo com o tipo de serviço requisitado pelo cliente seguindo uma estratégia de mapeamento predeterminada. Além disso, o ambiente simulado não altera a configuração dos recursos disponíveis. Os autores tentam maximizar o lucro reutilizando as VMs criadas, aplicando a abordagem de multialocação de recursos, o que pode causar a violação do SLA.

O trabalho apresentado em [Calheiros, Ranjan e Buyya \(2011\)](#) propõe uma abordagem adaptativa com foco na automação do gerenciamento de tarefas e na escalabilidade dos recursos a fim de garantir a QoS contratada pelos clientes. Para isso, os autores utilizam o modelo de desempenho analítico e informações sobre a carga imposta ao sistema, o que possibilita uma melhor atribuição dos recursos evitando um desempenho indesejado. O modelo de desempenho analítico permite prever o impacto gerado sobre a QoS na atribuição de um conjunto de recursos computacionais. Esse modelo ajuda o provedor a definir quais configurações de recursos são as mais adequadas para determinada carga e quando esses recursos devem ser incrementados ou reduzidos ([MENASCE et al., 2004](#)). Além disso, os autores utilizam uma política de balanceamento de carga simples para o provisionamento de recursos, na qual novas VMs são criadas com configurações fixas na máquina física hospedeira com o menor número de aplicações em execução. No entanto, não há alterações nos recursos que compõem as VMs.

Nos trabalhos analisados os autores apresentam diferentes mecanismos para o gerenciamento de recursos, mas em todos eles, não há alterações nas configurações dos recursos computacionais que compõem uma VM durante o tempo de execução, com impactos nos custos. Dessa forma, eles não aplicam a escalabilidade vertical, ao contrário do módulo proposto neste projeto que aplica os conceitos de escalabilidade vertical e horizontal. Portanto, considerando a escalabilidade em um ambiente em nuvem, no qual a demanda de serviços muda constantemente, um ambiente com configurações fixas das máquinas virtuais pode não ser o mais apropriado visando o uso eficiente dos recursos e a satisfação dos clientes com relação ao cumprimento do contrato. Dessa forma, a próxima Seção apresenta as características do módulo proposto.

5.4 ReMM – *Resource Management Module*

O módulo desenvolvido neste trabalho, chamado ReMM, considera a manipulação dos recursos computacionais durante o tempo de execução, respeitando as métricas de QoS definidas no SLA, a fim de atender qualquer demanda de requisições. Esse módulo aplica tanto a escalabilidade horizontal quanto a vertical de forma dinâmica com impacto sobre o preço. Dessa forma, ele tenta satisfazer o cliente, garantindo a QoS contratada a um preço justo, e o provedor, com o uso eficiente dos recursos disponíveis no sistema. A Figura 7 apresenta o funcionamento do módulo proposto. As principais funcionalidades são descritas a seguir:

Figura 7 – Funcionamento do ReMM

- **Controle de Admissão:** pode rejeitar ou aceitar uma requisição de serviço de acordo com as políticas do provedor. Esse módulo é responsável por organizar as requisições aceitas na Fila de Requisições atribuindo prioridades de acordo com as definições presentes no contrato, realizando, por exemplo, uma diferenciação de serviços e de clientes;
- **Fila de Requisições:** armazena as requisições de serviços até que elas sejam enviadas para execução pelo algoritmo de escalonamento utilizado pelo Escalonador;
- **Escalonador:** responsável por encaminhar as aplicações (serviços) para serem executadas nos recursos virtuais providos usando políticas de escalonamento;
- **ReMM:** responsável por prover os recursos computacionais contratados e por definir onde estes recursos serão hospedados. Além disso, se necessário, esse módulo altera a quantidade de recursos alocados de acordo com as informações recebidas do Monitor de Desempenho durante o tempo de execução com impacto sobre o custo. Pode aplicar tanto a escalabilidade vertical quanto a horizontal;

- **Monitor de Desempenho:** coleta informações sobre a execução das requisições e as envia para análises no ReMM. Considera uma margem de variação de desempenho definida no SLA (descrita em maiores detalhes na Seção 5.5.2) que mensura o desempenho do serviço em determinado instante de monitoramento;
- **Modelo de Negócio:** responsável por definir o modelo de negócio aplicado para cada cliente de acordo com os parâmetros definidos no SLA. Esse é um módulo dinâmico, pois considera que o contrato pode ser renegociado durante o tempo de execução do serviço. Por isso, esse módulo deve considerar todas as alterações dos recursos para definir o custo final do serviço.

Uma requisição do cliente será respondida por um provedor, o qual utiliza três camadas básicas para oferecer diferentes serviços: camada de aplicação (SaaS), camada de plataforma (PaaS) e camada de infraestrutura (IaaS). A camada de aplicação gerencia todas as aplicações que são oferecidas para os clientes. A camada de plataforma inclui as políticas de mapeamento e de escalonamento responsáveis por transferir à camada de infraestrutura as exigências feitas pelos clientes na definição do SLA e, dessa forma, alocar as máquinas virtuais para atendê-las. Por fim, a camada de infraestrutura controla a inicialização e remoção de instâncias, bem como a alteração dos recursos de uma VM de forma transparente ao cliente.

Inicialmente, cliente e provedor devem negociar um SLA, no qual as métricas de QoS bem como os detalhes do contrato, como por exemplo, níveis de segurança e o prazo para a execução do serviço, devem ser definidos (1). Após esse procedimento, diferentes clientes requisitam serviços a um provedor. O módulo Controle de Admissão é responsável por analisar a requisição e decidir se ela pode ser atendida ou não (2). Durante a sobrecarga do sistema, por exemplo, o provedor de serviços pode definir regras para decidir entre rejeitar novas requisições ou violar o SLA. A violação do SLA deve implicar em penalidades ao provedor. Se a requisição for aceita, ela será armazenada no módulo Fila de Requisições (3), onde ela receberá uma prioridade que definirá a sequência de execução (pode-se aplicar uma diferenciação de clientes, por exemplo).

Em seguida, o ReMM alocará os recursos virtuais baseado nas informações definidas no SLA (4), hospedando-os nos recursos físicos (5). Após a alocação dos recursos, o módulo Escalonador encaminha as requisições (6) para serem executadas nos recursos providos (7.a) (7.b) utilizando políticas de escalonamento. Em intervalos de tempo, o módulo Monitor de Desempenho coleta informações sobre as execuções das requisições (8) e as envia ao ReMM (9.a). Essas informações são comparadas com as informações definidas no SLA (9.b) e, caso alguma medição não esteja de acordo com o contrato, o ReMM ajusta dinamicamente a quantidade de recursos utilizando os algoritmos de escalabilidade horizontal e vertical, visando cumprir o SLA (10). Todas as alterações no sistema influenciam no custo definido no Modelo de Negócio (11).

As alterações realizadas pelo ReMM por meio dos seus algoritmos de escalabilidade seguem os pseudocódigos apresentados nas Figuras 8a e 8b. Por meio das experimentações apresentadas na Seção 5.5.1 definiu-se que o algoritmo Vertical altera apenas a quantidade de vCPUs das VMs, onde as alterações seguem um percentual de 100%, determinando que 8 vCPUs é o número máximo permitido por VM e 1 vCPU é o valor mínimo. No algoritmo Horizontal, o ReMM altera a quantidade de instâncias, onde 1 instância é a quantidade mínima permitida. Ambos os algoritmos implicam em alterações nos custos proporcionais a quantidade de recursos alterados. Além disso, há uma margem do SLA nos algoritmos que corresponde à uma taxa aceitável de variação do prazo de execução exigido por um cliente. Por exemplo, se um cliente exige um prazo de execução de 100 segundos com variação de SLA de 10%, significa que o tempo de execução do serviço pode variar entre 90 e 110 segundos.

<p style="text-align: center;">(a) Algoritmo Vertical</p> <pre> /*Analisa se o tempo obtido no monitoramento está de acordo com a Margem do SLA*/ if (tempoExecucao < margemInferior) /*Verifica se há pelos menos duas vCPUs na VM*/ if (quantidadeVCPUS > 1) /*Reduz quantidade de vCPUs pela metade*/ quantidadeVCPUS = quantidadeVCPUS / 2; /*Incrementa quantidade de alterações*/ quantidadeAlteracoes++; endif endif else if (tempoExecucao > margemSuperior) /*Verifica se não atingiu o máximo de vCPUs por VM*/ if (quantidadeVCPUS <= 8) /*Dobra quantidade de vCPUs*/ quantidadeVCPUS = quantidadeVCPUS * 2; /*Incrementa quantidade de alterações*/ quantidadeAlteracoes++; endif endif endif </pre>	<p style="text-align: center;">(b) Algoritmo Horizontal</p> <pre> /*Analisa se o tempo obtido no monitoramento está de acordo com a Margem do SLA*/ if (tempoExecucao < margemInferior) /*Verifica se há pelos menos duas instâncias*/ if (quantidadeInstancias > 1) /*Finaliza instância após execução das requisições alocadas*/ shutdown(instancia); /*Incrementa quantidade de alterações*/ quantidadeAlteracoes++; endif endif else if (tempoExecucao > margemSuperior) /*Aumenta quantidade de instâncias create (instancia); /*Incrementa quantidade de alterações*/ quantidadeAlteracoes++; endif endif </pre>
---	---

Figura 8 – Pseudocódigos dos algoritmos de escalabilidade disponíveis no ReMM

Essas definições foram baseadas nas análises apresentadas na próxima Seção que consideram experimentos executados por meio de técnicas de aferição e de simulação.

5.5 Análise de Desempenho no Provisionamento de Recursos em Nuvem

Nesta Seção são apresentados dois conjuntos de experimentos essenciais para as definições dos algoritmos de escalabilidade e para a validação do ReMM. No primeiro conjunto de experimentos (apresentado na Seção 5.5.1) é realizada uma avaliação em uma ambiente real para definir os recursos computacionais e a proporção que esses recursos devem ser alterados para atender as solicitações dos clientes em uma nuvem com diferentes cargas. Baseada nessa análise, um segundo conjunto de experimentos executados em um ambiente simulado foi projetado (Se-

ção 5.5.2), no qual são avaliados os impactos gerados sobre as variáveis de resposta na utilização do ReMM. O simulador CloudSim 3.0.3³ foi usado para modelar, implementar e executar os experimentos em um ambiente em nuvem com o ReMM. O CloudSim permite a modelagem e simulação de *data centers* em grande escala, de servidores virtualizados, de políticas para a alocação das máquinas virtuais nas máquinas físicas, além de políticas para o mapeamento e escalonamento entre as aplicações e as VMs, ou seja, todas as entidades necessárias no processo de gerenciamento de uma nuvem (CALHEIROS *et al.*, 2011).

Esses experimentos são importantes para validar o módulo proposto e definir os recursos que devem ser alterados e a proporção em que eles devem ser alterados pelos algoritmos do ReMM na tentativa de garantir o SLA e o uso eficiente dos recursos a um preço justo. Dessa forma, o ReMM pode ser utilizado em ambientes nos quais um determinado cliente considera segurança e desempenho como fatores importantes. Sendo assim, se necessário, o ReMM pode alterar a quantidade de recursos alocados na tentativa de contrapor a sobrecarga imposta pelos mecanismos de segurança exigidos pelo cliente, impactando nos custos.

Um planejamento fatorial completo foi utilizado em ambos os conjuntos de experimentos seguindo a metodologia apresentada por Jain (1991), na qual os fatores correspondem as características do ambiente analisado e os níveis são as possíveis variações que o ambiente pode apresentar.

Todos os experimentos foram executados 10 vezes, pois por meio das 10 repetições percebeu-se que os resultados não apresentaram grandes variações. Por isso, concluiu-se que 10 era um número razoável para a quantidade de repetições.

5.5.1 Primeiro Conjunto de Experimentos – Definição da Taxa de Alteração dos Recursos

Nesta Seção é apresentada uma análise para definir os recursos computacionais e a proporção que eles devem ser alterados para atender as requisições dos clientes. Para isso, foram realizados experimentos considerando diversas configurações de máquinas virtuais, executando cargas do tipo *System* e *CPU-Bound*, por meio dos seguintes *benchmarks*:

- **Apache**⁴: consiste em um *benchmark System-Bound* que mensura a quantidade de requisições por segundo que um sistema consegue atender, com 100 requisições realizadas simultaneamente em um total de 1 milhão.
- **Smallpt**⁵: consiste em um *benchmark CPU-Bound* que realiza a renderização de imagens utilizando o algoritmo Monte Carlo e mostra como variável de resposta o tempo de

³ <http://www.cloudbus.org/cloudsim/>

⁴ <https://openbenchmarking.org/test/pts/apache>

⁵ <https://openbenchmarking.org/test/pts/smallpt>

execução dado em segundos.

Nos experimentos com o *benchmark Apache* foram considerados cinco fatores com diferentes quantidades de níveis (Tabela 3). Esses fatores e níveis foram utilizados para a composição dos diversos cenários analisados. Cada nível superior segue uma porcentagem de 100% de aumento em relação ao nível inferior. Dessa forma, variou-se a quantidade de recursos providos como tamanho de disco, quantidade de memória RAM, número de vCPUs e de VMs.

Tabela 3 – Fatores e níveis para os experimentos com o *benchmark Apache*

Fatores	Níveis
Tamanho de Disco	8GB e 16GB
Tipo de Rede	Gigabit e Megabit
Quantidade de Memória RAM	512MB e 1024MB
Número de VMs	1, 2 e 4
Número de vCPUs	1, 2, 4 e 8

Após a análise das influências dos fatores, novos experimentos foram executados com o *benchmark Smallpt*, nos quais apenas os números de VMs e de vCPUs foram variados (Tabela 4) para analisar o comportamento do sistema com diferentes cargas.

Tabela 4 – Fatores e níveis para os experimentos com o *benchmark Smallpt*

Fatores	Níveis
Tamanho de Disco	8GB
Tipo de Rede	Gigabit
Quantidade de Memória RAM	512MB
Número de VMs	1, 2, 4 e 8
Número de vCPUs	1, 2, 4 e 8

Vale destacar que, os resultados com ambos os *benchmarks* mostram as variáveis de resposta com um intervalo de confiança de 95%. Além disso, para a execução dos experimentos foi utilizada uma máquina física responsável por hospedar as máquinas virtuais que executam a carga imposta. As configurações do ambiente são apresentadas na Tabela 5.

Tabela 5 – Especificação do ambiente

Máquina	Física	Virtual
Processador	Core 2 Quad 2.4 GHz	Core 2 Quad 2.4 GHz
Núcleos	4	Varia
Memória RAM	8GB	Varia
Disco	160GB	Varia
Rede	-	Varia
S.O.	Ubuntu Server 11.10	Ubuntu 10.04 kernel 2.6.32-33-server
Hipervisor	Xen 4.1	-

A forma que cada hipervisor associa os recursos físicos e virtuais é um fator importante que necessita ser considerado nos experimentos, uma vez que esse fator é a base para um provisionamento de recursos eficiente. Entretanto, essa associação varia de acordo com o

hipervisor utilizado. Para as análises propostas o hipervisor Xen foi utilizado, o qual utiliza o algoritmo *Credit Scheduler* (CHERKASOVA; GUPTA; VAHDAT, 2007). Esse algoritmo considera a quantidade total de vCPUs presentes no sistema e as divide entre as CPUs. Portanto, há momentos em que, de acordo com a configuração dos experimentos, um núcleo físico pode estar ocioso, sobrecarregado ou balanceado.

5.5.1.1 Resultados com o benchmark Apache

A Figura 9 é composta por 4 gráficos referentes à quantidade média de requisições por segundo que uma VM atendeu durante a execução dos experimentos. Para cada gráfico existe uma diferente combinação dos níveis que compõem os fatores definidos no planejamento dos experimentos. No entanto, mesmo com diferentes configurações, observa-se que o comportamento apresentado em todos os conjuntos de experimentos manteve-se praticamente o mesmo.

Figura 9 – Média de requisições atendidas por cada VM em um ambiente com 8GB de disco

De acordo com os gráficos, à medida que novas VMs foram adicionadas ao sistema, a concorrência por recursos computacionais físicos tornou-se maior, reduzindo, consequentemente, a quantidade média de requisições atendidas por cada VM. Isso pode ser visto na comparação entre os experimentos nos quais o número de vCPUs foi igual a 4 e variou-se o número de VMs entre 1, 2 e 4. Para esses experimentos, um aumento de 100% no número de VMs gerou uma

redução de aproximadamente 30% (1 para 2 VMs) e 46% (2 para 4 VMs) na quantidade de requisições atendidas. Para esses casos, nenhuma CPU ficou ociosa desde o início da execução dos experimentos.

Por outro lado, os experimentos com 1 VM com 1 vCPU possuem resultados semelhantes aos apresentados nos experimentos com 2 VMs com 1 vCPU. Isso é justificado devido à ocorrência de ociosidade dos recursos físicos durante a execução dos experimentos. Essa ociosidade ocorreu pelo fato do número de núcleos presentes nas VMs ser menor que o número de núcleos físicos disponíveis. O mesmo comportamento é observado nos experimentos com 1 VM com 2 vCPUs e 2 VMs com 2 vCPUs. Porém, no caso do experimento com 1 VM com 2 vCPUs, havia um total de 2 vCPUs para serem executadas em 4 núcleos físicos. Dessa forma, dois núcleos físicos ficaram ociosos. No caso do experimento com 2 VMs com 2 vCPUs, havia um total de 4 vCPUs para serem executadas em 4 núcleos físicos. Nesse caso, cada CPU recebeu uma vCPU e todas as vCPUs foram executadas paralelamente. Dessa forma, os resultados foram semelhantes, pois essa análise considera a quantidade média de requisições atendidas por segundo por cada VM. A Figura 10 exemplifica o comportamento descrito.

Figura 10 – Uso do processador

Ainda na Figura 9, nos experimentos com 4 VMs tem-se que, quanto maior o número de vCPUs, menor foi o número de requisições atendidas por segundo. Para esse conjunto de experimentos, o número de núcleos físicos foi um fator limitante, uma vez que, com o aumento do número de vCPUs, aumentou-se também a concorrência por esses recursos e, consequentemente, houve uma redução na quantidade de requisições atendidas. Esse comportamento também foi observado nos experimentos com 8 vCPUs e 1 e 2 VMs.

Por outro lado, nos experimentos com 1 e 2 VMs, a concorrência pelos recursos físicos foi menor resultando em um maior número de requisições atendidas. Para 1 VM tem-se que o aumento no número de vCPUs de 1 para 2 e posteriormente de 2 para 4 resultou em um aumento de aproximadamente 73% e 46%, respectivamente, na variável de resposta. Para 2 VMs, esse mesmo aumento no número de vCPUs resultou em um acréscimo de aproximadamente 75% e 6%, respectivamente.

O comportamento descrito através da Figura 9 é aplicado à Figura 11 onde a quantidade de disco, de 8GB para 16GB foi alterada. Dessa forma, analisando a alteração da quantidade de memória (de 512MB para 1GB), da rede (de Gigabit para Megabit) e de disco (de 8GB para 16GB) por meio da comparação entre os dados presentes nas Figuras 9 e 11, percebe-se que o comportamento do sistema manteve-se o mesmo. Dessa forma, considerando o planejamento dos

Figura 11 – Média de requisições atendidas por cada VM em um ambiente com 16GB de disco

experimentos e a carga utilizada (*System-Bound*), pode-se dizer que não há diferenças estatísticas quando se trata da combinação dos níveis desses fatores. Essa afirmação é comprovada na próxima Seção por meio da análise da influência dos fatores.

5.5.1.2 Influência dos Fatores

Neste trabalho foi utilizado o modelo fatorial completo que mede a variável de resposta utilizando a combinação entre todos os níveis dos fatores (JAIN, 1991). Assim, foram realizadas análises considerando a combinação dos níveis dos fatores Tamanho de Disco, Tipo de Rede, Quantidade de Memória RAM, Número de VMs e Número de vCPUS. Pelo fato do fator Número de vCPUs possuir 4 níveis, foi realizada uma análise combinando os níveis em 2 a 2 para determinar a influência de cada fator sobre a variável de resposta. Por outro lado, para o fator Número de VMs, que possui 3 níveis, foram considerados os níveis extremos, ou seja, 1 e 4 VMs.

Em todos os gráficos que compõem a Figura 12, a combinação dos fatores número de VMs e vCPUs foi a que mais influenciou nos resultados com 53,75% (Figura 12a), 43,34% (Figura 12b) e 46,43% (Figura 12c). Como dito anteriormente, à medida que novas VMs e vCPUs foram adicionadas no sistema, a concorrência por recursos físicos de processamento

tornou-se maior, impactando sobre a variável de resposta (quantidade de requisições atendidas por segundo).

Figura 12 – Influência dos fatores

Vale ressaltar que apenas na Figura 12a o número de vCPUs foi o segundo com mais influência (22,39%), seguido pelo número de VMs (16,76%). Por meio das combinações apresentadas nesse gráfico, foram registrados momentos nos quais o número de núcleos virtuais foi menor, igual e maior que o número de núcleos físicos. Por essa razão, os respectivos percentuais de influência foram obtidos.

Na Figura 12b, embora a influência do número de VMs (27,62%) seja maior que a influência do número de vCPUs (25,34%), verificou-se que a combinação entre esses dois fatores (43,34%) obteve uma redução no percentual de influência de, aproximadamente, 19% em relação à Figura 12a (53,75%). Esse percentual foi agregado às influências dos fatores número de VMs e número de vCPUs, que tiveram um aumento de, aproximadamente, 65% e 13%, respectivamente. Na Figura 12c, o terceiro fator de maior influência, ou seja, o número de vCPUs, com 11,83%, apresentou uma redução de, aproximadamente, 53% no percentual de influência em relação à Figura 12b (25,34%).

Dessa forma, verificou-se que, por meio da combinação apresentada na Figura 12b, a ociosidade dos recursos de processamento durante os experimentos foi menor quando comparada à Figura 12a, enquanto que a disputa por recursos de processamento também foi menor, quando comparada à Figura 12c. Portanto, dado o planejamento de experimentos e a carga utilizada, a combinação dos fatores e dos respectivos níveis apresentada na Figura 12b proporcionou o uso mais eficiente dos recursos disponíveis.

Além disso, verificou-se que as influências exercidas pelos fatores Tamanho de Disco, Tipo de Rede e Quantidade de Memória RAM foram mínimas como mencionado anteriormente, menores que 1%. Portanto, com base nessas análises, novos experimentos foram realizados considerando uma carga estritamente *CPU-Bound* baseada no *benchmark Smallpt*. A próxima Seção apresenta a análise dos resultados.

5.5.1.3 Resultados com o benchmark Smallpt

Nas Seções anteriores verificou-se que a quantidade de VMs e de núcleos virtuais foram os fatores que mais influenciaram sobre a variável de resposta considerando uma carga *System-Bound*. Os resultados motivaram a execução de novos experimentos considerando uma carga estritamente *CPU-Bound*. O objetivo é determinar se o mesmo comportamento do sistema é mantido com diferentes cargas. Para isso, a mesma especificação do ambiente de execução dos experimentos apresentada na Tabela 5 foi utilizada.

Na Figura 13, verificou-se que a medida que novas VMs foram adicionadas ao sistema, a concorrência por recursos computacionais tornou-se maior aumentando, consequentemente, o tempo de execução da carga imposta. Isso pode ser visto na comparação entre os experimentos com 4 vCPUs, na qual o aumento de 100% no número de VMs de 1 para 2, de 2 para 4 e de 4 para 8 gerou um aumento de aproximadamente 95%, 105% e 100% na variável de resposta, respectivamente.

Figura 13 – Tempo médio de execução em segundos

Uma análise interessante a se considerar consiste na comparação entre as colunas que estão em diagonal. Analisando a maior diagonal que compõe a Figura 13, ou seja, a diagonal composta pelos resultados dos experimentos com 1 VM com 8 vCPUs, 2 VMs com 4 vCPUs, 4 VMs com 2 vCPUs e 8 VMs com 1 vCPU, verificou-se que, apesar de todos os experimentos serem compostos por um total de 8 vCPUs, os experimentos com uma quantidade menor de VMs apresentaram resultados melhores. Em outras palavras, o experimento com 1 VM com 8 vCPUs obteve um tempo de execução inferior em aproximadamente 49% quando comparado com o experimento com 2 VMs e 4 vCPUs. Considerando a comparação entre 1 VM com 8 vCPUs e 4 VMs e 2 vCPUs, tem-se que o experimento com 1 VM obteve um tempo de execução menor que o experimento com 4 VMs em aproximadamente 78%. Para uma comparação entre os extremos da diagonal, ou seja, entre 1 VM com 8 vCPUs e 8 VMs com 1 vCPU, essa redução chega a

aproximadamente 92%. Dessa forma, considerando um ambiente com a mesma quantidade total de vCPUs, a combinação de 1 VM com 8 vCPUs apresentou um melhor desempenho do que a combinação de 8 VMs com 1 vCPU. O mesmo é válido para ambientes com 2 VMs com 4 vCPUs e 4 VMs com 2 vCPUs, nos quais a alteração de 4 para 2 VMs gerou uma redução de aproximadamente 57% no tempo de execução.

Além disso, vale ressaltar que o experimento com 1 VM e 1 vCPU possui resultado semelhante ao experimento com 2 VMs com 1 vCPU. Isso é justificado pela ocorrência de ociosidade dos recursos físicos nesses cenários durante a execução dos experimentos. Esse comportamento foi discutido na análise dos experimentos apresentados nas Figuras 9 e 11. O mesmo pode ser observado no experimento com 2 VMs com 1 vCPU que apresentou resultado semelhante ao experimento com 2 VMs com 2 vCPUs. Esse comportamento pode ser exemplificado por meio da Figura 10.

Por fim, verificou-se que o aumento no número de núcleos virtuais proporcionou uma redução no tempo de execução do sistema. Porém, isso foi válido até que o número de vCPUs atingiu o número de núcleos físicos. A partir desse ponto, verificou-se que o aumento do número de núcleos virtuais gerou uma maior concorrência por recursos físicos, prejudicando o desempenho do sistema. A Figura 14 apresenta uma comparação com relação ao aumento do número de vCPUs. Por meio dela, comprova-se o comportamento descrito. Pode-se verificar que, com o aumento de 100% no número de núcleos virtuais de 1 para 2 e de 2 para 4, obteve-se uma redução de aproximadamente 44% e 56%, respectivamente, no tempo de execução. A partir desse ponto, verificou-se que o aumento no número de vCPUs, de 8 para 16, de 16 para 32 e de 32 para 64 aumentou o tempo de execução em aproximadamente 97%, 129% e 100%, respectivamente.

Figura 14 – Tempo de execução vs número de vCPUs

As análises apresentadas nesse primeiro conjunto de experimentos visaram definir quais recursos computacionais e em que proporção eles devem ser alterados para atender as requisições dos clientes em um ambiente em nuvem. Diferentes cargas foram consideradas e em todas elas, as alterações na quantidade de memória, tamanho do disco e tipo de rede não exerceram

grande influência sobre a variável de resposta. Por outro lado, foi mais eficiente aumentar o número de vCPUs ao invés do número de VMs. Porém, essa taxa de aumento deve considerar o número disponível de núcleos físicos. Nos experimentos nos quais o número de núcleos virtuais excedeu a quantidade de núcleos físicos o desempenho do ambiente foi prejudicado, visto que a concorrência por recursos físicos foi maior.

Com base nos resultados, os algoritmos vertical e horizontal foram definidos. Esse algoritmos são validados por meio dos experimentos apresentados na próxima Seção, nos quais é analisado o comportamento do desempenho de um ambiente simulado com o ReMM.

5.5.2 Segundo Conjunto de Experimentos – Ambiente Simulado com o ReMM

Nesta Seção são apresentados alguns experimentos com e sem a presença do ReMM em ambientes simulados utilizando o simulador CloudSim. Nesses ambientes os recursos físicos e suas limitações não são considerados, isto é, os recursos físicos são considerados ilimitados e eles podem ser alocados aos clientes, com o consequente impacto no custo a ser pago. Os resultados mostram o impacto no custo e na métrica de QoS utilizada para a verificação de conformidade com o prazo de execução definido no SLA, com a alteração dos recursos disponíveis usando os algoritmos vertical e horizontal. A Tabela 6 mostra os fatores e níveis definidos.

Tabela 6 – Planejamento dos experimentos

Fatores	Níveis
Instância	<i>m3.medium</i> , <i>m3.large</i> , <i>m3.xlarge</i> , <i>m3.2xlarge</i>
Imagem	Pequena, Média, Grande
Ambiente	Comum, Vertical, Horizontal

Um cliente requisita a execução de renderização de uma determinada imagem que pode ser Pequena, Média ou Grande por meio do algoritmo de Monte Carlo. Esse tipo de requisição de serviço foi modelado baseado no *benchmark Smallpt*, descrito nas experimentações anteriores. As requisições podem ser executadas em ambientes com e sem o módulo ReMM, que são configurados com quatro tipos diferentes de instâncias, *m3.medium*, *m3.large*, *m3.xlarge* ou *m3.2xlarge*, modeladas baseadas nos tipos de instâncias M3 da Amazon⁶. Cada cliente possui a sua própria VM dedicada, ou seja, os recursos computacionais alocados não são compartilhados com outros clientes, e ele realiza 10 requisições de serviço durante cada experimento. Dessa forma, cada requisição é mapeada para a respectiva VM do cliente. A configuração inicial da VM varia de acordo com o experimento.

⁶ <http://aws.amazon.com/ec2/instance-types/>

5.5.2.1 Variáveis de Resposta

Duas variáveis de resposta foram utilizadas nas análises: o tempo médio de execução (TME) e o custo final de contratação dos recursos computacionais virtualizados. O TME mostra o tempo médio gasto para executar todas as requisições dos clientes. O custo final (em dólar) apresenta o preço que o cliente deve pagar dada a quantidade de recursos utilizados. Ele é definido de acordo com a instância contratada e muda conforme o ReMM altera a quantidade de recursos durante o tempo de execução.

Antes da execução das requisições, um prazo é definido no SLA, o qual é responsável por delimitar o tempo contratado por um cliente para executar as requisições, em segundos. Esse prazo possui uma taxa de variação, chamada Margem de SLA, a qual representa a taxa aceitável de variação do prazo de execução do serviço. A sua função é apresentada na Figura 15.

Figura 15 – Margem de variação do SLA

Nos experimentos, o cliente requisita um prazo de execução de 100 segundos, com uma Margem de SLA de 20%, ou seja, o tempo de execução de uma requisição pode variar entre 80 e 120 segundos. O módulo ReMM analisa o TME em períodos de tempo e, caso o resultado não esteja de acordo com o prazo contratado, ele altera a quantidade de recursos alocados para aquele cliente, impactando o custo final. Vale ressaltar que nos experimentos realizados, o tempo de rede gasto para a troca de mensagens entre o cliente e o provedor não foi considerado.

Na Figura 15, o TME no instante 2 é menor do que o tempo de execução exigido. Portanto, o módulo ReMM deve reduzir a quantidade de recursos alocados durante o tempo de execução de forma que a próxima requisição do cliente seja executada com uma quantidade de recursos diferente. Esta alteração é necessária, pois nesse exemplo a VM alocada para o cliente está utilizando mais recursos do que o necessário. Dessa forma, o cliente pagará mais e a sua requisição não será executada no tempo requisitado. No momento 3, há uma situação na qual

os recursos alocados não são suficientes para executar a requisição no tempo exigido, pois o TME está fora do limite contratado (Margem do SLA). Sendo assim, o ReMM deve aumentar a quantidade de recursos como uma tentativa de atingir a Margem do SLA.

Por meio dessas operações, o módulo proposto tenta garantir o tempo médio de execução requisitado, respeitando a Margem de SLA e alterando a configuração dos recursos, quando necessário, com uma alteração no preço. Essas alterações variam de acordo com o algoritmo utilizado pelo ReMM.

5.5.2.2 Análise dos Resultados

Os gráficos nas Figuras 16 e 17 apresentam os resultados para o planejamento de experimentos descrito na Seção anterior. Os algoritmos de escalabilidade vertical e horizontal foram aplicados e comparados com um ambiente Comum sem o ReMM, usando o tempo médio de execução em segundos e o custo final em dólares.

Figura 16 – Tempos Médios de Execuções

De acordo com os resultados obtidos nos experimentos com o ambiente Comum, os tempos médios de execuções aumentaram conforme o tamanho da imagem tornou-se maior. Por outro lado, conforme as instâncias tornaram-se mais potentes, houve reduções nos tempos médios de execuções das requisições.

A quantidade de recursos disponíveis nas instâncias utilizadas em um ambiente Comum foi insuficiente para atender praticamente todas as requisições dentro do prazo exigido no SLA (representado pelas linhas tracejadas), com exceção da instância *m3.xlarge* executando

Figura 17 – Custo por hora pago por um cliente

requisições com imagens Pequenas. Dessa forma, na tentativa de cumprir o SLA, o ReMM alterou as quantidades de VMs e de vCPUs.

Com exceção dos experimentos com o algoritmo horizontal e com instâncias *m3.medium* executando requisições com imagens Grandes e com instâncias *m3.2xlarge* executando imagens Pequenas, o ReMM cumpriu todos os contratos assinados (a Tabela 7 mostra todos os resultados obtidos).

No primeiro caso (instância *m3.medium*), a quantidade inicial de recursos disponíveis foi considerada muito pequena para atender as requisições com imagens Grandes, resultando em um tempo médio de execução de 180,60 segundos. Dessa forma, novas instâncias *m3.medium* foram adicionadas pelo ReMM, aumentando o custo e reduzindo o TME. Entretanto, a taxa de alteração dos recursos usada na escalabilidade horizontal não foi suficiente para garantir o SLA antes que as execuções das requisições terminassem. Dessa forma, 10 alterações na quantidade de instâncias foram realizadas.

Tabela 7 – Resultados dos experimentos

Exp.		TME (s)			Custo/Hora (\$)		
Instância	Imagem	Comum	Horizontal	Vertical	Comum	Horizontal	Vertical
<i>m3.medium</i>	Pequena	380	102,7	101,3	0,1082	0,4328	0,1352
	Média	720	115,2	100,7	0,1082	0,8656	0,1712
	Grande	1400	180,6	103,3	0,1082	1,0820	0,3424
<i>m3.large</i>	Pequena	190	97,4	97,4	0,2166	0,4332	0,2346
	Média	360	107,5	96	0,2166	0,8664	0,2706
	Grande	700	112	97,9	0,2166	1,7328	0,5412
<i>m3.xlarge</i>	Pequena	95	95	95	0,4387	0,4387	0,4387
	Média	180	92,3	92,3	0,4387	0,8774	0,4747
	Grande	350	104,5	93,3	0,4387	1,7548	0,9494
<i>m3.2xlarge</i>	Pequena	47,5	47,5	90,5	0,8775	0,8775	0,8415
	Média	90	90	90	0,8775	0,8775	0,8775
	Grande	175	89,7	89,7	0,8775	1,7550	1,7550

No segundo caso (instância *m3.2xlarge*), a quantidade de recursos disponíveis para a execução de requisições com imagens Pequenas foi excessiva (8 vCPUs). Portanto, o ReMM tentou, sem sucesso, reduzir o número de instâncias *m3.2xlarge*, pois o número mínimo de instâncias havia sido atingido.

Nos experimentos com instância *m3.medium*, as alterações realizadas pelo algoritmo vertical no número de vCPUs geraram um aumento de aproximadamente 25%, 58% e 216% nos custos finais durante a execução das requisições com imagens Pequenas, Médias e Grandes, respectivamente. Entretanto, para as mesmas requisições, as alterações na quantidade de vCPUs reduziram os tempos médios de execuções em aproximadamente 73%, 86% e 93%, respectivamente, em relação ao ambiente Comum, tornando essas alterações essenciais para o cumprimento do SLA. Considerando o algoritmo horizontal, as alterações na quantidade de instâncias *m3.medium* proporcionaram reduções nos TMEs nas execuções das requisições com imagens Pequenas, Médias e Pesadas, aproximadamente 73%, 84% e 87%, respectivamente. Entretanto, essas alterações impactaram os custos finais gerando aumentos de, aproximadamente, 300%, 700% and 900%, respectivamente.

As alterações aplicadas pelos algoritmos vertical e horizontal nas instâncias *m3.large* geraram resultados similares quando ambos foram comparados com os resultados de um ambiente Comum. O algoritmo vertical aplicou um aumento no número de vCPUs de 100% na execução de requisições com imagens Pequenas, 300% para Médias e 700% para Grandes (Tabela 8). Esses aumentos de vCPUs resultaram em reduções dos tempos médios de execuções de aproximadamente 49% (imagem Pequena), 73% (Média) e 86% (Grande). Por outro lado, o uso do algoritmo horizontal resultou em acréscimos nos custos finais por hora de aproximadamente 100%, 300% e 700%, respectivamente (Figura 17b).

Na instância *m3.xlarge*, a quantidade de recursos disponíveis para executar requisições com imagens Pequenas foi considerada suficiente para todos os experimentos. Dessa forma, o ReMM não alterou a quantidade de recursos e de instâncias *m3.xlarge*, resultando em tempos

Tabela 8 – Quantidade de recursos utilizados

Exp.		Nº vCPUs Utilizadas			Nº Instâncias Utilizadas		
Instância	Imagem	Comum	Horizontal	Vertical	Comum	Horizontal	Vertical
<i>m3.medium</i>	Pequena	1	4	4	1	4	1
	Média	1	8	8	1	8	1
	Grande	1	10	16	1	10	2
<i>m3.large</i>	Pequena	2	4	4	1	2	1
	Média	2	8	8	1	4	1
	Grande	2	16	16	1	8	2
<i>m3.xlarge</i>	Pequena	4	4	4	1	1	1
	Média	4	8	8	1	2	1
	Grande	4	16	16	1	4	2
<i>m3.2xlarge</i>	Pequena	8	8	4	1	1	1
	Média	8	8	8	1	1	1
	Grande	8	16	16	1	2	2

médios de execuções e custos finais iguais. Por outro lado, o aumento no tamanho das imagens exigiu um acréscimo na quantidade de recursos nos outros experimentos, 100% para imagens Médias e 300% para Grandes, em ambos os ambientes Vertical e Horizontal. No entanto, embora os percentuais de aumento de recursos sejam iguais para os dois ambientes, a forma que cada algoritmo aplicou a alteração dos recursos implicou em custos diferentes (17c).

No primeiro caso (ambiente Vertical), o aumento de vCPUs proporcionou acréscimos nos custos finais de aproximadamente 8% (imagem Média) e 116% (Grande), e reduções nos TMEs de 49% para requisições com imagens Médias e 73% com Grandes. No segundo caso (ambiente Horizontal), os aumentos nos custos finais foram de aproximadamente 100% (Média) e 300% (Grande), enquanto que as reduções nos tempos médios de execuções foram de 49% com imagens Médias e de 70% com Grandes.

Considerando os resultados dos experimentos nos quais a instância *m3.2xlarge* foi utilizada, o TME das requisições com imagens Pequenas no ambiente Comum foi menor quando comparado com um ambiente com o algoritmo Vertical, 47,5 e 90,5 segundos, respectivamente. Embora o TME com o ambiente Comum tenha sido aproximadamente 47% menor, o SLA não foi cumprido. Por essa razão, o ReMM reduziu a quantidade de vCPUs em 50% com o algoritmo Vertical (de 8 para 4), além do custo final em 4%, garantindo o cumprimento do SLA. Dessa forma, a melhor alternativa para a execução de requisições de renderizações de imagens Pequenas, em aproximadamente 100 segundos, é a alocação de uma instância *m3.xlarge* ao invés de uma *m3.2xlarge*, pois nenhuma alteração foi necessária para a execução de requisições com imagens Pequenas em uma instância *m3.xlarge*. Assim, há uma redução no custo final de aproximadamente 50%, de \$ 0,8775 (*m3.2xlarge*) para \$ 0,4387 (*m3.xlarge*).

Para a execução de requisições com imagens Médias na instância *m3.2xlarge*, a quantidade de recursos disponíveis foi suficiente em todos os ambientes. Dessa forma, nenhuma alteração foi realizada na quantidade de vCPUs e de instâncias, resultando nos mesmos valores para os TMEs e para os custos finais (Figuras 16 e 17d). Por outro lado, a alteração do tamanho

da imagem, de Média para Grande, exigiu um aumento de 100% dos recursos nos ambientes com escalabilidade vertical e horizontal, resultando em uma redução de aproximadamente 49% nos tempos médios de execuções e em aumentos de 100% nos custos finais.

No ReMM, a alteração no custo usando a escalabilidade vertical segue a mesma proporção de alteração da quantidade de vCPUs, enquanto que na escalabilidade horizontal, tipo de escalabilidade que a grande maioria de provedores utilizam, a alteração no custo segue a mesma proporção de alteração do número de instâncias e, conseqüentemente, de todos os recursos. Embora o número de vCPUs em ambos os tipos de escalabilidade seja o mesmo, salvas as exceções discutidas anteriormente, o número de instâncias alocadas tem um forte impacto sobre as variáveis de resposta. Por essa razão, o custo na escalabilidade horizontal foi maior em todos os casos nos quais a quantidade de recursos foi alterada. A Tabela 9 apresenta análises comparativas considerando os ambientes com os algoritmos vertical e horizontal em relação ao ambiente Comum, com base nos tempos médios de execuções e nos custos finais obtidos nos experimentos.

Tabela 9 – Comparativo dos percentuais de aumento e de redução em relação ao ambiente Comum

Instâncias	% de Redução do TME		% de Aumento do Custo Final		Imagem
	Horizontal	Vertical	Horizontal	Vertical	
<i>m3.medium</i>	73	73	300	25	Pequena
	84	86	700	58	Média
	87	93	900	216	Grande
<i>m3.large</i>	49	49	100	8	Pequena
	70	73	300	25	Média
	84	86	700	150	Grande
<i>m3.xlarge</i>	0	0	0	0	Pequena
	49	49	100	8	Média
	70	73	300	116	Grande
<i>m3.2xlarge</i>	0	Aumento de 90	0	Redução de 4	Pequena
	0	0	0	0	Média
	49	49	100	100	Grande

5.6 Considerações Finais

A utilização de mecanismos de segurança acarreta sobrecarga nos serviços prestados em uma nuvem, o que pode causar a quebra do SLA e o uso inadequado dos recursos computacionais. Uma alternativa para contrapor essa sobrecarga é alterar a quantidade de recursos alocados com impacto sobre o custo do serviço. Porém, quando se trata do gerenciamento de recursos em nuvem, são necessários mecanismos eficientes capazes de alocar os recursos necessários e garantir o cumprimento do SLA a um preço justo de acordo com a demanda de serviços.

Neste capítulo foi apresentado um novo módulo para o gerenciamento de recursos em uma nuvem. O ReMM é um módulo dinâmico e autogerenciável que visa garantir as exigências feitas por um cliente na definição do SLA e o uso eficiente dos recursos disponíveis. Para isso,

sempre que necessário, ele altera os recursos *on-the-fly*, usando tanto a escalabilidade horizontal quanto a vertical com impacto sobre o custo.

Os recursos computacionais exigidos pelos clientes são alocados nos recursos físicos e as requisições executadas. Se a variável de resposta não estiver de acordo com as cláusulas do contrato, o ReMM pode alterar a configuração dos recursos baseado nas informações coletadas pelo Monitor de Desempenho, o que implica em reajustes no custo do serviço durante o tempo de execução. Assim, é possível que um cliente exija níveis mais rígidos de segurança sem perda de desempenho do serviço.

O próximo Capítulo apresenta análises de um serviço em nuvem, nas quais é possível confrontar o *overhead* gerado pelos mecanismos de segurança exigidos pelos clientes com o ReMM e avaliar os impactos sobre as variáveis de resposta definidas. Com base nos resultados, modelos de negócio de diferentes ambientes foram definidos, os quais consideram a sobrecarga imposta pelos mecanismos de segurança, o desempenho e os custos do serviço.

AVALIAÇÃO DE DESEMPENHO E DEFINIÇÃO DE MODELOS DE NEGÓCIO PARA AMBIENTES EM NUVEM

6.1 Considerações Iniciais

Visto que a presença de mecanismos de segurança ocasiona impacto negativo no desempenho dos serviços executados em uma nuvem, como pôde ser observado nos trabalhos analisados no Capítulo 4, torna-se interessante confrontar essa sobrecarga com a alteração de recursos fornecida pelo módulo de gerenciamento proposto no Capítulo 5. No entanto, as alterações dos recursos tem impacto sobre o custo pago pelo cliente e, consequentemente, no modelo de negócio.

Este Capítulo apresenta uma avaliação de desempenho que considera a sobrecarga imposta por diferentes mecanismos de segurança em um ambiente simulado. Para isso, o módulo ReMM foi inserido na arquitetura CloudSim-BEQoS (Bursty, Energy, QoS) (PARDO, 2012) (CENTURION, 2015), descrita na Seção 6.2. Essa inserção possibilitou a modelagem de um ambiente em nuvem bem próximo de um ambiente real, no qual diversos clientes requisitam serviços a um provedor, proporcionando taxas de chegada de requisições diferentes. Dessa forma, a Seção 6.3 apresenta o planejamento dos experimentos analisados na Seção 6.4, utilizando dois cenários baseados nos trabalhos apresentados em Popa *et al.* (2011) e Casola *et al.* (2011). Por meio deles, pode-se analisar o impacto sobre o desempenho de um serviço em nuvem na utilização de mecanismos de segurança, bem como a influência do ReMM sobre as variáveis de resposta na tentativa de contrapor o *overhead* gerado, garantir o SLA e o uso eficiente dos recursos a um preço justo. Com base nas análises, modelos de negócio que caracterizam os dois cenários foram definidos na Seção 6.4.3.

6.2 Inserção do ReMM na Arquitetura CloudSim-BEQoS

A CloudSim-BEQoS é uma arquitetura desenvolvida em projetos de doutorado do Laboratório de Sistemas Distribuídos e Programação Concorrente (LaSDPC) utilizando a API do simulador CloudSim (PARDO, 2012) (CENTURION, 2015). Esses projetos visam a construção de um *Broker* com políticas que garantam QoS e a eficiência energética, bem como proporcionar análises de desempenho de serviços na utilização de diferentes rajadas de requisições como cargas. A união de todas essas características deu origem a uma arquitetura complementar aos módulos disponíveis no CloudSim, contribuindo não apenas com as pesquisas desenvolvidas no LaSDPC, mas também com outros grupos de pesquisa voltados ao planejamento de capacidade, modelos de desempenho e avaliação de desempenho de sistemas computacionais em nuvem.

A CloudSim-BEQoS foi projetada visando à possibilidade de avaliar a influência das rajadas no desempenho dos serviços; fornecer mecanismos ao *Broker* para que ele atue como um agente de intermediação, de modo a garantir qualidade de serviço aos clientes da nuvem e; disponibilizar políticas de escalonamento, por meio da diferenciação de serviços baseada em prioridades, a fim de evitar picos de demandas de energia.

A arquitetura é composta por três camadas: a camada cliente, a de gerenciamento e a de infraestrutura. Na camada cliente encontram-se as entidades clientes responsáveis pelo envio concorrente de requisições à nuvem; os diferentes modos de submissão de requisições implementados, que podem ocorrer de forma individual ou em grupo; e os modelos de cargas de trabalho com e sem rajadas. A camada de gerenciamento é responsável pelas tarefas de escalonamento, alocação das requisições nas máquinas virtuais e pelo monitoramento tanto das requisições quanto dos recursos. Por fim, na camada de infraestrutura encontram-se os recursos físicos dos *data centers* que hospedam as máquinas virtuais.

O processo inicia com o cliente enviando uma requisição ou um grupo de requisições para serem processadas na nuvem. Essas requisições são recebidas pelo *Broker*, que realiza algumas operações antes de encaminhá-las aos recursos computacionais da nuvem para serem processadas. Inicialmente, o *Broker* sinaliza ao monitor de requisições a chegada de uma ou um grupo de requisições e também a demanda de serviço (em Milhões de Instruções – MI) requerida por essas requisições. Essas informações são utilizadas pelo monitor de requisições para calcular as taxas de chegadas de requisições e as taxas de chegadas de demandas de serviços, em intervalos de tempo, durante a simulação. Logo após, o *Broker* se comunica com o monitor de recursos para consultar o percentual de consumo dos recursos e com o monitor de energia para verificar o percentual de consumo, em *watts*, também do *data center*. Baseado na carga atual dos recursos e das políticas de escalonamento adotadas (considerando requisições individuais ou em grupos, com ou sem critérios de QoS e/ou diferenciação de serviços) o *Broker* seleciona máquinas virtuais do *data center* e encaminha as requisições para essas VMs. A requisição após ser processada é encaminhada ao *Broker*, para que este envie a resposta completa ao cliente

(PARDO, 2012) (CENTURION, 2015). O diagrama de sequência apresentado na Figura 18 apresenta o comportamento descrito.

Existem três modos de submissão das requisições: vetor de requisições, requisições em tempo real e grupo de requisições. A demanda de serviço (em MI) imposta pela requisição é definida durante a sua criação, podendo ser modelada sem ou com rajadas. Da mesma forma, o processo de chegada das requisições pode se apresentar ou não em forma de rajadas, nas quais os intervalos de tempo entre o envio de duas requisições sucessivas de um mesmo cliente são gerados durante a simulação, podendo ser logo após o envio da primeira requisição ou após o recebimento completo da requisição prévia.

A carga de trabalho pode ser modelada para representar situações sem e com rajadas no processo de chegada das requisições e/ou nas demandas de serviços impostas aos recursos da nuvem. Para situações com rajadas, a carga de trabalho é modelada com base nas metodologias encontradas na literatura (LU *et al.*, 2010) (MI *et al.*, 2010) (PACHECO-SANCHEZ *et al.*, 2011) (CASALE *et al.*, 2012) (LU *et al.*, 2013) (YIN *et al.*, 2014). Na maioria desses trabalhos, as cargas de trabalho que se apresentam em forma de rajadas são modeladas como um processo estocástico, usando a classe MAP, para capturar as características e propriedades deste tipo de carga, como alta variabilidade e chegadas de requisições em forma condensada em curtos períodos ao longo do tempo.

Dessa forma, o modelo de carga de trabalho adotado e implementado na CloudSim-BEQoS é construído usando três processos MAPs com dois estados para modelar rajadas de origens distintas (no processo de chegada e/ou nas demandas de serviços) e de diferentes níveis de intensidades e variabilidades. As rajadas originadas no processo de chegada das requisições estão relacionadas aos intervalos de tempo entre chegadas de requisições (representadas pelos *think times* entre o envio de duas requisições sucessivas), enquanto que, as rajadas originadas na demanda de serviço estão associadas às demandas impostas aos recursos do sistema, como memória, CPU e servidores, por exemplo, resultante das solicitações de serviços. Esses três MAPs, nomeados de MAP_{IAT} (*MAP for Inter-Arrival Time*), MAP_{SD} (*MAP for Service Demand*) e MAP_{sync} (*MAP Synchronized*), são usados para representar diferentes modelos de carga de trabalho. Uma descrição detalhada sobre o CloudSim-BEQoS, bem como uma análise do impacto de diferentes modelos de cargas implementados com e sem rajadas é apresentada em Centurion (2015).

A inserção do ReMM na arquitetura CloudSim-BEQoS consiste da criação de um monitor de desempenho com as mesmas características descritas no Capítulo 5, no qual o tempo de execução das requisições é coletado em intervalo de tempos. Essa informação é enviada ao ReMM que aplica os algoritmos de escalabilidade, quando necessário, na tentativa de atingir o acordo firmado e o uso eficiente dos recursos. Dessa forma, essa inserção possibilita que as configurações das máquinas virtuais e/ou o número de instâncias alocadas sejam alteradas durante o tempo de execução de acordo com a demanda de serviços com ou sem a presença de

Figura 18 – Diagrama de sequência para o processamento de uma requisição de serviço (CENTURION, 2015)

Figura 19 – Operações realizadas pelo ReMM no CloudSim-BEQoS

rajadas, impactando no custo final pago pelos clientes. Os diagramas de sequência apresentados nas Figuras 18 e 19 apresentam as interações realizadas no processo de envio de uma requisição e de análise do tempo que determina se os recursos devem ser alterados ou não, com impacto no

custo.

Além do ReMM, foi inserido um módulo de tarifação, não disponível na versão do simulador CloudSim 3.0.3, que permite a cobrança dos recursos computacionais de acordo com a utilização dos mesmos. Como mencionado anteriormente no Capítulo 5, os provedores como Amazon EC2 e Microsoft Azure cobram por suas instâncias de acordo com a classe. Dessa forma, o preço é definido pelo conjunto de recursos computacionais utilizados. Essa abordagem não permite definir o preço unitário de um recurso como vCPU, memória ou disco.

Por essa razão, foi introduzido no simulador CloudSim-BEQoS um módulo para a definição de preços baseada na quantidade de recursos utilizados na composição de uma VM e no número de instâncias. Dessa forma foi possível definir um modelo de negócio diferente do aplicado pela maioria dos provedores, que cobram pelos serviços de acordo com a classe de instância utilizada, oferecendo ou não a opção de escalabilidade horizontal dos recursos.

O módulo de tarifação foi desenvolvido com base nos cálculos de custo aplicados pela *Right Scale*¹. O *Right Scale* fornece o *Plan for Cloud*, um simulador que permite ao usuário analisar hipoteticamente as opções de compras com diferentes configurações de recursos em diferentes nuvens. Criou-se assim um tabelamento de preços que permite definir o custo de alocação de cada recurso que compõe uma VM separadamente (Equação 6.1), ao invés de em conjunto, bem como o custo gasto para a alocação de todas as instâncias que compõem o ambiente em nuvem (Equação 6.2).

$$Custo = (Quantidade_{vCPU} \times Preço_{vCPU}) + (Quantidade_{Memória} \times Preço_{Memória}) + (Quantidade_{Disco} \times Preço_{Disco}) \quad (6.1)$$

$$\sum_i^{totalVMs} Custov_{Mi} \quad (6.2)$$

Com as inserções dos módulos ReMM e de tarifação, o CloudSim-BEQoS tornou-se mais dinâmico e autogerenciável, exonerando clientes e provedores das tarefas como a alteração da quantidade de recursos para atender à demanda de serviço dentro das exigências definidas no SLA. As análises apresentadas nas próximas Seções mostram os cenários utilizados com diferentes configurações e o impacto nas variáveis de resposta com a utilização de mecanismos de segurança e do ReMM.

6.3 Planejamento dos Experimentos

O objetivo dos experimentos propostos nesta Seção é analisar o comportamento de ambientes simulados na aplicação de diferentes mecanismos de segurança e o impacto sobre as

¹ <http://www.planforcloud.com/>

variáveis de resposta na utilização do CloudSim-BEQoS com o módulo ReMM, na tentativa de contrapor a sobrecarga introduzida pelos mecanismos de segurança.

Para a análise proposta foram desenvolvidos dois cenários compostos por 3 entidades: usuários, cliente e provedor (Figura 20). O cliente contrata a infraestrutura de um provedor para alocar o seu serviço. Esse serviço fornece o armazenamento e compartilhamento de dados (por exemplo, músicas e filmes) entre diversos usuários. No entanto, vários aspectos críticos de segurança estão envolvidos nesse ambiente, tais como ameaças ao acesso e ao armazenamento seguro dos dados, conforme apresentado nos Capítulos 3 e 4.

Figura 20 – Relação entre provedores, clientes e usuários

Sendo assim, nos cenários abordados o usuário pode escolher entre ter um serviço com ou sem segurança, o qual deve ser executado em uma determinada VM de forma transparente dentro do prazo definido no SLA. Porém, a utilização de mecanismos de segurança acarreta sobrecarga no serviço, a qual será confrontada pelo ReMM. Para isso, o ReMM deve analisar o tempo de execução da requisição e compará-lo com o prazo definido no SLA, que nesse caso é de 100 segundos, com margem de variação do SLA de 15%. Caso o resultado não esteja de acordo com o desempenho contratado, a quantidade de recursos alocados deve ser alterada de acordo com o algoritmo de escalabilidade utilizado, impactando nas variáveis de resposta.

Serão utilizados três tipos de ambiente para a execução dos experimentos (Comum, Vertical e Horizontal), os quais aplicam modelos de negócio diferentes. O ambiente Comum consiste de um ambiente não-escalável utilizado em alguns provedores, no qual as requisições de serviço devem ser executadas nas VMs alocadas sem alterações na quantidade de recursos

durante o tempo de execução. Dessa forma, o provisionamento de recursos é estático e os custos permanecem inalterados.

O ambiente Horizontal aplica um modelo de negócio dinâmico utilizado por alguns provedores como a Amazon (*Auto Scaling*²). Esse modelo permite a alteração da quantidade de recursos alocados durante o tempo de execução. Nesse caso, a escalabilidade horizontal é aplicada, gerando alterações nos custos proporcionais ao número de instâncias alocadas. Consequentemente, os custos tendem a ser mais expressivos.

Por fim, o ambiente Vertical aplica um modelo de negócio que considera a alteração de recursos específicos que compõem uma VM. Nos experimentos foram consideradas alterações na quantidade de vCPUs de uma VM, as quais proporcionam alterações mais brandas nos custos, uma vez que elas são proporcionais ao número de vCPUs alocadas. Ou seja, uma instância possui um custo inicial, o qual pode sofrer alterações caso o número de núcleos virtuais seja alterado.

Para a composição da sobrecarga imposta pelos mecanismos de segurança serão utilizados os trabalhos apresentados em [Popa et al. \(2011\)](#) e [Casola et al. \(2011\)](#), descritos no Capítulo 4. Além disso, foram utilizadas rajadas *MAP_{IAT}* para o envio de requisições individuais dos usuários em tempo real implementadas no CloudSim-BEQoS, o que possibilitou a aplicação de diferentes cargas que variam de acordo com o cenário.

As requisições em tempo real são dinamicamente criadas e enviadas durante a simulação. Esse tipo de submissão possibilita a criação de simulações mais realísticas, pois torna o comportamento do usuário mais próximo do que se espera em um ambiente real. Essas requisições são enviadas por meio do processo *MAP_{IAT}*, o qual implementa rajadas originadas no processo de chegada das requisições, por meio da geração de uma sequência de *think times*, os quais correspondem ao tempo de espera entre o envio de uma requisição e outra.

Nos cenários 1 e 2 foram utilizadas rajadas com taxas de chegada de requisições diferentes: com *think times* aguardando resposta e com *think times* após o envio da requisição prévia. Ambas correspondem ao tempo de espera entre o envio de duas requisições sucessivas pelo mesmo usuário. No primeiro caso, após receber a resposta de sua requisição, o usuário aguarda um intervalo de tempo (*think time*) e então envia a próxima requisição à nuvem. Por outro lado, no segundo caso o *think time* começa a ser considerado imediatamente após o envio de uma requisição, e dessa forma, o tempo entre o envio de uma requisição e outra é menor, aumentando a concorrência por recursos. Dessa forma, tem-se a simulação de dois cenários distintos para a avaliação proposta: Cenário 1 e Cenário 2.

O Cenário 1 é baseado no trabalho apresentado em [Popa et al. \(2011\)](#), no qual os mecanismos de segurança geram uma sobrecarga sobre o desempenho do serviço de 17%. Para a representação das requisições são utilizadas rajadas originadas na chegada dessas requisições com *think times* aguardando a resposta da requisição prévia.

² <https://aws.amazon.com/pt/autoscaling/>

No Cenário 2 é utilizada a contramedida de segurança proposta por Casola *et al.* (2011), a qual impõe uma sobrecarga de 60% ao desempenho do serviço. Para a representação das requisições dos usuários são utilizadas rajadas com *think times* logo após o envio da requisição prévia. Portanto, a definição dos dois cenários serve para análises dos ambientes simulados utilizando o CloudSim-BEQoS com o ReMM na tentativa de contrapor as sobrecargas impostas pelas contramedidas de segurança. Sendo assim, nenhum tipo de comparação é estabelecida entre os Cenários 1 e 2. A Tabela 10 apresenta as especificações de cada cenário analisado.

Tabela 10 – Especificações dos cenários analisados

Propriedades	Cenário 1	Cenário 2
Taxa de chegada das requisições	<i>Think times</i> aguardando resposta	<i>Think times</i> após envio da requisição prévia
<i>Overhead</i> de Segurança	17% (POPA <i>et al.</i> , 2011)	60% (CASOLA <i>et al.</i> , 2011)

Para a execução dos experimentos nos dois cenários, um provedor composto por *data centers* foi configurado, os quais hospedam as máquinas virtuais que executam as requisições de serviço. Considerou-se que os recursos físicos são ilimitados, e dessa forma, todas as requisições são aceitas e executadas pelas VMs. Além disso, não há diferenciação de usuários por meio de prioridades. As configurações dos *data centers* são apresentadas na Tabela 11. Vale destacar que no CloudSim-BEQoS a capacidade do processamento é dada em MIPS (Milhões de Instruções por Segundo).

Tabela 11 – Especificação das máquinas físicas presentes nos *data centers*

Processador	Intel Xeon 8 cores 2.6 Ghz
Capacidade de Processamento	10.000 MIPS/core
Memória Principal	32GB
Disco	2TB
S.O.	Ubuntu Server 11.10
Virtualizador	Xen 4.1

As máquinas virtuais utilizadas são as mesmas utilizadas na validação do ReMM. Essas instâncias foram modeladas baseadas nos tipos de instâncias M3 da Amazon³. A Tabela 12 apresenta as configurações de cada VM, bem como o custo inicial de cada instância. Vale lembrar que o custo pode variar de acordo com o ambiente utilizado, onde o ReMM pode alterar as configurações de uma determinada VM ou o número de instâncias durante o tempo de execução.

Tabela 12 – Especificação das instâncias

Instância	Núcleo Virtual	Memória Principal (GB)	Disco SSD	Custo Inicial/Hora (\$)
<i>m3.medium</i>	1	3,75	1 x 4	\$ 0,1082
<i>m3.large</i>	2	7,5	1 x 32	\$ 0,2166
<i>m3.xlarge</i>	4	15	2 x 40	\$ 0,4387
<i>m3.2xlarge</i>	8	30	2 x 80	\$ 0,8775

Os experimentos apresentados na próxima Seção foram conduzidos com o objetivo de avaliar métricas relacionadas ao desempenho do serviço prestado por um cliente utilizando a

³ <https://aws.amazon.com/pt/ec2/instance-types/>

infraestrutura de um provedor, com ou sem a utilização de mecanismos de segurança. Desta forma, foram consideradas as seguintes variáveis de resposta:

- **Tempo Médio de Execução (TME):** média do tempo gasto na execução de uma requisição. Nos experimentos foi definido que cada requisição deve ser executada em aproximadamente 100 segundos com variação de até 15%. Considerando que o objetivo deste projeto é confrontar segurança e desempenho, optou-se por utilizar o TME ao invés do TMR (Tempo Médio de Resposta), dado que o *overhead* imposto pelos mecanismos de segurança exerce impacto direto sobre a execução de um serviço. O TMR, além de utilizar o TME, é composto pelo tempo de tráfego dos dados, o qual não será abordado nesses experimentos.
- **Custo Médio Final por VM (\$/Hora):** corresponde a média do custo final na utilização de uma VM considerando as possíveis alterações dos recursos alocados.
- **Custo Médio Final do Ambiente (\$/Hora):** representa a média do custo para a alocação de todas as VMs utilizadas para atender a demanda de requisições durante o tempo de observação do ambiente.
- **Quantidade de Requisições Atendidas:** taxa média de requisições atendidas por um ambiente em nuvem durante o tempo de observação.
- **Quantidade de Recursos Utilizados:** refere-se à quantidade de vCPUs e de instâncias utilizadas em cada experimento para a execução das requisições.

O planejamento dos experimentos foi baseado no modelo fatorial completo (JAIN, 1991). Dessa forma, 3 fatores e seus respectivos níveis foram considerados e apresentados na Tabela 13 (ambos os cenários utilizam esse planejamento).

Tabela 13 – Fatores e níveis

Fatores	Níveis
Ambiente (A)	Comum, Vertical ou Horizontal
<i>Overhead</i> do Mecanismo de Segurança (B)	Cenário 1 (0% ou 17%), Cenário 2 (0% ou 60%)
Instância (C)	<i>m3.medium</i> , <i>m3.large</i> , <i>m3.xlarge</i> ou <i>m3.2xlarge</i>

Na Tabela 13, o fator Ambiente (A) define se os experimentos serão executados em um ambiente Comum (sem o módulo ReMM), Vertical (no qual o ReMM aplica o algoritmo de escalabilidade vertical para alterar os recursos, quando necessário) ou Horizontal (o ReMM aplica o algoritmo horizontal). O fator *Overhead* do Mecanismo de Segurança (B) considera a execução dos experimentos sem segurança ou com segurança (nesse caso o *overhead* varia de acordo com o cenário). A escolha do valor que quantifica a sobrecarga é baseada no estudo apresentado no Capítulo 4, mais especificamente nos trabalhos apresentados em Popa *et al.*

(2011) e Casola *et al.* (2011). Por fim, o fator Instância (C) corresponde ao tipo de máquina virtual alocada para executar os serviços.

Outros fatores importantes são a quantidade de usuários requisitando serviços e o número de VMs disponíveis inicialmente para atender a essa demanda. Para esses fatores foram considerados valores fixos, nos quais 100 usuários requisitam serviços que são executados inicialmente por 50 VMs com configurações que variam de acordo com o experimento.

Cada experimento foi executado 10 vezes, com tempo de observação de aproximadamente 5500 segundos ou 1,5 hora (no CloudSim-BEQoS a unidade de tempo é representada em segundos). Por meio das 10 repetições percebeu-se que os resultados obtidos para cada variável de resposta não apresentavam grandes variações. Dessa forma, foram calculadas a média, o desvio padrão e o intervalo de confiança de 95% de cada configuração dos experimentos.

6.4 Análise dos Resultados

As análises apresentadas nesta Seção consideram a execução de requisições de serviço dos usuários com prazos definidos nos SLAs, bem como a presença ou não de mecanismos de segurança, seguindo o planejamento definido na Seção anterior. Para contrapor a sobrecarga imposta pelos mecanismos de segurança é utilizado o ReMM, o qual utiliza dois algoritmos de escalabilidade para alterar a quantidade de recursos durante o tempo de execução das requisições, caracterizando dois ambientes diferentes. Para uma melhor abordagem dos dados analisados, as avaliações foram divididas em duas Seções de acordo com o cenário. Vale lembrar que o objetivo não é comparar os dois cenários e sim medir os comportamentos das variáveis de resposta com diferentes taxas de chegadas de requisições e contramedidas de segurança, juntamente com o ReMM. A partir dos resultados pode-se definir modelos de negócio que representem cada ambiente relacionando segurança, desempenho e custo.

6.4.1 Cenário 1

Nesta Seção são apresentadas análises do Cenário 1 com rajadas originadas no processo de chegada das requisições individuais em tempo real, com *think times* aguardando resposta em diferentes ambientes, com e sem *overhead* de segurança de 17% (POPA *et al.*, 2011).

6.4.1.1 Experimentos sem sobrecarga de segurança – Cenário 1

Na Figura 21 e Tabela 14 são apresentados os tempos médios das execuções (em segundos) das requisições atendidas por diferentes máquinas virtuais sem a utilização de mecanismos de segurança durante o tempo de observação definido no planejamento dos experimentos. Vale

lembrar que no contrato assinado foi estipulado que as requisições deveriam ser executadas em aproximadamente 100 segundos, com 15% de variação (entre 85 e 115 segundos).

Figura 21 – Tempos médios de execuções de requisições sem *overhead* de segurança – Cenário 1

Tabela 14 – Resultados obtidos em ambientes sem *overhead* de segurança – Cenário 1

Ambiente Comum					
TME (s)	Des. Padrão	Int. Confiança	Lim. Inferior	Lim. Superior	Instâncias
301,99	7,38	4,57	297,41	306,56	<i>m3.medium</i>
149,08	5,49	3,40	145,67	152,48	<i>m3.large</i>
77,10	3,60	2,23	74,86	79,33	<i>m3.xlarge</i>
38,52	3,31	2,05	36,47	40,57	<i>m3.2xlarge</i>
Ambiente Vertical					
TME (s)	Des. Padrão	Int. Confiança	Lim. Inferior	Lim. Superior	Instâncias
100,65	8,49	5,26	95,38	105,91	<i>m3.medium</i>
101,56	9,22	5,71	95,84	107,27	<i>m3.large</i>
100,79	8,79	5,45	95,34	106,25	<i>m3.xlarge</i>
100,45	9,33	5,78	94,66	106,23	<i>m3.2xlarge</i>
Ambiente Horizontal					
TME (s)	Des. Padrão	Int. Confiança	Lim. Inferior	Lim. Superior	Instâncias
109,33	7,55	4,68	104,65	114,01	<i>m3.medium</i>
106,84	8,32	5,16	101,68	112,00	<i>m3.large</i>
108,69	9,76	6,05	102,63	114,74	<i>m3.xlarge</i>
111,13	12,47	7,73	103,40	118,86	<i>m3.2xlarge</i>

O primeiro bloco de dados corresponde às execuções das requisições de serviços em um ambiente Comum, ou seja, sem a alteração dos recursos. À medida que os tipos de instâncias foram alterados nos experimentos (*m3.medium*, *m3.large*, *m3.xlarge* e *m3.2xlarge*), houve reduções nos tempos médios de execuções, visto que a potência computacional tornou-se maior. No entanto, considerando o SLA contratado, em nenhum dos experimentos o contrato foi respeitado nesse tipo de ambiente. Além disso, os custos médios finais para a alocação de uma única VM foram iguais aos custos iniciais, pois não foram realizadas alterações nos recursos

disponíveis. O mesmo comportamento é observado para os custos médios finais do ambiente. A Tabela 15 apresenta os valores das variáveis de resposta no ambiente Comum.

Tabela 15 – Variáveis de resposta em um ambiente Comum, sem *overhead* de segurança – Cenário 1

Instâncias	TME (s)	Custo Médio Inicial por VM (\$/H)	Custo Médio Final por VM (\$/H)	Custo Médio Inicial do Ambiente (\$/H)	Custo Médio Final do Ambiente (\$/H)
<i>m3.medium</i>	301,99	0,1082	0,1082	5,41	5,41
<i>m3.large</i>	149,08	0,2166	0,2166	10,83	10,83
<i>m3.xlarge</i>	77,10	0,4387	0,4387	21,94	21,94
<i>m3.2xlarge</i>	38,52	0,8774	0,8774	43,87	43,87

Os custos médios iniciais correspondem aos preços dos recursos por hora no início das experimentações, isto é, antes das possíveis alterações na quantidade de recursos. Ao final das experimentações são gerados os custos médios finais, os quais, em caso de alterações na quantidade de recursos, sofrem alterações proporcionais ao número de recursos alterados.

Por outro lado, nos experimentos nos quais o módulo ReMM foi utilizado (ambientes Vertical e Horizontal), verificou-se que todos os tempos médios de execuções das requisições foram de acordo com os prazos definidos nos SLAs. É importante destacar que os intervalos de confiança dos TMEs (Tabela 14) de todos os experimentos executados nos ambientes Vertical e Horizontal se interceptam, e dessa forma, não existem diferenças estatísticas entre eles. Comparando esses resultados com os obtidos em um ambiente Comum, verifica-se que houve reduções nos TMEs de aproximadamente 64% na utilização de instâncias *m3.medium* e de 28% nas *m3.large*, enquanto que para as instâncias *m3.xlarge* e *m3.2xlarge*, houve acréscimos de 40% e 188%, respectivamente. Todas as alterações realizadas variaram de acordo com o algoritmo de escalabilidade utilizado.

Nos experimentos nos quais a escalabilidade vertical foi aplicada (bloco Vertical das Figuras 21 e 22), o ReMM alterou a quantidade de vCPUs das instâncias impactando proporcionalmente os custos médios finais de cada VM e do ambiente. A Tabela 16 apresenta os valores obtidos em cada experimento para esse ambiente. Houve acréscimos nos custos médios finais de aproximadamente 16% nas instâncias *m3.medium* e 4% nas *m3.large*. No entanto, nas instâncias *m3.xlarge* e *m3.2xlarge* houve reduções nos custos, 2% e 5%, respectivamente, visto que a quantidade de vCPUs nesses experimentos foi reduzida. A Figura 22 apresenta os custos médios finais de cada VM e de cada ambiente.

Tabela 16 – Variáveis de resposta em um ambiente Vertical, sem *overhead* de segurança – Cenário 1

Instâncias	TME (s)	Custo Médio Inicial por VM (\$/H)	Custo Médio Final por VM (\$/H)	Custo Médio Inicial do Ambiente (\$/H)	Custo Médio Final do Ambiente (\$/H)
<i>m3.medium</i>	100,65	0,1082	0,1261	5,41	6,31
<i>m3.large</i>	101,56	0,2166	0,2256	10,83	11,28
<i>m3.xlarge</i>	100,79	0,4387	0,4297	21,94	21,49
<i>m3.2xlarge</i>	100,45	0,8774	0,8326	43,87	41,63

Para o ambiente Horizontal, os custos médios finais por VM foram iguais aos custos iniciais, pois esse tipo de escalabilidade alterou a quantidade de instâncias e manteve as configurações internas das VMs. No entanto, para os custos médios finais do ambiente verificou-se que houve acréscimos de 100% no número de instâncias *m3.medium* e *m3.large*, resultando em

Figura 22 – Custos Médios Finais por Hora sem *overhead* de segurança – Cenário 1

um acréscimo de 100% no custo médio final de todas as VMs em relação ao ambiente Comum. Por outro lado, nos experimentos com instâncias *m3.xlarge* e *m3.2xlarge* houve reduções na quantidade de instâncias, 24% (de 50 para 38) e 60% (de 50 para 20), respectivamente, e por esse motivo houve reduções nos custos médios finais do ambiente na mesma proporção. Essas reduções ocorreram, pois o número de instâncias alocadas inicialmente (50 instâncias) foi considerado excessivo para a execução da demanda de serviço. A Tabela 17 apresenta os valores obtidos nos experimentos.

Tabela 17 – Variáveis de resposta em um ambiente Horizontal, sem *overhead* de segurança – Cenário 1

Instâncias	TME (s)	Custo Médio Inicial por VM (\$/H)	Custo Médio Final por VM (\$/H)	Custo Médio Inicial do Ambiente (\$/H)	Custo Médio Final do Ambiente (\$/H)
<i>m3.medium</i>	109,33	0,1082	0,1082	5,41	10,82
<i>m3.large</i>	106,84	0,2166	0,2166	10,83	21,66
<i>m3.xlarge</i>	108,69	0,4387	0,4387	21,94	16,67
<i>m3.2xlarge</i>	111,13	0,8774	0,8774	43,87	17,55

Essas alterações na quantidade de recursos (Tabela 18) influenciaram no número médio de requisições atendidas em cada experimento. Observa-se na Figura 23, que no ambiente Comum o número médio de requisições atendidas cresceu à medida que se alterou o tipo de instância. Por outro lado, nos ambientes com o ReMM, os números médios de requisições atendidas foram similares devido às alterações realizadas durante o tempo de execução dos experimentos, nos quais a quantidade média de recursos utilizados foram similares.

Tabela 18 – Quantidade de recursos utilizados e impacto sobre o número de requisições atendidas no Cenário 1 sem segurança

Instância	Comum			Vertical			Horizontal		
	Nº Médio vCPUs	Nº Médio Instâncias	Média Req. Atendidas	Nº Médio vCPUs	Nº Médio Instâncias	Média Req. Atendidas	Nº Médio vCPUs	Nº Médio Instâncias	Média Req. Atendidas
<i>m3.medium</i>	1	50	4166	3	50	6511	1	100	6526
<i>m3.large</i>	2	50	5224	3	50	6513	2	100	6515
<i>m3.xlarge</i>	4	50	7840	3	50	6505	4	38	6510
<i>m3.2xlarge</i>	8	50	9448	3	50	6501	8	20	6517

Figura 23 – Número de requisições atendidas sem a presença de mecanismos de segurança – Cenário 1

6.4.1.2 Experimentos com sobrecarga de segurança – Cenário 1

Na Figura 24 e Tabela 19 são apresentados os resultados para os experimentos nos quais mecanismos de segurança foram utilizados. Conforme mencionado anteriormente, a utilização de mecanismos de segurança acarreta em *overhead* sobre o desempenho do serviço. Para esse cenário, a sobrecarga de segurança foi de aproximadamente 17%.

Figura 24 – Tempos médios de execuções das requisições com *overhead* de segurança – Cenário 1

No ambiente Comum, observou-se que, assim como na Figura 21, os tempos médios de execuções das requisições diminuíram conforme o tipo de instância foi alterado. Com exceção dos experimentos com instâncias *m3.xlarge*, com tempo médio de execução de 98,63 segundos, nenhum dos SLAs foi cumprido. Dessa forma, esse tipo de instância foi considerado suficiente para a execução de requisições de serviços com *overhead* de segurança de aproximadamente 17%, não necessitando de alterações. Com relação aos custos médios finais por VM e do ambiente,

Tabela 19 – Resultados obtidos em ambientes com *overhead* de segurança

Ambiente Comum					
TME (s)	Des. Padrão	Int. Confiança	Lim. Inferior	Lim. Superior	Instâncias
398,91	9,87	6,12	392,78	405,03	<i>m3.medium</i>
193,62	5,29	3,28	190,34	196,91	<i>m3.large</i>
98,63	8,00	4,96	93,66	103,59	<i>m3.xlarge</i>
50,23	6,18	3,83	46,39	54,07	<i>m3.2xlarge</i>
Ambiente Vertical					
TME (s)	Des. Padrão	Int. Confiança	Lim. Inferior	Lim. Superior	Instâncias
107,30	8,11	5,02	102,27	112,32	<i>m3.medium</i>
108,55	8,21	5,09	103,46	113,64	<i>m3.large</i>
107,36	8,54	5,29	102,06	112,66	<i>m3.xlarge</i>
107,81	7,21	4,47	103,34	112,29	<i>m3.2xlarge</i>
Ambiente Horizontal					
TME (s)	Des. Padrão	Int. Confiança	Lim. Inferior	Lim. Superior	Instâncias
112,08	3,36	2,08	110,00	114,16	<i>m3.medium</i>
112,43	4,43	2,75	109,68	115,18	<i>m3.large</i>
112,69	3,67	2,27	110,41	114,96	<i>m3.xlarge</i>
112,57	4,38	2,71	109,85	115,29	<i>m3.2xlarge</i>

ambos mantiveram-se os mesmos, visto que nesse ambiente não foram realizadas alterações na quantidade de recursos. Os comportamentos dos custos médios finais podem ser observados nas Figuras 25a e 25b.

Figura 25 – Custos Médios Finais por Hora com *overhead* de segurança – Cenário 1

Na tentativa de contrapor o *overhead* imposto pelos mecanismos de segurança e garantir o SLA, o ReMM alterou as quantidades de recursos disponíveis nos ambientes utilizando tanto o algoritmo vertical, quanto o horizontal. Dessa forma, comparando os resultados entre os ambientes Comum e os ambientes com o ReMM, observou-se reduções de aproximadamente 72% e 42% na utilização das instâncias *m3.medium* e *m3.large*, respectivamente. Por outro lado, para as instâncias *m3.xlarge* e *m3.2xlarge*, houve acréscimos nos tempos médios de execuções das requisições de aproximadamente 14% e 124%, respectivamente, em ambientes com a escalabilidade horizontal (para esse ambiente os resultados foram estatisticamente diferentes dos resultados obtidos no ambiente Comum). Embora não haja diferenças estatísticas entre os tempos

médios de execuções dos ambientes Vertical e Horizontal, os custos médios finais apresentam valores diferentes.

Para o ambiente Vertical, o aumento na quantidade de vCPUs resultou em acréscimos nos custos médios finais por VM e do ambiente de 25% nas instâncias *m3.medium* e 8% nas *m3.large*. Nos experimentos com instâncias *m3.xlarge* não foram necessárias alterações na quantidade de vCPUs e, dessa forma, os custos médios finais por VM e do ambiente foram iguais aos respectivos custos iniciais médios. Por outro lado, na utilização de instâncias *m3.2xlarge*, houve reduções na quantidade de recursos resultando em reduções nos custos médios finais, aproximadamente 4%. Esses valores são apresentados na Tabela 20.

Tabela 20 – Variáveis de resposta em um ambiente com escalabilidade vertical e *overhead* de segurança – Cenário 1

Instâncias	TME (s)	Custo Médio Inicial por VM (\$/H)	Custo Médio Final por VM (\$/H)	Custo Médio Inicial do Ambiente (\$/H)	Custo Médio Final do Ambiente (\$/H)
<i>m3.medium</i>	107,30	0,1082	0,1348	5,41	6,74
<i>m3.large</i>	108,55	0,2166	0,2345	10,83	11,73
<i>m3.xlarge</i>	107,36	0,4387	0,4387	21,94	21,94
<i>m3.2xlarge</i>	107,81	0,8774	0,8415	43,87	42,08

Nos experimentos nos quais a escalabilidade horizontal foi aplicada, observou-se acréscimos na quantidade de instâncias *m3.medium*, o que ocasionou a elevação do custo médio final do ambiente em 200%. Para os experimentos com instâncias *m3.large*, houve um aumento de 100% na quantidade de instâncias, refletindo na mesma proporção sobre o custo médio final do ambiente. Assim como no ambiente Vertical, não houve alterações no ambiente horizontal com instâncias *m3.xlarge* e, dessa forma, o custo médio final do ambiente foi igual ao custo inicial. Por fim, para as instâncias *m3.2xlarge* foi necessária uma redução de 50% na quantidade de instâncias (de 50 para 25 instâncias), o que acarretou em uma redução do custo médio final do ambiente na mesma proporção. A Tabela 21 apresenta os resultados desses experimentos. Conforme mencionado anteriormente, o algoritmo horizontal verificou que a quantidade de instâncias alocadas inicialmente era excessiva e por essa razão reduziu o número de instâncias com impacto nos custos.

Tabela 21 – Variáveis de resposta em um ambiente com escalabilidade horizontal e *overhead* de segurança – Cenário 1

Instâncias	TME (s)	Custo Médio Inicial por VM (\$/H)	Custo Médio Final por VM (\$/H)	Custo Médio Inicial do Ambiente (\$/H)	Custo Médio Final do Ambiente (\$/H)
<i>m3.medium</i>	112,08	0,1082	0,1082	5,41	16,23
<i>m3.large</i>	112,43	0,2166	0,2166	10,83	21,66
<i>m3.xlarge</i>	112,69	0,4387	0,4387	21,94	21,94
<i>m3.2xlarge</i>	112,57	0,8774	0,8774	43,87	21,94

Assim como nos experimentos sem a presença de mecanismos de segurança, as alterações na quantidade de recursos impactaram na quantidade média de requisições atendidas. Comparado com os experimentos sem segurança, o ambiente Comum apresentou reduções no número de requisições atendidas devido ao *overhead* imposto pelos mecanismos de segurança. Por outro lado, nos ambientes Vertical e Horizontal os números médios de requisições atendidas (sem e com

segurança) foram similares. Isso ocorreu, pois o número de recursos utilizados nos experimentos que consideram a aplicação de contramedidas de segurança foi maior, dada a necessidade de contrapor o *overhead* imposto. A Figura 26 e a Tabela 22 apresentam os respectivos valores.

Figura 26 – Número de requisições atendidas com a presença de mecanismos de segurança – Cenário 1

Tabela 22 – Quantidade de recursos utilizados e impacto sobre o número de requisições atendidas no Cenário 1 com segurança

	Comum			Vertical			Horizontal		
Instância	Nº Médio vCPUs	Nº Médio Instâncias	Média Req. Atendidas	Nº Médio vCPUs	Nº Médio Instâncias	Média Req. Atendidas	Nº Médio vCPUs	Nº Médio Instâncias	Média Req. Atendidas
m3.medium	1	50	2941	4	50	6522	1	150	6529
m3.large	2	50	3665	4	50	6506	2	100	6494
m3.xlarge	4	50	6236	4	50	6501	4	50	6503
m3.2xlarge	8	50	7702	4	50	6508	8	25	6525

Na próxima Seção é apresentada uma análise da influência de cada fator sobre as variáveis de resposta TME e custos médios finais por VM e do ambiente.

6.4.1.3 Influência dos Fatores no Cenário 1

Para as análises das influências dos fatores foram considerados os fatores Ambiente (A), *Overhead* dos Mecanismos de Segurança (B) e Instância (C). Pelo fato do fator Ambiente possuir 3 níveis (Comum, Vertical e Horizontal), foram realizadas análises combinando os níveis em 2 a 2 para determinar a influência de cada fator sobre as variáveis de resposta. Por outro lado, para o fator Instância, que possui 4 níveis, foram considerados no Cenário 1 os níveis extremos, ou seja, as instâncias dos tipos *m3.medium* e *m3.2xlarge*. Esses níveis foram escolhidos com base nos resultados apresentados na Seção anterior, nos quais a utilização desses níveis foi mais impactante nas variáveis de resposta. Vale lembrar que para o fator *Overhead* foram considerados experimentos sem (0%) e com a presença de mecanismos de segurança (17%).

Na Tabela 23 são apresentadas as influências dos fatores para a combinação entre os ambientes Comum e Vertical. Verifica-se que os tempos médios de execuções foram mais

influenciados pela combinação dos fatores Ambiente e Instância (AC), com 40,45% de influência. Em seguida, tem-se o fator C com 40,36% e o fator A com 15,04%. Esses percentuais são justificados pelas reduções dos TMEs provocados pelas diferenças computacionais entre as instâncias *m3.medium* e *m3.2xlarge* e pelas alterações realizadas pelo algoritmo vertical. Essas diferenças computacionais também justificam os percentuais obtidos nos custos médios finais por VM e do ambiente, 99,80% para ambos.

Tabela 23 – Percentuais de influência dos fatores nos ambientes Comum e Vertical – Cenário 1

Combinação	TME (%)	Custo Médio Final por VM (%)	Custo Médio Final por Ambiente (%)
A	15,04	0,02	0,01
B	1,62	0,00	0,00
C	40,36	99,80	99,80
AB	0,97	0,00	0,00
AC	40,45	0,18	0,18
BC	0,77	0,00	0,00
ABC	0,80	0,00	0,00

Ambiente (A), *Overhead* dos Mecanismos de Segurança (B) e Instância (C).

Portanto, independente da sobrecarga imposta pelos mecanismos de segurança, que nessa combinação influenciou em apenas 1,62% (fator B), o tipo de instância foi o fator determinante nos resultados obtidos em conjunto com o algoritmo vertical que alterou os recursos computacionais disponíveis respeitando o SLA contratado com impacto nos custos.

Na Tabela 24 são apresentados os percentuais de influência na combinação dos ambientes Comum e Horizontal. Verifica-se o mesmo comportamento descrito na combinação anterior (ambientes Comum e Vertical), pois assim como o algoritmo de escalabilidade vertical, o algoritmo horizontal cumpriu os SLAs. Dessa forma, tem-se a combinação AC com o maior percentual de influência (41,66%), seguida por C (41,04%) e A (13,10%). Para o fator B a influência sobre o TME foi de apenas 1,40%.

Tabela 24 – Percentuais de influência dos fatores nos ambientes Comum e Horizontal – Cenário 1

Combinação	TME (%)	Custo Médio Final por VM (%)	Custo Médio Final por Ambiente (%)
A	13,10	0,00	7,67
B	1,40	0,00	0,72
C	41,04	100,00	59,76
AB	1,20	0,00	0,72
AC	41,66	0,00	31,11
BC	0,83	0,00	0,01
ABC	0,78	0,00	0,01

Ambiente (A), *Overhead* dos Mecanismos de Segurança (B) e Instância (C).

Ao contrário do ambiente Vertical, o Horizontal altera a quantidade de instâncias e mantém a configuração original interna de cada VM. Por essa razão, o fator C foi o único a influenciar no custo médio final por VM, uma vez que esses custos não sofreram alterações. Por

outro lado, as alterações na quantidade de instâncias fizeram com que o fator A exercesse 7,67% de influência sobre os custos médios finais do ambiente. O fator mais impactante, assim como na combinação anterior do ambientes, foi o C com 59,76%, seguido pela combinação AC com 31,11%. Portanto, o tipo de instância utilizada e a forma com que os recursos do ambiente foram alterados foram os fatores mais impactantes nas variáveis de resposta.

Por fim, a Tabela 25 apresenta os percentuais de influência dos fatores sobre as variáveis de resposta nos ambientes Vertical e Horizontal. Vale lembrar que, embora ambos apliquem a escalabilidade de recursos, o primeiro considera apenas a alteração da quantidade de vCPUs, enquanto que o segundo considera a alteração da quantidade de instâncias. Por esse motivo, tem-se que o fator mais impactante sobre os TMEs foi o fator A com 65,35% de influência. Ao contrário do que foi observado nas combinações anteriores, nas quais o fator B (*Overhead*) praticamente não exerceu influência nas variáveis de resposta, nessa combinação ele apresentou uma influência de 25,92% nos TMEs. A combinação entre os dois fatores A e B apareceu em terceiro lugar com influência de 7,55%.

Tabela 25 – Percentuais de influência dos fatores nos ambientes Vertical e Horizontal – Cenário 1

Combinação	TME (%)	Custo Médio Final por VM (%)	Custo Médio Final por Ambiente (%)
A	65,35	0,02	8,01
B	25,92	0,00	1,00
C	0,53	99,80	60,55
AB	7,55	0,00	0,70
AC	0,31	0,18	29,72
BC	0,03	0,00	0,01
ABC	0,32	0,00	0,01

Ambiente (A), *Overhead* dos Mecanismos de Segurança (B) e Instância (C).

Embora o tipo de instância, *m3.medium* ou *m3.2xlarge*, não tenha praticamente influenciado nos tempos médios de execuções (0,53%), esse fator exerceu grandes impactos nos custos médios finais, 99,80% nos custos por VM e 60,55% nos custos do ambiente. No segundo caso, vale destacar a influência da combinação AC com 29,72% causada pelo tipo de instância utilizada e o tipo de algoritmo responsável por alterar a quantidade de recursos.

Portanto, independente do tipo de instância escolhido, o ReMM alterou a quantidade de recursos disponíveis para garantir o SLA na presença ou não de mecanismos de segurança com impactos nos custos. No entanto, quando se trata dos custos médios finais, o tipo de instância utilizada exerceu grande impacto, pois quanto mais potente é a instância, maior é o custo inicial, o qual serve de base para a geração do custo final que considera as possíveis alterações dos recursos.

6.4.2 Cenário 2

O segundo cenário considera ambientes em nuvem executando requisições com *think times* após o envio da requisição prévia, com e sem a presença de mecanismo de segurança que geram uma sobrecarga de 60% (CASOLA *et al.*, 2011).

6.4.2.1 Experimentos sem sobrecarga de segurança – Cenário 2

A Figura 27 e a Tabela 26 apresentam os resultados obtidos sem a utilização de mecanismos de segurança. Considerando um ambiente Comum, no qual não há alterações na quantidade de recursos, verificou-se que os SLAs não foram cumpridos, com exceção do experimento com instâncias *m3.2xlarge*, no qual o intervalo superior do TME está dentro da margem de variação do SLA de 15%, ou seja, entre 85 e 115 segundos. À medida que a potência computacional foi maior, reduziu-se os tempos médios de execuções das requisições. Além disso, os custos médios finais de cada VM e do ambiente não sofreram alterações (Figuras 28a e 28b).

Figura 27 – Tempos médios de execuções de requisições sem *overhead* de segurança – Cenário 2

Nos experimentos com o ReMM verificou-se que todos os contratos foram cumpridos com tempos médios de execuções em torno de 100 segundos, não havendo diferenças estatísticas entre os experimentos, uma vez que os intervalos de confiança se sobrepuseram. As reduções nos TMEs em relação ao ambiente Comum foram de aproximadamente 82% com instâncias *m3.medium*, 66% com *m3.large* e 29% com *m3.xlarge*, enquanto que com instâncias *m3.2xlarge* houve um acréscimo de aproximadamente 39% no TME em relação ao ambiente Comum.

No ambiente Vertical as alterações na quantidade de vCPUs proporcionaram alterações nos custos médios finais de cada VM e do ambiente. Houve acréscimos nos custos de aproximadamente 40% com instâncias *m3.medium*, 16% com *m3.large* e 4% com *m3.xlarge*. Por outro

Tabela 26 – Resultados obtidos em ambientes sem *overhead* de segurança – Cenário 2

Ambiente Comum					
TME (s)	Des. Padrão	Int. Confiança	Lim. Inferior	Lim. Superior	Instâncias
653,24	11,34	7,03	646,21	660,28	<i>m3.medium</i>
330,73	9,85	6,10	324,62	336,84	<i>m3.large</i>
161,00	8,42	5,22	155,77	166,22	<i>m3.xlarge</i>
82,49	6,92	4,29	78,20	86,78	<i>m3.2xlarge</i>
Ambiente Vertical					
TME (s)	Des. Padrão	Int. Confiança	Lim. Inferior	Lim. Superior	Instâncias
114,49	13,65	8,46	106,02	122,95	<i>m3.medium</i>
113,72	9,02	5,59	108,13	119,31	<i>m3.large</i>
113,91	10,55	6,54	107,37	120,45	<i>m3.xlarge</i>
114,55	11,60	7,19	107,35	121,74	<i>m3.2xlarge</i>
Ambiente Horizontal					
TME (s)	Des. Padrão	Int. Confiança	Lim. Inferior	Lim. Superior	Instâncias
112,98	16,67	10,33	102,65	123,31	<i>m3.medium</i>
113,71	8,89	5,51	108,20	119,23	<i>m3.large</i>
114,26	10,42	6,46	107,80	120,72	<i>m3.xlarge</i>
114,36	11,09	6,87	107,49	121,23	<i>m3.2xlarge</i>

Figura 28 – Custos Médios Finais por Hora sem *overhead* de segurança – Cenário 2

lado, nos experimentos com instâncias *m3.2xlarge* houve reduções de 2% nos custos. A Tabela 27 apresenta os valores de cada experimento em um ambiente Vertical.

Tabela 27 – Variáveis de resposta em um ambiente com escalabilidade vertical, sem *overhead* de segurança – Cenário 2

Instâncias	TME (s)	Custo Médio Inicial por VM (\$/H)	Custo Médio Final por VM (\$/H)	Custo Médio Inicial do Ambiente (\$/H)	Custo Médio Final do Ambiente (\$/H)
<i>m3.medium</i>	114,49	0,1082	0,1519	5,41	7,60
<i>m3.large</i>	113,72	0,2166	0,2522	10,83	12,61
<i>m3.xlarge</i>	113,91	0,4387	0,4567	21,94	22,83
<i>m3.2xlarge</i>	114,55	0,8774	0,8594	43,87	42,97

Com relação ao ambiente Horizontal, as alterações na quantidade de instâncias exerceram maior impacto nos custos médios finais das instâncias do tipo *m3.medium* e *m3.large*, acréscimos de 400% e 200%, respectivamente. Por outro lado, nos experimentos com instâncias *m3.xlarge* e *m3.2xlarge*, o ReMM alternou entre aumentar e reduzir as quantidades de instâncias do ambiente

até que o número máximo de alterações por VM (10 alterações) foi alcançado. Dessa forma, embora os tempos médios de execuções estejam de acordo com o SLA, esse tipo de escalabilidade não foi eficiente nesses casos, uma vez que os custos médios finais foram iguais aos custos médios iniciais. Dessa forma, torna-se necessária a aplicação de técnicas de otimização. A Tabela 28 apresenta os resultados dos experimentos.

Tabela 28 – Variáveis de resposta em um ambiente com escalabilidade horizontal sem *overhead* de segurança – Cenário 2

Instâncias	TME (s)	Custo Médio Inicial por VM (\$/H)	Custo Médio Final por VM (\$/H)	Custo Médio Inicial do Ambiente (\$/H)	Custo Médio Final do Ambiente (\$/H)
<i>m3.medium</i>	112,98	0,1082	0,1082	5,41	27,05
<i>m3.large</i>	113,71	0,2166	0,2166	10,83	32,49
<i>m3.xlarge</i>	114,26	0,4387	0,4387	21,94	21,94
<i>m3.2xlarge</i>	114,36	0,8774	0,8774	43,87	43,87

As alterações realizadas pelo ReMM influenciaram o número médio de requisições atendidas em cada experimento, gerando resultados estatisticamente iguais nos ambientes Vertical e Horizontal. Por outro lado, no ambiente Comum, à medida que se alterou o tipo de instância, aumentou-se a quantidade de requisições atendidas. A Figura 29 e a Tabela 29 apresentam os resultados obtidos.

Figura 29 – Número de requisições atendidas sem a presença de mecanismos de segurança – Cenário 2

Tabela 29 – Quantidade de recursos utilizados e impacto sobre o número de requisições atendidas no Cenário 2 sem segurança

Instância	Comum			Vertical			Horizontal		
	Nº Médio vCPUs	Nº Médio Instâncias	Média Req. Atendidas	Nº Médio vCPUs	Nº Médio Instâncias	Média Req. Atendidas	Nº Médio vCPUs	Nº Médio Instâncias	Média Req. Atendidas
<i>m3.medium</i>	1	50	496	5	50	1099	1	250	1167
<i>m3.large</i>	2	50	594	5	50	1073	2	150	1099
<i>m3.xlarge</i>	4	50	895	5	50	1071	4	50	1104
<i>m3.2xlarge</i>	8	50	1061	6	50	1046	8	50	1088

6.4.2.2 Experimentos com sobrecarga de segurança – Cenário 2

Na Figura 30 e na Tabela 30 são apresentados os resultados para os experimentos nos quais foi considerada a utilização de mecanismos de segurança com sobrecarga de 60%. A utilização de mecanismos de segurança influenciou negativamente nos experimentos nos quais o ReMM não foi utilizado. Como exemplo, pode-se citar os experimentos em um ambiente com instâncias *m3.medium*, nos quais o tempo médio de execução foi de 859,97 segundos. Comparando com os experimentos nos quais a instância *m3.2xlarge* foi utilizada, único tipo de instância onde os SLAs foram cumpridos em um ambiente Comum, houve uma redução na variável de resposta de aproximadamente 87%.

Figura 30 – Tempos médios de execuções de requisições com *overhead* de segurança – Cenário 2Tabela 30 – Resultados obtidos em ambientes com *overhead* de segurança – Cenário 2

Ambiente Comum					
TME (s)	Des. Padrão	Int. Confiança	Lim. Inferior	Lim. Superior	Instâncias
859,97	12,91	8,00	851,97	867,98	<i>m3.medium</i>
430,98	12,51	7,75	423,22	438,73	<i>m3.large</i>
217,71	12,39	7,68	210,03	225,39	<i>m3.xlarge</i>
112,04	10,82	6,70	105,33	118,74	<i>m3.2xlarge</i>
Ambiente Vertical					
TME (s)	Des. Padrão	Int. Confiança	Lim. Inferior	Lim. Superior	Instâncias
114,92	13,24	8,20	106,71	123,13	<i>m3.medium</i>
110,78	11,25	6,97	103,81	117,76	<i>m3.large</i>
110,90	13,02	8,07	102,83	118,98	<i>m3.xlarge</i>
112,55	12,16	7,54	105,01	120,09	<i>m3.2xlarge</i>
Ambiente Horizontal					
TME (s)	Des. Padrão	Int. Confiança	Lim. Inferior	Lim. Superior	Instâncias
113,00	16,77	10,39	102,60	123,40	<i>m3.medium</i>
110,50	12,66	7,85	102,64	118,35	<i>m3.large</i>
109,96	11,21	6,94	103,01	116,91	<i>m3.xlarge</i>
109,24	12,27	7,60	101,63	116,84	<i>m3.2xlarge</i>

Por outro lado, nos ambientes Vertical e Horizontal, o *overhead* imposto pelos mecanismos de segurança foi suprido pelas alterações na quantidade de recursos. Dessa forma, os contratos foram garantidos gerando reduções nos tempos médios de execuções de aproximadamente 87%, 74% e 49% nos ambientes com instâncias *m3.medium*, *m3.large* e *m3.xlarge*, respectivamente, enquanto que com as instâncias *m3.2xlarge* os resultados nos três ambientes foram similares, uma vez que não houve diferenças estatísticas entre os valores obtidos.

Considerando que o *overhead* imposto foi de 60%, o ReMM alterou a quantidade de recursos na tentativa de contrapor essa sobrecarga e garantir o SLA. As alterações feitas impactaram na quantidade média de requisições atendidas, como mostram a Figura 31 e a Tabela 31.

Figura 31 – Número de requisições atendidas com a presença de mecanismos de segurança – Cenário 2

Tabela 31 – Quantidade de recursos utilizados e impacto sobre o número de requisições atendidas no Cenário 2 com segurança

Instância	Comum			Vertical			Horizontal		
	Nº Médio vCPUs	Nº Médio Instâncias	Média Req. Atendidas	Nº Médio vCPUs	Nº Médio Instâncias	Média Req. Atendidas	Nº Médio vCPUs	Nº Médio Instâncias	Média Req. Atendidas
<i>m3.medium</i>	1	50	350	7	50	1130	1	400	1143
<i>m3.large</i>	2	50	428	7	50	1040	2	200	1067
<i>m3.xlarge</i>	4	50	598	7	50	1000	4	100	1101
<i>m3.2xlarge</i>	8	50	1062	8	50	1060	8	50	1048

Para os ambientes com o algoritmo vertical, as alterações nas quantidades de vCPUs provocaram acréscimos nos custos médios finais de aproximadamente 56% nas instâncias *m3.medium*, 25% nas *m3.large* e 8% nas *m3.xlarge*. A quantidade de recursos disponíveis nas instâncias *m3.2xlarge* foi considerada suficiente para as execuções das requisições de serviço com mecanismos de segurança e, dessa forma, não foram necessárias alterações. A Tabela 32 e a Figura 32 apresentam os resultados obtidos.

Nos experimentos com a escalabilidade horizontal, houve aumentos expressivos nas quantidades de instâncias *m3.medium* e *m3.large*, 700% e 300%, respectivamente. Para instâncias *m3.xlarge* esse aumento foi de 100%, enquanto que, não houve alterações nos ambientes com

Tabela 32 – Variáveis de resposta em um ambiente com escalabilidade vertical e *overhead* de segurança – Cenário 2

Instâncias	TME (s)	Custo Médio Inicial por VM (\$/H)	Custo Médio Final por VM (\$/H)	Custo Médio Inicial do Ambiente (\$/H)	Custo Médio Final do Ambiente (\$/H)
<i>m3.medium</i>	114,92	0,1082	0,1686	5,41	8,43
<i>m3.large</i>	110,78	0,2166	0,2698	10,83	13,49
<i>m3.xlarge</i>	110,90	0,4387	0,4744	21,94	23,72
<i>m3.2xlarge</i>	112,55	0,8774	0,8774	43,87	43,87

Figura 32 – Custos Médios Finais por Hora com *overhead* de segurança – Cenário 2

instâncias *m3.2xlarge*. Esses aumentos nas quantidades de instâncias provocaram acréscimos proporcionais aos respectivos custos médios finais do ambiente. Os valores referentes ao ambiente Horizontal são apresentados na Tabela 33.

Tabela 33 – Variáveis de resposta em um ambiente com escalabilidade horizontal e *overhead* de segurança – Cenário 2

Instâncias	TME (s)	Custo Médio Inicial por VM (\$/H)	Custo Médio Final por VM (\$/H)	Custo Médio Inicial do Ambiente (\$/H)	Custo Médio Final do Ambiente (\$/H)
<i>m3.medium</i>	113,00	0,1082	0,1082	5,41	43,28
<i>m3.large</i>	110,50	0,2166	0,2166	10,83	43,32
<i>m3.xlarge</i>	109,96	0,4387	0,4387	21,94	43,87
<i>m3.2xlarge</i>	109,24	0,8774	0,8774	43,87	43,87

A próxima Seção apresenta análises das influências exercidas por cada fator sobre as variáveis de resposta TME e custos médios finais no Cenário 2.

6.4.2.3 Influência dos Fatores no Cenário 2

Para as análises dos percentuais de influência dos fatores Ambiente (A), *Overhead* dos Mecanismos de Segurança (B) e Instância (C) no Cenário 2, foram realizadas combinações 2 a 2 dos níveis do fator Ambiente. Além disso, pelo fato do fator Instância possuir 4 níveis, foram considerados os níveis *m3.medium* e *m3.large*, os quais possuem diferenças computacionais menores em relação às outras possíveis combinações de instâncias. Vale lembrar que para o fator *Overhead* (B) foram considerados experimentos sem (0%) e com a presença de mecanismos de segurança (60%), obtidos dos estudos apresentados por [Casola et al. \(2011\)](#).

A Tabela 34 apresenta os percentuais de influência dos fatores mencionados sobre as variáveis de resposta dos ambientes Comum e Vertical. Considerando as alterações realizadas pelo algoritmo de escalabilidade vertical, necessárias para o cumprimento dos SLAs, o fator com maior impacto sobre os tempos médios de execuções foi o A com 71,21% de influência. Em seguida tem-se o fator C com 12,29% e a combinação AC com 11,97%. Além disso, dada as diferenças computacionais presentes nos tipos de instâncias analisadas, onde as instâncias *m3.medium* possuem 1 vCPU e as *m3.large* possuem 2 vCPUs cada, o fator C deixou de exercer total influência sobre os custos médios finais, ou seja, a diferença dos recursos presentes nas instâncias tornou-se menor, e por isso, o fator A exerceu mais influência nos custos médios finais com aproximadamente 17,33%, enquanto que o fator C influenciou em aproximadamente 81,47%.

Tabela 34 – Percentuais de influência dos fatores nos ambientes Comum e Vertical – Cenário 2

Combinação	TME (%)	Custo Médio Final por VM (%)	Custo Médio Final por Ambiente (%)
A	71,21	17,33	17,35
B	1,99	0,55	0,54
C	12,29	81,47	81,45
AB	2,06	0,55	0,54
AC	11,97	0,11	0,11
BC	0,26	0,00	0,00
ABC	0,23	0,00	0,00

Ambiente (A), *Overhead* dos Mecanismos de Segurança (B) e Instância (C).

Com relação aos ambientes Comum e Horizontal, verifica-se que os percentuais de influência dos fatores sobre os tempos médios de execuções são similares aos apresentados na combinação anterior. A ordem de influência se manteve a mesma com o fator A em primeiro com 71,29%, seguido por C com 12,15% e AC com 12,04%. A Tabela 35 apresenta esses valores. Com relação aos custos médios finais, verifica-se que o fator C exerceu 100% de influência sobre os custos médios finais por VM, uma vez que não há alterações nos recursos internos de uma máquina virtual em ambos os ambientes Comum e Horizontal. Por outro lado, dada a pequena diferença de recursos computacionais entre os tipos de instâncias analisadas, o algoritmo de escalabilidade horizontal destacou-se pelas alterações no número de instâncias e por isso, exerceu a maior influência sobre os custos médios finais do ambiente com 87,66%.

Por fim, nos percentuais de influência dos fatores nos ambientes Vertical e Horizontal apresentados na Tabela 36, verifica-se que vários fatores influenciaram nos tempos médios de execuções das requisições, ao contrário do que foi observado nas outras combinações dos ambientes. Observa-se que o fator C foi o que mais influenciou nos TMEs com 30,81%, seguido pela combinação BC com 30,07%. Até então, o fator B praticamente não havia influenciado nas variáveis de resposta das outras combinações. No entanto, considerando os ambientes Vertical e Horizontal, a sua influência foi de 22,43% sobre os tempos médios de execuções. Vale destacar o impacto gerado pelo fator A com 9,55%. Essa redução da influência em relação às

Tabela 35 – Percentuais de influência dos fatores nos ambientes Comum e Horizontal – Cenário 2

Combinação	TME (%)	Custo Médio Final por VM (%)	Custo Médio Final por Ambiente (%)
A	71,29	0	87,66
B	1,98	0	4,97
C	12,15	100	1,81
AB	2,06	0	4,97
AC	12,04	0	0,19
BC	0,26	0	0,20
ABC	0,23	0	0,20

Ambiente (A), *Overhead* dos Mecanismos de Segurança (B) e Instância (C).

demais combinações de ambientes deve-se ao fato de que em ambos os ambientes os tempos médios de execuções foram similares, diferenciando-se nas formas em que a escalabilidade dos recursos foi aplicada. Os impactos dessas alterações dos recursos ficam claras na Tabela 36, na qual para os custos médios finais por VM o fator C foi o mais influente com 81,47%. Isso é justificado pelo fato das alterações feitas pelo algoritmo vertical, que altera a quantidade de vCPUs, gerarem aumentos nos custos por VMs proporcionais à quantidade de vCPUs. Dessa forma, as alterações nas vCPUs são ofuscadas pelo tipo de instâncias usadas. Por outro lado, no custo médio final por ambiente o fator que mais influenciou foi o A com 85,79%, dado que o algoritmo Horizontal alterou a quantidade de instâncias disponíveis para contrapor o *overhead* imposto pelos mecanismos de segurança sobre o TME e garantir o SLA.

Tabela 36 – Percentuais de influência dos fatores nos ambientes Vertical e Horizontal – Cenário 2

Combinação	TME (%)	Custo Médio Final por VM (%)	Custo Médio Final por Ambiente (%)
A	9,55	13,33	85,79
B	22,43	0,55	6,56
C	30,81	81,47	1,92
AB	0,32	0,55	5,10
AC	6,81	0,11	0,17
BC	30,07	0,00	0,23
ABC	0,01	0,00	0,24

Ambiente (A), *Overhead* dos Mecanismos de Segurança (B) e Instância (C).

Dessa forma, o custo de uma instância é mais expressivo que o custo de uma vCPU dada a variedade de recursos que a compõem. Por essa razão, o fator Instância destacou-se nos custos médios finais por VM, enquanto que o fator Ambiente destacou-se nos custos médios finais do ambiente.

6.4.3 Modelos de Negócio

Em computação em nuvem observa-se que a infraestrutura é o pilar do estabelecimento de preços e da qualidade com que os demais serviços serão prestados. Dessa forma, a manutenção dos recursos de infraestrutura se torna cada vez mais importante. O usuário tem a visão de um

recurso único e de uso exclusivo, e por isso, espera-se o máximo de desempenho dos atributos selecionados para o seu serviço. Em contrapartida, o provedor possui várias máquinas físicas e precisa hospedar as máquinas virtuais dos clientes de forma a maximizar o seu lucro e economizar recursos, principalmente, energia. Essas considerações influenciam o modelo de negócio definido por meio do desempenho obtido e do preço cobrado do cliente.

Nos experimentos apresentados neste Capítulo foram realizadas análises do comportamento de um serviço em diferentes ambientes com e sem a presença de mecanismos de segurança. Para isso, dois cenários foram definidos. No primeiro, a utilização de mecanismos de segurança implicou em um *overhead* de 17% sobre o desempenho do serviço. Para representar as requisições dos usuários foram utilizadas rajadas com *think times* aguardando resposta, o que gera um intervalo de tempo maior entre uma requisição e outra. No segundo cenário, os mecanismos de segurança sobrecarregaram o serviço em aproximadamente 60%, prejudicando o atendimento das requisições, as quais são representadas por rajadas com *think times* após o envio da requisição prévia, isto é, o tempo entre o envio de uma requisição e outra é menor. Dessa forma, o objetivo nos dois cenários foi verificar o comportamento das variáveis de resposta com diferentes configurações de ambientes não estabelecendo qualquer tipo de comparação entre os cenários.

Para executar a demanda de serviços foram utilizadas instâncias de máquinas virtuais, as quais possuem configurações e preços diferentes. Essas VMs foram hospedadas em diferentes ambientes. O primeiro, chamado Comum, consiste de um ambiente não-escalável com relação ao provimento de recursos virtuais. Dessa forma, o modelo de negócio aplicado não foi eficiente, pois, dada a demanda por serviços e os recursos computacionais virtualizados disponíveis para atendê-la, verificou-se que na grande maioria dos experimentos as requisições não foram executadas dentro do prazo definido no SLA.

A utilização de mecanismos de segurança contribuiu para os aumentos nos tempos médios de execuções das requisições. A Tabela 37 apresenta uma comparação entre os tempos médios obtidos em um ambiente Comum com e sem a presença de mecanismos de segurança nos dois cenários. Vale mencionar que, como não houve alterações na quantidade de recursos no ambiente Comum, os custos médios finais permaneceram sem modificações.

Tabela 37 – Tabela comparativa dos TMEs (em segundos) em um ambiente Comum

Instância	Cenário 1		Cenário 2	
	Sem Segurança	Com Segurança	Sem Segurança	Com Segurança
<i>m3.medium</i>	301,99	398,91	653,24	859,97
<i>m3.large</i>	149,08	193,62	330,73	430,98
<i>m3.xlarge</i>	77,10	98,63	161,00	217,71
<i>m3.2xlarge</i>	38,52	50,23	82,49	112,04

Considerando os resultados apresentados para os experimentos nos ambientes Vertical e Horizontal, verificou-se que o módulo ReMM soube contrapor a sobrecarga gerada pelos mecanismos de segurança sobre as variáveis de resposta, garantindo o SLA com alterações nos

recursos, e consequentemente, impactando os custos e as quantidades médias de requisições atendidas.

Embora não haja diferenças estatísticas nos TMEs nos ambientes Vertical e Horizontal dos dois cenários, o modelo de negócio adotado e a forma com que cada ambiente aplicou a escalabilidade dos recursos para contrapor o *overhead* e garantir o SLA em cada cenário impactou os custos médios finais. Dessa forma, é possível estabelecer uma relação entre a quantidade de recursos utilizados, o tempo médio de execução obtido e o custo pago pelo serviço considerando uma única VM e o ambiente como um todo (isto é, todas as instâncias).

As análises dos percentuais apresentadas na Tabela 38 consideram as comparações entre os ambientes com (Vertical ou Horizontal) e sem (Comum) a utilização do ReMM no Cenário 1. Nos casos onde os percentuais de aumento ou redução dos TMEs são menores ou iguais a margem de variação do SLA (15%) não houve alterações nos custos médios finais, e dessa forma, as instâncias utilizadas foram consideradas ideais para a execução das requisições de serviço. Outra informação importante é o prazo definido no SLA para a execução de uma requisição sem ou com segurança (0% ou 17% de sobrecarga), o qual foi de 100 segundos para esse cenário.

Tabela 38 – Percentuais aproximados de aumento (+) e de redução (-) exercidos pelo ReMM sobre as variáveis de resposta na tentativa de contrapor a sobrecarga de segurança e garantir o SLA no Cenário 1

Ambiente Vertical em relação ao Ambiente Comum						
Instância	Sem Segurança			Com Segurança		
	TME	Custo/VM	Custo/Ambiente	TME	Custo/VM	Custo/Ambiente
<i>m3.medium</i>	- 67%	+ 16%	+ 16%	- 73%	+ 25%	+ 25%
<i>m3.large</i>	- 32%	+ 4%	+ 4%	- 44%	+ 8%	+ 8%
<i>m3.xlarge</i>	+ 31%	- 2%	- 2%	+ 9%	0%	0%
<i>m3.2xlarge</i>	+ 161%	- 5%	- 5%	+ 115%	- 4%	- 4%
Ambiente Horizontal em relação ao Ambiente Comum						
Instância	Sem Segurança			Com Segurança		
	TME	Custo/VM	Custo/Ambiente	TME	Custo/VM	Custo/Ambiente
<i>m3.medium</i>	- 64%	0%	+ 100%	- 72%	0%	+ 200%
<i>m3.large</i>	- 28%	0%	+ 100%	- 42%	0%	+ 100%
<i>m3.xlarge</i>	+ 41%	0%	- 24%	+ 14%	0%	0%
<i>m3.2xlarge</i>	+ 188%	0%	- 60%	+ 124%	0%	- 50%

As reduções nos tempos médios de execuções proporcionadas pelo ambiente Vertical em relação ao ambiente Comum com instâncias *m3.medium* e *m3.large* exigiram acréscimos no número de vCPUs das instâncias, resultando em aumentos proporcionais nos custos médios finais. Por outro lado, visando garantir o SLA esse ambiente (Vertical) reduziu a quantidade de vCPUs das instâncias *m3.xlarge* e *m3.2xlarge* (instâncias com maior potência computacional), e por essa razão houve acréscimos nos TMEs. Além disso, verificou-se que a utilização de mecanismos de segurança foi penosa para as instâncias com potência computacional menor, elevando os custos médios finais.

Nas análises comparativas entre os ambientes Horizontal e Comum vale destacar que, embora os percentuais de aumentos/reduções nos TMEs sejam semelhantes às análises apresentadas no parágrafo anterior, os impactos gerados sobre os custos médios finais são maiores,

pois o algoritmo horizontal trabalha com a alteração do número de instâncias, enquanto que o vertical considera a alteração de vCPUs. A Tabela 38 apresenta os valores utilizados nas análises comparativas descritas, os quais podem ser utilizados para a definição do melhor modelo de negócio para o Cenário 1.

As análises dos percentuais apresentadas na Tabela 39 consideram as alterações realizadas pelos ambientes Vertical e Horizontal em relação ao ambiente Comum. Nessas análises a sobrecarga imposta pelos mecanismos de segurança foi de 60%, o que acarretou em uma quantidade maior de recursos utilizados para contrapor esse *overhead*. Vale destacar também que a taxa de chegada das requisições nesse cenário utilizou intervalos de tempos menores entre uma requisição e outra, proporcionando um número maior de requisições de serviços. Esses fatos exigiram alterações nas quantidades de recursos para que as exigências definidas nos SLAs fossem cumpridas.

Tabela 39 – Percentuais aproximados de aumento (+) e de redução (-) exercidos pelo ReMM sobre as variáveis de resposta na tentativa de contrapor o *overhead* e garantir o SLA no Cenário 2

Ambiente Vertical em relação ao Ambiente Comum						
Instância	Sem Segurança			Com Segurança		
	TME	Custo/VM	Custo/Ambiente	TME	Custo/VM	Custo/Ambiente
<i>m3.medium</i>	- 83%	+ 40%	+ 40%	- 87%	+ 56%	+ 56%
<i>m3.large</i>	- 66%	+ 16%	+ 16%	- 74%	+ 25%	+ 25%
<i>m3.xlarge</i>	- 29%	+ 4%	+ 4%	- 49%	+ 8%	+ 8%
<i>m3.2xlarge</i>	+ 39%	- 2%	- 2%	+ 0,04%	0%	0%
Ambiente Horizontal em relação ao Ambiente Comum						
Instância	Sem Segurança			Com Segurança		
	TME	Custo/VM	Custo/Ambiente	TME	Custo/VM	Custo/Ambiente
<i>m3.medium</i>	- 83%	0%	+ 400%	- 87%	0%	+ 700%
<i>m3.large</i>	- 66%	0%	+ 200%	- 74%	0%	+ 300%
<i>m3.xlarge</i>	- 29%	0%	Não foi preciso *	- 49%	0%	+ 100%
<i>m3.2xlarge</i>	+ 37%	0%	Não foi preciso *	- 2%	0%	0%

* Os experimentos com requisições sem segurança executadas em instâncias *m3.xlarge* e *m3.2xlarge* alcançaram o máximo de alterações permitidas em um ambiente Horizontal. Por essa razão, considerou-se que os custos médios do ambiente (\$/H) não foram precisos.

No ambiente Vertical os TMEs das requisições foram reduzidos com as alterações do número de vCPUs das instâncias *m3.medium*, *m3.large* e *m3.xlarge*, gerando acréscimos proporcionais nos custos médios finais. Por outro lado, para a instância *m3.2xlarge* executando requisições sem segurança houve uma redução no número de vCPUs, proporcionando um acréscimo no TME de 39% que garantiu conformidade com a margem de variação do SLA. Para a execução de requisições com segurança em instâncias *m3.2xlarge*, o percentual de aumento do TME foi pequeno (0,04%), valor menor que a margem de variação do SLA (15%) e, por isso, não houve alterações nos custos médios finais. Dessa forma, a instância utilizada foi considerada ideal para a execução desse tipo de requisição de serviço.

Para o ambiente Horizontal, verificou-se que os percentuais de acréscimos/reduções dos TMEs foram similares aos obtidos no ambiente Vertical. No entanto, os impactos sobre os custos

médios finais foram demasiados, principalmente nos experimentos com instâncias *m3.medium* e *m3.large*, nos quais houve acréscimos de 700% e 300% nos custos médios finais do ambiente (\$/H), respectivamente.

Portanto, com base nos resultados das experimentações, verificou-se que o modelo de negócio aplicado por um ambiente Comum não foi eficiente considerando as exigências relacionadas ao prazo de execução e a utilização de mecanismos de segurança definidas no SLA. Embora os custos finais permanecessem inalterados, visto que não há alterações na quantidade de recursos, a grande maioria dos SLAs não foi cumprida, independente da utilização ou não das contramedidas de segurança. Dessa forma, o modelo de negócio utilizado nesse ambiente não é recomendável para uma nuvem na qual a demanda de serviços muda constantemente.

No modelo de negócio adotado no ambiente Horizontal, verificou-se que os contratos foram cumpridos graças as alterações na quantidade de recursos aplicadas pelo algoritmo horizontal. No entanto, essas alterações consistem na adição ou remoção de instâncias e, consequentemente, de todos os recursos que compõem uma VM (vCPU, memória e disco). Dessa forma, os custos finais foram mais expressivos. Conforme mencionado no Capítulo 5, os provedores fornecem a opção de escalabilidade de recursos, a qual é aplicada de forma horizontal, onde o número de instâncias é alterado respeitando as cláusulas definidas no SLA. No entanto, a melhora de desempenho de um serviço não está diretamente relacionada com o aumento de todos os recursos que compõem uma VM. Talvez apenas o aumento no número de vCPUs seja o procedimento mais apropriado, economizando gastos e utilizando os recursos de forma mais eficiente. Essa abordagem é adotada pelo algoritmo Vertical.

Com base nos resultados das experimentações, verificou-se que o modelo de negócio aplicado pelo ambiente Vertical proporcionou o cumprimento das exigências definidas no SLA com pequenas alterações nos custos do serviço. Por essa razão, esse modelo mostrou-se a melhor opção para ambientes em nuvem que consideram a relação entre a sobrecarga imposta por contramedidas de segurança, desempenho e custo do serviço.

6.5 Considerações Finais

Neste Capítulo foi apresentada a metodologia utilizada para as avaliações propostas nesta tese de doutorado. Para a configuração dos ambientes simulados utilizados nas experimentações, foi inserido o módulo ReMM, bem como um módulo de tarifação na arquitetura ClouSim-BEQoS. Com o desenvolvimento desta arquitetura torna-se possível avaliar o impacto de diferentes demandas de requisições no desempenho dos serviços executados em nuvem, por meio da implementação de diferentes modelos de cargas de trabalho. Dessa forma, pode-se alterar a quantidade de recursos virtuais disponíveis durante o tempo de execução para atender as variações na demanda utilizando o ReMM, com impactos sobre o custo e na quantidade de requisições

atendidas.

Para as experimentações foram definidos dois cenários com diferentes sobrecargas impostas pelos mecanismos de segurança e com diferentes intervalos de tempo entre o envio de requisições. Assim, três ambientes em nuvem com diferentes modelos de negócio foram definidos para as avaliações das variáveis de resposta definidas com diferentes configurações dos experimentos, relacionando três métricas de destaque neste projeto: desempenho, sobrecarga de segurança e custo. Dessa forma, foi possível analisar os modelos de negócio para cada ambiente em diferentes cenários. O próximo Capítulo apresenta as conclusões finais desta tese, além das contribuições e trabalhos futuros.

CONCLUSÕES

7.1 Considerações Iniciais

Com a popularização de serviços oferecidos por meio de ambientes em nuvem, vários aspectos críticos de segurança despertam o interesse no âmbito acadêmico e industrial, na intenção de disponibilizar mecanismos eficientes no combate às mais variadas ameaças. Sabe-se que a aplicação de técnicas e metodologias de segurança influencia diretamente sobre o desempenho de um sistema, fazendo com que segurança e desempenho sejam muitas vezes duas grandezas inversamente proporcionais. Dessa forma, caso o provedor de serviços não gerencie a sua infraestrutura computacional de forma eficiente, a demanda por serviços poderá ser atendida sem a qualidade requerida pelos clientes e os recursos computacionais podem ser utilizados de forma ineficiente.

Há diversos trabalhos disponíveis na literatura que analisam o impacto de diferentes mecanismos de segurança sobre o desempenho dos mais variados ambientes em nuvem. No entanto, não foram encontrados trabalhos que consideram o impacto desse embate sobre o modelo de negócio.

Em face ao exposto, o objetivo deste trabalho de doutorado foi definir modelos de negócios para ambientes em nuvem a partir dos resultados da avaliação de desempenho de um serviço na aplicação de diferentes mecanismos de segurança, onde considerou-se a alteração dos recursos computacionais sempre que necessário. Com base nos resultados obtidos, foi possível demonstrar que em um ambiente em nuvem pode-se manter o desempenho do serviço mesmo com a sobrecarga imposta pelos mecanismos de segurança, por meio da alteração dos recursos computacionais virtualizados. No entanto, a alteração da quantidade dos recursos exerceu impacto direto sobre o custo pago pelo cliente.

7.2 Conclusões Finais

Para atingir o objetivo proposto foram realizados estudos de diversos mecanismos de segurança disponíveis na literatura que visam combater ameaças relacionadas ao acesso, armazenamento e manipulação de dados de serviços oferecidos por meio de máquinas virtuais. Com base nos resultados, foi possível caracterizar e quantificar a sobrecarga imposta por diferentes mecanismos de segurança sobre o desempenho de diferentes ambientes.

Visando contrapor a sobrecarga imposta pelas contramedidas de segurança, foi proposto o ReMM (*Resource Management Module*). O ReMM é um módulo dinâmico e autogerenciável responsável por alterar a quantidade de recursos alocados, de forma a garantir o SLA e o uso eficiente dos recursos a um preço justo. Para isso, dois algoritmos de escalabilidade são utilizados: vertical e horizontal. O primeiro altera a quantidade de vCPUs de uma máquina virtual, impactando o custo da VM na mesma proporção. No segundo, a quantidade de instâncias é alterada, exercendo um impacto mais expressivo sobre o custo, pois considera todos os recursos que compõem a instância, como quantidade de núcleos virtuais, de memória e de disco.

O ReMM foi inserido na arquitetura CloudSim-BEQoS, desenvolvida por [Pardo \(2012\)](#) e [Centurion \(2015\)](#), juntamente com um módulo de tarifação. Dessa forma, foi possível modelar cenários nos quais diversos usuários requisitavam serviços de compartilhamento de arquivos, utilizando ou não mecanismos de segurança. Diferentes intensidades de rajadas foram utilizadas para o envio das requisições que foram executadas em máquinas virtuais com configurações semelhantes às disponibilizadas pela Amazon. Um prazo de execução foi definido nos SLAs, de forma que o ReMM alterou a quantidade de recursos alocados em cada experimento sempre que necessário. Essas alterações refletiram nas variáveis de resposta Tempo Médio de Execução, Custos Médios Finais por VM e do Ambiente, Número de Requisições Atendidas e Quantidade de Recursos Utilizados. Dessa forma, foi possível confrontar a sobrecarga imposta pelos mecanismos de segurança sobre o desempenho do serviço em ambientes com e sem a presença do ReMM, avaliando os impactos sobre as variáveis de resposta em diferentes cenários.

O projeto contemplou um volume razoável de experimentos e foi desenvolvido em sua maioria em um ambiente simulado. Em todas as fases foram elaborados planejamentos dos experimentos, coletas de dados e análises dos resultados obtidos utilizando métodos estatísticos. Por meio das avaliações de desempenho realizadas foram definidos modelos de negócios para ambientes em nuvem em dois cenários utilizados para a execução de diferentes demandas de serviços na presença ou não de mecanismos de segurança.

7.3 Contribuições

A principal contribuição deste trabalho é a definição de modelos de negócio para ambientes de computação em nuvem que consideram atributos de QoS relacionados ao desempenho e a segurança. Para isso, inicialmente foi realizada uma caracterização e quantificação da sobrecarga imposta por diferentes mecanismos de segurança sobre o desempenho de um sistema em nuvem. Em seguida, foi proposto um módulo de gerenciamento de recursos chamado ReMM. Avaliações de desempenho de serviços fornecidos em dois cenários foram realizadas, nas quais foi possível confrontar as sobrecargas impostas pelos mecanismos de segurança e o ReMM, e avaliar os impactos sobre as variáveis de resposta. Com os resultados foram definidos modelos de negócio para três ambientes em nuvem. Assim, as contribuições e os principais pontos abordados com o desenvolvimento deste projeto são:

- Definição de modelos de negócio que relacionam a sobrecarga imposta pelos mecanismos de segurança, o desempenho e o custo. Os modelos de negócio permitem que os clientes e provedores analisem os comportamentos das variáveis de resposta na utilização de diferentes máquinas virtuais, executando requisições de serviços que podem exigir ou não a presença de mecanismos de segurança e de um módulo de gerenciamento de recursos para contrapor a sobrecarga imposta. Dessa forma, há a possibilidade de execuções de requisições com a utilização de contramedidas que garantam a segurança do serviço sem afetar o desempenho, uma vez que o ReMM pode aplicar dois algoritmos de escalabilidade para alterar a quantidade de recursos durante o tempo de execução, com impactos nos custos.
- Análise de dois cenários que consideram diferentes mecanismos de segurança e diferentes taxas de chegadas de requisições, bem como o módulo de gerenciamento de recursos. Por meio do planejamento dos experimentos foi possível avaliar o comportamento das variáveis de resposta, confrontando a sobrecarga imposta pelos mecanismos de segurança com as alterações na quantidade de recursos realizadas *on-the-fly* pelo ReMM, com impactos sobre os custos.
- Estudo da sobrecarga imposta por mecanismos de segurança sobre o desempenho de diferentes sistemas. Foram analisados vários trabalhos disponíveis na literatura que realizam uma avaliação de desempenho na aplicação de contramedidas de segurança. Dentre eles, os trabalhos de [Popa et al. \(2011\)](#) e [Casola et al. \(2011\)](#) foram utilizados na composição de dois cenários utilizados nas experimentações, os quais implicam em sobrecargas de 17% e 60%, respectivamente, sobre o desempenho do ambiente.
- Proposta e validação de um mecanismo para o gerenciamento de recursos do provedor. Os resultados das experimentações para a validação do ReMM foram submetidos e aceitos para publicação pelo *Journal Plos One*, o qual possui fator de impacto de 3.234.

BATISTA, B.G.; ESTRELLA, J.C.; FERREIRA, C.H.G.; LEITE FILHO, D.M.; NAKAMURA, L.H.V.; REIFF-MARGANIEC, S.; SANTANA, M.J.; SANTANA, R.H.C. *Performance Evaluation of Resource Management in Cloud Computing Environments*. Plos One, v. 10, 2015.

O ReMM possui dois algoritmos de escalabilidade: vertical e horizontal. Esses algoritmos alteram a quantidade de recursos com base nas análises para definir quais recursos computacionais e em que proporção eles devem ser alterados para atender diferentes demandas de serviços. Esses resultados foram publicados na *IEEE 10th World Congress on Services*, o qual possui Qualis B2:

BATISTA, B.G.; ESTRELLA, J.C.; SANTANA, M.J.; SANTANA, R.H.C.; REIFF-MARGANIEC, S. *Performance Evaluation in a Cloud with the Provisioning of Different Resources Configurations*. In: 2014 IEEE World Congress on Services (SERVICES), 2014, Anchorage, p. 309-316.

- Inserção do ReMM e de um módulo de tarifação na arquitetura CloudSim-BEQoS (PARDO, 2012) (CENTURION, 2015), permitindo a modelagem e simulação dos ambientes de testes com diferentes taxas de chegadas das requisições de serviço. Dessa forma, tem-se uma arquitetura que pode auxiliar não apenas as pesquisas desenvolvidas no LaSDPC, mas também as pesquisas desenvolvidas por outros grupos voltados ao planejamento de capacidade, modelos de desempenho, otimização de sistemas e avaliação de desempenho de sistemas computacionais em nuvem.

7.4 Trabalhos Futuros

O presente trabalho de doutorado não finaliza as possibilidades de estudos relacionadas às avaliações de desempenho dos serviços prestados sob demanda com a presença de mecanismos de segurança. Outros estudos devem ser conduzidos a partir dos resultados e constatações encontradas durante o desenvolvimento desta tese. Dentre os possíveis trabalhos futuros, destacam-se os seguintes estudos:

- **Adição de novos módulos ao ReMM:** outros módulos podem ser inseridos no ReMM, como um módulo para predição, análise e balanceamento de carga, bem como novas políticas de escalonamento que visam, por exemplo, a economia de energia.
- **Definição de um algoritmo de escalabilidade híbrido:** é possível desenvolver um novo algoritmo de escalabilidade com as propriedades dos algoritmos vertical e horizontal.
- **Otimização dos algoritmos de escalabilidade:** as alterações realizadas pelo ReMM devem ser as mais precisas possíveis, uma vez que a alteração constante dos recursos prejudica o desempenho do serviço. Dessa forma, técnicas de otimização de sistemas

podem ser aplicadas de forma a se obter um planejamento de capacidade e de demanda mais eficiente.

- **Experimentos com recursos físicos limitantes:** a utilização de recursos físicos limitantes possibilita a análise da taxa de requisições aceitas e rejeitadas pelo Controle de Admissão, variando a carga aplicada sobre o ambiente com e sem rajadas. Pode-se também incluir políticas de alocação de recursos que considerem a expansão de uma nuvem privada por meio da utilização de recursos fornecidos por uma nuvem pública.
- **Diferenciação de clientes:** é possível definir classes de clientes, os quais podem ser mais ou menos prioritários, exigindo recursos computacionais dedicados ou compartilhados.
- **Segurança como serviço:** o termo SecaaS (*Security as a Service*) vem sendo explorado nos últimos anos como uma forma de se prover mecanismos de segurança em forma de serviços. Pode-se então compor diferentes serviços utilizando diversos mecanismos de segurança disponíveis na literatura e definir novos modelos de negócio para essa classe de serviços.
- **Desenvolvimento de um protótipo real:** embora os experimentos finais apresentados nesta tese tenham sido executados em um ambiente simulado, está em fase de desenvolvimento um protótipo com características similares ao ReMM utilizando máquinas reais disponíveis no LASDPC e em provedores como a Amazon. Dessa forma, será possível comparar dados estatísticos oriundos de ambientes simulados e prototipados, além de análises sobre diferentes topologias de redes.

REFERÊNCIAS

- ABD, S. K.; AL-HADDAD, S.; HASHIM, F.; ABDULLAH, A. A review of cloud security based on cryptographic mechanisms. In: IEEE. **Biometrics and Security Technologies (ISBAST), 2014 International Symposium on**. [S.l.], 2014. p. 106–111. Citado na página 35.
- AGUIAR, E.; ZHANG, Y.; BLANTON, M. An overview of issues and recent developments in cloud computing and storage security. In: **High Performance Cloud Auditing and Applications**. [S.l.]: Springer, 2014. p. 3–33. Citado na página 45.
- AHUJA, S. P.; KOMATHUKATTIL, D. A survey of the state of cloud security. **Network and Communication Technologies**, v. 1, n. 2, p. p66, 2012. Citado na página 45.
- ALHAMAD, M.; DILLON, T.; CHANG, E. Service level agreement for distributed services: A review. In: IEEE. **Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on**. [S.l.], 2011. p. 1051–1054. Citado na página 22.
- ALHAMAZANI, K.; RANJAN, R.; MITRA, K.; RABHI, F.; JAYARAMAN, P. P.; KHAN, S. U.; GUABTNI, A.; BHATNAGAR, V. An overview of the commercial cloud monitoring tools: research dimensions, design issues, and state-of-the-art. **Computing**, Springer, v. 97, n. 4, p. 357–377, 2014. Citado na página 13.
- ALI, M.; KHAN, S. U.; VASILAKOS, A. V. Security in cloud computing: Opportunities and challenges. **Information Sciences**, Elsevier, v. 305, p. 357–383, 2015. Citado na página 46.
- ANURADHA, V.; SUMATHI, D. A survey on resource allocation strategies in cloud computing. In: IEEE. **Information Communication and Embedded Systems (ICICES), 2014 International Conference on**. [S.l.], 2014. p. 1–7. Citado na página 71.
- ARCHER, J.; BOEHME, A.; CULLINANE, D.; KURTZ, P.; PUHLMANN, N.; REAVIS, J. Top threats to cloud computing v1. 0. **Cloud Security Alliance**, 2010. Citado na página 32.
- ARDAGNA, D.; CASALE, G.; CIAVOTTA, M.; PÉREZ, J. F.; WANG, W. Quality-of-service in cloud computing: modeling techniques and their applications. **Journal of Internet Services and Applications**, Springer, v. 5, n. 1, p. 1–17, 2014. Citado 2 vezes nas páginas 2 e 18.
- ARMBRUST, M.; FOX, A.; GRIFFITH, R.; JOSEPH, A.; KATZ, R.; KONWINSKI, A.; LEE, G.; PATTERSON, D.; RABKIN, A.; STOICA, I. *et al.* A view of cloud computing. **Communications of the ACM**, ACM, v. 53, n. 4, p. 50–58, 2010. Citado na página 9.
- ATKINSON, B.; DELLA-LIBERA, G.; HADA, S.; HONDO, M.; HALLAM-BAKER, P.; KLEIN, J.; LAMACCHIA, B.; LEACH, P.; MANFERDELLI, J.; MARUYAMA, H. *et al.* Web services security (ws-security). **Specification, Microsoft Corporation**, 2002. Citado na página 63.
- BARHAM, P.; DRAGOVIC, B.; FRASER, K.; HAND, S.; HARRIS, T.; HO, A.; NEUGEBAUER, R.; PRATT, I.; WARFIELD, A. Xen and the art of virtualization. In: ACM. **ACM SIGOPS Operating Systems Review**. [S.l.], 2003. v. 37, n. 5, p. 164–177. Citado na página 17.

BATES, A.; MOOD, B.; VALAFAR, M.; BUTLER, K. Towards secure provenance-based access control in cloud environments. In: ACM. **Proceedings of the third ACM conference on Data and application security and privacy**. [S.l.], 2013. p. 277–284. Citado 3 vezes nas páginas 58, 60 e 62.

BELENKY, A.; ANSARI, N. Tracing multiple attackers with deterministic packet marking (dpm). In: IEEE. **Communications, Computers and signal Processing, 2003. PACRIM. 2003 IEEE Pacific Rim Conference on**. [S.l.], 2003. v. 1, p. 49–52. Citado na página 51.

BENSLIMANE, Y.; PLAISANT, M.; BERNARD, P.; BAHLI, B. Key challenges and opportunities in cloud computing and implications on service requirements: Evidence from a systematic literature review. In: IEEE. **Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on**. [S.l.], 2014. p. 114–121. Citado na página 9.

BERMAN, F.; FOX, G.; HEY, A. **Grid computing: making the global infrastructure a reality**. [S.l.]: Wiley, 2003. Citado na página 22.

BETHENCOURT, J.; SAHAI, A.; WATERS, B. Ciphertext-policy attribute-based encryption. In: IEEE. **Security and Privacy, 2007. SP'07. IEEE Symposium on**. [S.l.], 2007. p. 321–334. Citado na página 61.

BLAZE, M.; BLEUMER, G.; STRAUSS, M. Divertible protocols and atomic proxy cryptography. In: **Advances in Cryptology – EUROCRYPT'98**. [S.l.]: Springer, 1998. p. 127–144. Citado na página 61.

BREIVOLD, H. P.; CRNKOVIC, I.; RADOSEVIC, I.; BALATINAC, I. Architecting for the cloud: A systematic review. In: IEEE. **Computational Science and Engineering (CSE), 2014 IEEE 17th International Conference on**. [S.l.], 2014. p. 312–318. Citado 2 vezes nas páginas 2 e 10.

BRUNETTE, G.; MOGULL, R. Security guidance for critical areas of focus in cloud computing v2. 1. **Cloud Security Alliance**, p. 1–76, 2009. Citado 3 vezes nas páginas 32, 33 e 34.

BURNETT, S.; PAINE, S. **The RSA Security's Official Guide to Cryptography**. [S.l.]: McGraw-Hill, Inc., 2001. Citado na página 40.

BUYYA, R.; ABRAMSON, D.; GIDDY, J.; STOCKINGER, H. Economic models for resource management and scheduling in grid computing. **Concurrency and computation: practice and experience**, Wiley Online Library, v. 14, n. 13-15, p. 1507–1542, 2003. Citado na página 21.

BUYYA, R.; ABRAMSON, D.; VENUGOPAL, S. The grid economy. **Proceedings of the IEEE, IEEE**, v. 93, n. 3, p. 698–714, 2005. Citado na página 21.

CALHEIROS, R. N.; RANJAN, R.; BELOGLAZOV, A.; ROSE, C. A. D.; BUYYA, R. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. **Software: Practice and Experience**, Wiley Online Library, v. 41, n. 1, p. 23–50, 2011. Citado 2 vezes nas páginas 6 e 76.

CALHEIROS, R. N.; RANJAN, R.; BUYYA, R. Virtual machine provisioning based on analytical performance and qos in cloud computing environments. In: IEEE. **Parallel Processing (ICPP), 2011 International Conference on**. [S.l.], 2011. p. 295–304. Citado 2 vezes nas páginas 68 e 72.

CARISSIMI, A. Virtualização: da teoria a soluções. **Minicursos do Simpósio Brasileiro de Redes de Computadores—SBRC**, v. 2008, p. 173–207, 2008. Citado 4 vezes nas páginas 13, 14, 16 e 17.

CASALE, G.; MI, N.; CHERKASOVA, L.; SMIRNI, E. Dealing with burstiness in multi-tier applications: Models and their parameterization. **Software Engineering, IEEE Transactions on**, IEEE, v. 38, n. 5, p. 1040–1053, 2012. Citado na página 95.

CASOLA, V.; CUOMO, A.; RAK, M.; VILLANO, U. The cloudgrid approach: Security analysis and performance evaluation. **Future Generation Computer Systems**, Elsevier, 2011. Citado 10 vezes nas páginas 29, 59, 62, 93, 99, 100, 102, 112, 117 e 127.

CATTEDDU, D. Cloud computing: benefits, risks and recommendations for information security. **Web Application Security**, Springer, p. 17–17, 2010. Citado na página 32.

CENTURION, A. M. **Impacto das rajadas no desempenho de serviços executados em ambientes em nuvens**. Tese (Doutorado) — Universidade de São Paulo, 2015. Citado 7 vezes nas páginas 11, 93, 94, 95, 96, 126 e 128.

CHAKRAVARTHY, M. H.; KANNAN, E. A review on secured cloud computing environment. **American Journal of Applied Sciences**, v. 11, n. 8, p. 1224–1228, 2014. Citado na página 28.

CHANG, V.; WALTERS, R. J.; WILLS, G. The development that leads to the cloud computing business framework. **International Journal of Information Management**, Elsevier, v. 33, n. 3, p. 524–538, 2013. Citado na página 18.

CHARD, K. Drive: A distributed economic meta-scheduler for the federation of grid and cloud systems. Victoria University of Wellington, 2011. Citado 2 vezes nas páginas 2 e 18.

CHERKASOVA, L.; GUPTA, D.; VAHDAT, A. Comparison of the three cpu schedulers in xen. **SIGMETRICS Performance Evaluation Review**, v. 35, n. 2, p. 42–51, 2007. Citado na página 78.

CHHABRA, S.; ROGERS, B.; SOLIHIN, Y.; PRVULOVIC, M. Secureme: a hardware-software approach to full system security. In: ACM. **Proceedings of the international conference on Supercomputing**. [S.l.], 2011. p. 108–119. Citado 2 vezes nas páginas 59 e 62.

CHIEU, T.; MOHINDRA, A.; KARVE, A.; SEGAL, A. Dynamic scaling of web applications in a virtualized cloud computing environment. In: IEEE. **e-Business Engineering, 2009. ICEBE'09. IEEE International Conference on**. [S.l.], 2009. p. 281–286. Citado na página 10.

CHONKA, A.; XIANG, Y.; ZHOU, W.; BONTI, A. Cloud security defence to protect cloud computing against http-dos and xml-dos attacks. **Journal of network and computer applications**, Elsevier, v. 34, n. 4, p. 1097–1107, 2011. Citado 2 vezes nas páginas 9 e 51.

CHOUBEY, R.; DUBEY, R.; BHATTACHARJEE, J. A survey on cloud computing security, challenges and threats. **International Journal on Computer Science and Engineering (IJCSE)**, Citeseer, v. 3, n. 3, p. 1227–1231, 2011. Citado na página 44.

COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T. **Sistemas distribuídos-conceitos e projeto**. [S.l.]: Bookman, 2007. Citado 2 vezes nas páginas 37 e 41.

COUTINHO, E. F.; SOUSA, F. R. de C.; REGO, P. A. L.; GOMES, D. G.; SOUZA, J. N. de. Elasticity in cloud computing: a survey. **annals of telecommunications-Annales des télécommunications**, Springer, p. 1–21, 2015. Citado na página 67.

DURAO, F.; CARVALHO, J. F. S.; FONSEKA, A.; GARCIA, V. C. A systematic review on cloud computing. **The Journal of Supercomputing**, Springer, v. 68, p. 1321–1346, 2014. Citado na página 10.

ESTRELLA, J.; SANTANA, R. Wsarch: Uma arquitetura para a provisão de web services com qualidade de serviço. Biblioteca Digital de Teses e Dissertações da USP, 2011. Citado na página 2.

FELICIANO, G.; AGOSTINHO, L.; GUIMARÃES, E.; CARDOZO, E. Gerência de identidades federadas em nuvens: Enfoque na utilização de soluções abertas. **Short course, XI SBSeg, Brazil**, 2011. Citado na página 32.

FERNANDES, D. A.; SOARES, L. F.; GOMES, J. V.; FREIRE, M. M.; INÁCIO, P. R. Security issues in cloud environments: a survey. **International Journal of Information Security**, Springer, v. 13, n. 2, p. 113–170, 2014. Citado 3 vezes nas páginas 30, 32 e 45.

FOSTER, I.; ZHAO, Y.; RAICU, I.; LU, S. Cloud computing and grid computing 360-degree compared. In: IEEE. **Grid Computing Environments Workshop, 2008. GCE'08**. [S.l.], 2008. p. 1–10. Citado na página 10.

FOX, A.; GRIFFITH, R.; JOSEPH, A.; KATZ, R.; KONWINSKI, A.; LEE, G.; PATTERSON, D.; RABKIN, A.; STOICA, I. Above the clouds: A berkeley view of cloud computing. **Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS**, v. 28, p. 13, 2009. Citado na página 9.

GARG, S.; VERSTEEG, S.; BUYYA, R. Smicloud: A framework for comparing and ranking cloud services. In: IEEE. **Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on**. [S.l.], 2011. p. 210–218. Citado na página 19.

_____. A framework for ranking of cloud computing services. **Future Generation Computer Systems**, Elsevier, 2012. Citado na página 19.

GUO, L.; MCGOUGH, A.; AKRAM, A.; COLLING, D.; MARTYNIK, J.; KRZNARIC, M. Enabling qos for service-oriented workflow on grid. In: IEEE. **Computer and Information Technology, 2007. CIT 2007. 7th IEEE International Conference on**. [S.l.], 2007. p. 1077–1082. Citado na página 18.

GUZEK, M.; BOUVRY, P.; TALBI, E.-G. A survey of evolutionary computation for resource management of processing in cloud computing. **Computational Intelligence Magazine, IEEE**, IEEE, v. 10, n. 2, p. 53–67, 2015. Citado na página 68.

HASHEM, I. A. T.; YAQOOB, I.; ANUAR, N. B.; MOKHTAR, S.; GANI, A.; KHAN, S. U. The rise of “big data” on cloud computing: review and open research issues. **Information Systems**, Elsevier, v. 47, p. 98–115, 2015. Citado 2 vezes nas páginas 2 e 10.

HASSAN, S.; AL-JUMEILY, D.; HUSSAIN, A. Autonomic computing paradigm to support system's development. In: IEEE. **Developments in eSystems Engineering (DESE), 2009 Second International Conference on**. [S.l.], 2009. p. 273–278. Citado na página 20.

HE, H.; LI, R.; DONG, X.; ZHANG, Z. Secure, efficient and fine-grained data access control mechanism for p2p storage cloud. **Cloud Computing, IEEE Transactions on**, IEEE, v. 2, n. 4, p. 471–484, 2014. Citado 2 vezes nas páginas 61 e 62.

HE, J.; DONG, M.; OTA, K.; FAN, M.; WANG, G. Nssc: Self-service network security architecture for cloud computing. In: IEEE. **Computational Science and Engineering (CSE), 2014 IEEE 17th International Conference on**. [S.l.], 2014. p. 444–449. Citado 2 vezes nas páginas 61 e 62.

HEISER, J.; NICOLETT, M. Assessing the security risks of cloud computing. **Gartner Report**, 2008. Citado na página 32.

HU, H.; AHN, G.; KULKARNI, K. Anomaly discovery and resolution in web access control policies. In: ACM. **Proceedings of the 16th ACM symposium on Access control models and technologies**. [S.l.], 2011. p. 165–174. Citado na página 51.

HUGHES, J.; MALER, E. Security assertion markup language (saml) v2. 0 technical overview. **OASIS SSTC Working Draft sstc-saml-tech-overview-2.0-draft-08**, 2005. Citado na página 33.

HUSSAIN, H.; MALIK, S. U. R.; HAMEED, A.; KHAN, S. U.; BICKLER, G.; MIN-ALLAH, N.; QURESHI, M. B.; ZHANG, L.; YONGJI, W.; GHANI, N. *et al.* A survey on resource allocation in high performance distributed computing systems. **Parallel Computing**, Elsevier, v. 39, n. 11, p. 709–736, 2013. Citado na página 70.

HUU, T. T.; MONTAGNAT, J. Virtual resources allocation for workflow-based applications distribution on a cloud infrastructure. In: IEEE. **Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on**. [S.l.], 2010. p. 612–617. Citado 2 vezes nas páginas 69 e 71.

HWANG, J.; CHUANG, H.; HSU, Y.; WU, C. A business model for cloud computing based on a separate encryption and decryption service. In: IEEE. **Information Science and Applications (ICISA), 2011 International Conference on**. [S.l.], 2011. p. 1–7. Citado na página 23.

IBRAHIM, S.; HE, B.; JIN, H. Towards pay-as-you-consume cloud computing. In: IEEE. **Services Computing (SCC), 2011 IEEE International Conference on**. [S.l.], 2011. p. 370–377. Citado na página 22.

INOMATA, A.; MORIKAWA, T.; IKEBE, M.; OKAMOTO, Y.; NOGUCHI, S.; FUJIKAWA, K.; SUNAHARA, H.; RAHMAN, M. Proposal and evaluation of a dynamic resource allocation method based on the load of vms on iaas. In: IEEE. **New Technologies, Mobility and Security (NTMS), 2011 4th IFIP International Conference on**. [S.l.], 2011. p. 1–6. Citado na página 71.

IQBAL, W.; DAILEY, M. N.; ALI, I.; JANECEK, P.; CARRERA, D. Adaptive resource allocation for back-end mashup applications on a heterogeneous private cloud. In: IEEE. **Electrical Engineering/Electronics Computer Telecommunications and Information Technology (ECTI-CON), 2010 International Conference on**. [S.l.], 2010. p. 317–321. Citado na página 71.

JAIN, R. **The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling**. [S.l.]: New York, NY, USA, Wiley, 1991. Citado 3 vezes nas páginas 76, 80 e 101.

- JAVED, B.; BLOODSWORTH, P.; RASOOL, R. U.; MUNIR, K.; RANA, O. Cloud market maker: An automated dynamic pricing marketplace for cloud users. **Future Generation Computer Systems**, Elsevier, v. 54, p. 52–67, 2015. Citado na página 21.
- JENNINGS, B.; STADLER, R. Resource management in clouds: Survey and research challenges. **Journal of Network and Systems Management**, Springer, p. 1–53, 2014. Citado na página 68.
- JIN, S.; HUH, J. Secure mmu: Architectural support for memory isolation among virtual machines. In: IEEE. **Dependable Systems and Networks Workshops (DSN-W), 2011 IEEE/IFIP 41st International Conference on**. [S.l.], 2011. p. 217–222. Citado na página 53.
- KANDUKURI, B.; PATURI, V.; RAKSHIT, A. Cloud security issues. In: IEEE. **Services Computing, 2009. SCC'09. IEEE International Conference on**. [S.l.], 2009. p. 517–520. Citado na página 3.
- KARGER, P. Multi-level security requirements for hypervisors. In: IEEE. **Computer Security Applications Conference, 21st Annual**. [S.l.], 2005. p. 9–pp. Citado na página 53.
- KARUNAKARAN, S.; KRISHNASWAMY, V.; SUNDARRAJ, R. Decisions, models and opportunities in cloud computing economics: A review of research on pricing and markets. In: **Service Research and Innovation**. [S.l.]: Springer, 2014. p. 85–99. Citado na página 21.
- KHORSHED, M.; ALI, A.; WASIMI, S. A survey on gaps, threat remediation challenges and some thoughts for proactive attack detection in cloud computing. **Future Generation Computer Systems**, Elsevier, 2012. Citado na página 52.
- KOCHER, P.; LEE, R.; MCGRAW, G.; RAGHUNATHAN, A.; MODERATOR-RAVI, S. Security as a new dimension in embedded system design. In: ACM. **Proceedings of the 41st annual Design Automation Conference**. [S.l.], 2004. p. 753–760. Citado na página 37.
- KRISHNA, P. V. Honey bee behavior inspired load balancing of tasks in cloud computing environments. **Applied Soft Computing**, Elsevier, v. 13, n. 5, p. 2292–2303, 2013. Citado na página 18.
- KUROSE, J.; ROSS, K. **Redes de Computadores e a Internet: Uma Abordagem Top-Down**. [S.l.]: Pearson, 2013. Citado 8 vezes nas páginas 27, 34, 36, 38, 41, 42, 43 e 44.
- LANDWEHR, C. E. Computer security. **International Journal of Information Security**, Springer, v. 1, n. 1, p. 3–13, 2001. Citado na página 56.
- LAUREANO, M.; MAZIERO, C. Virtualização: Conceitos e aplicações em segurança. **Livro-Texto de Minicursos SBSeg**, p. 1–50, 2008. Citado 3 vezes nas páginas 14, 15 e 53.
- LEAVITT, N. Is cloud computing really ready for prime time. **Growth**, v. 27, n. 5, 2009. Citado na página 10.
- LEKKAS, D. Establishing and managing trust within the public key infrastructure. **Computer Communications**, Elsevier, v. 26, n. 16, p. 1815–1825, 2003. Citado na página 28.
- LI, J.; LI, B.; WO, T.; HU, C.; HUAI, J.; LIU, L.; LAM, K. Cyberguarder: A virtualization security assurance architecture for green cloud computing. **Future Generation Computer Systems**, Elsevier, v. 28, n. 2, p. 379–390, 2012. Citado na página 53.

- LI, Y.; LI, W.; JIANG, C. A survey of virtual machine system: Current technology and future trends. In: IEEE. **Electronic Commerce and Security (ISECS), 2010 Third International Symposium on**. [S.l.], 2010. p. 332–336. Citado 2 vezes nas páginas 16 e 17.
- LIN, J.-W.; CHEN, C.-H.; LIN, C.-Y. Integrating qos awareness with virtualization in cloud computing systems for delay-sensitive applications. **Future Generation Computer Systems**, Elsevier, v. 37, p. 478–487, 2014. Citado na página 17.
- LIU, Z.; WANG, S.; SUN, Q.; ZOU, H.; YANG, F. Cost-aware cloud service request scheduling for saas providers. **The Computer Journal**, Br Computer Soc, p. bxt009, 2013. Citado na página 18.
- LOMBARDI, F.; PIETRO, R. D. Secure virtualization for cloud computing. **Journal of Network and Computer Applications**, Elsevier, v. 34, n. 4, p. 1113–1122, 2011. Citado 5 vezes nas páginas 2, 32, 51, 59 e 62.
- LU, L.; CHERKASOVA, L.; PERSONE, V. de N.; MI, N.; SMIRNI, E. **AWAIT: Efficient overload management for busy multi-tier web services under bursty workloads**. [S.l.]: Springer, 2010. Citado na página 95.
- LU, X.; YIN, J.; CHEN, H.; ZHAO, X. An approach for bursty and self-similar workload generation. In: **Web Information Systems Engineering–WISE 2013**. [S.l.]: Springer, 2013. p. 347–360. Citado na página 95.
- MAGNUSSON, P. S.; CHRISTENSSON, M.; ESKILSON, J.; FORSGREN, D.; HALLBERG, G.; HOGBERG, J.; LARSSON, F.; MOESTEDT, A.; WERNER, B. Simics: A full system simulation platform. **Computer**, IEEE, v. 35, n. 2, p. 50–58, 2002. Citado na página 59.
- MAHBUB, K.; SPANOUDAKIS, G. Proactive sla negotiation for service based systems. In: IEEE. **Services (SERVICES-1), 2010 6th World Congress on**. [S.l.], 2010. p. 519–526. Citado na página 23.
- MANI, S.; RAO, S. Operating cost aware scheduling model for distributed servers based on global power pricing policies. In: ACM. **Proceedings of the Fourth Annual ACM Bangalore Conference**. [S.l.], 2011. p. 12. Citado na página 23.
- MANVI, S. S.; SHYAM, G. K. Resource management for infrastructure as a service (iaas) in cloud computing: A survey. **Journal of Network and Computer Applications**, Elsevier, v. 41, p. 424–440, 2014. Citado 3 vezes nas páginas 17, 24 e 71.
- MARTY, R. Cloud application logging for forensics. In: ACM. **Proceedings of the 2011 ACM Symposium on Applied Computing**. [S.l.], 2011. p. 178–184. Citado na página 51.
- MATHER, T.; KUMARASWAMY, S.; LATIF, S. **Cloud security and privacy: an enterprise perspective on risks and compliance**. [S.l.]: O'Reilly Media, Incorporated, 2009. Citado na página 34.
- MATTOS, D. Virtualização: Vmware e xen. **Universidade Federal do Rio de Janeiro**, 2008. Citado na página 16.
- MELL, P.; GRANCE, T. The nist definition of cloud computing (draft). **NIST special publication**, v. 800, p. 145, 2011. Citado 2 vezes nas páginas 1 e 10.

MENASCÉ, D. Virtualization: Concepts, applications, and performance modeling. In: COMPUTER MEASUREMENT GROUP; 1997. **CMG-CONFERENCE-**. [S.l.], 2005. v. 1, p. 407. Citado na página 14.

MENASCE, D. A.; ALMEIDA, V. A.; DOWDY, L. W.; DOWDY, L. **Performance by design: computer capacity planning by example**. [S.l.]: Prentice Hall Professional, 2004. Citado na página 72.

MENKEN, I.; BLOKDIJK, G. **Cloud Computing Foundation Complete Certification Kit-Study Guide Book and Online Course**. [S.l.]: Emereo Pty Ltd, 2010. Citado na página 13.

MI, N.; CASALE, G.; CHERKASOVA, L.; SMIRNI, E. Sizing multi-tier systems with temporal dependence: benchmarks and analytic models. **Journal of Internet Services and Applications**, Springer, v. 1, n. 2, p. 117–134, 2010. Citado na página 95.

MODI, C.; PATEL, D.; BORISANIYA, B.; PATEL, H.; PATEL, A.; RAJARAJAN, M. A survey of intrusion detection techniques in cloud. **Journal of Network and Computer Applications**, Elsevier, v. 36, n. 1, p. 42–57, 2013. Citado na página 52.

MORENO, E.; PEREIRA, F.; CHIARAMONTE, R. Criptografia em software e hardware. **São Paulo: Novatec**, 2005. Citado 2 vezes nas páginas 36 e 39.

MUSTAFA, S.; NAZIR, B.; HAYAT, A.; MADANI, S. A. *et al.* Resource management in cloud computing: Taxonomy, prospects, and challenges. **Computers & Electrical Engineering**, Elsevier, 2015. Citado na página 68.

PACHECO-SANCHEZ, S.; CASALE, G.; SCOTNEY, B.; MCCLEAN, S.; PARR, G.; DAWSON, S. Markovian workload characterization for qos prediction in the cloud. In: IEEE. **Cloud Computing (CLOUD), 2011 IEEE International Conference on**. [S.l.], 2011. p. 147–154. Citado na página 95.

PARDO, M. H. d. S. **Análise e Projeto de um broker como agente de intermediação e QoS em uma nuvem computacional híbrida**. Tese (Doutorado) — Qualificação de Doutorado. Universidade de São Paulo, 2012. Citado 5 vezes nas páginas 93, 94, 95, 126 e 128.

PEARCE, M.; ZEADALLY, S.; HUNT, R. Virtualization: Issues, security threats, and solutions. **ACM Computing Surveys (CSUR)**, ACM, v. 45, n. 2, p. 17, 2013. Citado na página 45.

PEIXOTO, M. L. M. **Oferecimento de QoS para computação em nuvens por meio de metaescalonamento**. Tese (Doutorado) — Universidade de São Paulo, 2012. Citado na página 18.

PENG, J.; ZHANG, X.; LEI, Z.; ZHANG, B.; ZHANG, W.; LI, Q. Comparison of several cloud computing platforms. In: IEEE. **Information Science and Engineering (ISISE), 2009 Second International Symposium on**. [S.l.], 2009. p. 23–27. Citado na página 10.

POPA, R. A.; LORCH, J. R.; MOLNAR, D.; WANG, H. J.; ZHUANG, L. Enabling security in cloud storage slas with cloudproof. In: **USENIX Annual Technical Conference**. [S.l.: s.n.], 2011. v. 242. Citado 8 vezes nas páginas 60, 62, 64, 93, 99, 100, 102 e 127.

PROVOS, N.; RAJAB, M. A.; MAVROMMATIS, P. Cybercrime 2.0: when the cloud turns dark. **Communications of the ACM**, ACM, v. 52, n. 4, p. 42–47, 2009. Citado na página 31.

REDDY, P. V. V.; RAJAMANI, L. Performance evaluation of hypervisors in the private cloud based on system information using sigar framework and for system workloads using passmark. **International Journal of Advanced Science and Technology**, v. 70, p. 17–32, 2014. Citado na página 16.

REESE, G. **Cloud Application Architectures: Building Applications and Infrastructure in the Cloud**. [S.l.]: O'Reilly Media, Incorporated, 2009. Citado na página 9.

REN, L.; ZHANG, L.; TAO, F.; ZHAO, C.; CHAI, X.; ZHAO, X. Cloud manufacturing: from concept to practice. **Enterprise Information Systems**, Taylor & Francis, v. 9, n. 2, p. 186–209, 2015. Citado na página 21.

RESCORLA, E. **SSL and TLS: designing and building secure systems**. [S.l.]: Addison-Wesley Reading, 2001. Citado na página 43.

RIMAL, B.; CHOI, E.; LUMB, I. A taxonomy and survey of cloud computing systems. In: IEEE. **INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on**. [S.l.], 2009. p. 44–51. Citado 3 vezes nas páginas 11, 13 e 17.

RISTENPART, T.; TROMER, E.; SHACHAM, H.; SAVAGE, S. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In: ACM. **Proceedings of the 16th ACM conference on Computer and communications security**. [S.l.], 2009. p. 199–212. Citado na página 56.

RITTINGHOUSE, J. **Cloud computing: implementation, management, and security**. [S.l.]: CRC, 2009. Citado 2 vezes nas páginas 9 e 11.

RODERO-MERINO, L.; VAQUERO, L. M.; CARON, E.; MURESAN, A.; DESPREZ, F. Building safe paas clouds: A survey on security in multitenant software platforms. **computers & security**, Elsevier, v. 31, n. 1, p. 96–108, 2012. Citado na página 45.

ROSE, R. Survey of system virtualization techniques. **Collections**, 2004. Citado na página 14.

ROSENBERG, J.; REMY, D. **Securing Web Services with WS-Security: Demystifying WS-Security, WS-Policy, SAML, XML Signature and XML Encryption**. [S.l.]: Pearson Higher Education, 2004. Citado 5 vezes nas páginas 11, 35, 36, 39 e 40.

SAMIMI, P.; PATEL, A. Review of pricing models for grid & cloud computing. In: IEEE. **Computers & Informatics (ISCI), 2011 IEEE Symposium on**. [S.l.], 2011. p. 634–639. Citado na página 22.

SANKA, S.; HOTA, C.; RAJARAJAN, M. Secure data access in cloud computing. In: IEEE. **Internet Multimedia Services Architecture and Application (IMSAA), 2010 IEEE 4th International Conference on**. [S.l.], 2010. p. 1–6. Citado 2 vezes nas páginas 58 e 62.

SHAMSOLMOALI, P.; ALAM, M. A. Ensuring data security and performance evaluation in cloud computing. In: **Intelligent Computing, Communication and Devices**. [S.l.]: Springer, 2015. p. 415–423. Citado 2 vezes nas páginas 2 e 32.

SHARKH, M. A.; JAMMAL, M.; SHAMI, A.; OUDA, A. Resource allocation in a network-based cloud computing environment: design challenges. **Communications Magazine, IEEE**, IEEE, v. 51, n. 11, p. 46–52, 2013. Citado na página 70.

- SRINIVASAMURTHY, S.; LIU, D. Survey on cloud computing security. In: **Proc. Conf. on Cloud Computing, CloudCom**. [S.l.: s.n.], 2010. v. 10. Citado na página 44.
- STALLINGS, W.; VIEIRA, D. **Criptografia e segurança de redes: princípios e práticas**. [S.l.]: Pearson Prentice Hall, 2008. Citado 6 vezes nas páginas 29, 35, 37, 38, 41 e 44.
- STANTCHEV, V.; SCHRÖPFER, C. Negotiating and enforcing qos and slas in grid and cloud computing. **Advances in Grid and Pervasive Computing**, Springer, p. 25–35, 2009. Citado na página 18.
- STEINER, M.; TSUDIK, G.; WAIDNER, M. Diffie-hellman key distribution extended to group communication. In: **ACM. Proceedings of the 3rd ACM conference on Computer and communications security**. [S.l.], 1996. p. 31–37. Citado na página 58.
- SUBASHINI, S.; KAVITHA, V. A survey on security issues in service delivery models of cloud computing. **Journal of Network and Computer Applications**, Elsevier, v. 34, n. 1, p. 1–11, 2011. Citado 5 vezes nas páginas 1, 2, 32, 33 e 44.
- TANENBAUM, A. Computer networks. **Prentice Hall**, 2003. Citado 9 vezes nas páginas 13, 27, 35, 36, 37, 38, 40, 41 e 43.
- TANENBAUM, A. S. Computer networks, 5-th edition. **ed: Prentice Hall**, 2011. Citado na página 2.
- VAQUERO, L.; RODERO-MERINO, L.; MORÁN, D. Locking the sky: a survey on iaas cloud security. **Computing**, Springer, v. 91, n. 1, p. 93–118, 2011. Citado 4 vezes nas páginas 2, 32, 33 e 45.
- VARADHARAJAN, V.; TUPAKULA, U. Security as a service model for cloud environment. **Network and Service Management, IEEE Transactions on**, IEEE, v. 11, n. 1, p. 60–75, 2014. Citado 2 vezes nas páginas 61 e 62.
- VASILOU, N. Overview of internet qos and web server qos. **UWO Reading Course Paper**, p. 1–37, 2000. Citado na página 18.
- VEGESNA, S. **IP quality of service**. [S.l.]: Cisco Systems, 2001. Citado na página 17.
- VELEV, D.; ZLATEVA, P. Cloud infrastructure security. **Open Research Problems in Network Security**, Springer, p. 140–148, 2011. Citado 6 vezes nas páginas 3, 10, 11, 28, 32 e 34.
- VIJAYALAKSHMI, P.; RAJA, K. Performance analysis of rsa and ecc in identity-based authenticated new multiparty key agreement protocol. In: **IEEE. Computing, Communication and Applications (ICCCA), 2012 International Conference on**. [S.l.], 2012. p. 1–5. Citado na página 37.
- VILLEGAS, D.; BOBROFF, N.; RODERO, I.; DELGADO, J.; LIU, Y.; DEVARAKONDA, A.; FONG, L.; SADJADI, S. M.; PARASHAR, M. Cloud federation in a layered service model. **Journal of Computer and System Sciences**, Elsevier, 2012. Citado na página 23.
- WANG, C.; WANG, Q.; REN, K.; LOU, W. Privacy-preserving public auditing for data storage security in cloud computing. In: **IEEE. INFOCOM, 2010 Proceedings IEEE**. [S.l.], 2010. p. 1–9. Citado na página 3.

- WANG, J.; ZHAO, Y.; JIANG, S.; LE, J. Providing privacy preserving in cloud computing. In: IEEE. **Test and Measurement, 2009. ICTM'09. International Conference on**. [S.l.], 2009. v. 2, p. 213–216. Citado na página [31](#).
- WANG, L.; KUNZE, M.; TAO, J.; LASZEWSKI, G. von. Towards building a cloud for scientific applications. **Advances in Engineering Software**, Elsevier, v. 42, n. 9, p. 714–722, 2011. Citado na página [10](#).
- WANG, L.; TAO, J.; KUNZE, M.; CASTELLANOS, A.; KRAMER, D.; KARL, W. Scientific cloud computing: Early definition and experience. In: IEEE. **High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on**. [S.l.], 2008. p. 825–830. Citado na página [10](#).
- WEBER, T. Um roteiro para exploração dos conceitos básicos de tolerância a falhas. **Relatório técnico, Instituto de Informática UFRGS**, 2002. Citado 2 vezes nas páginas [2](#) e [18](#).
- WEI, L.; ZHU, H.; CAO, Z.; DONG, X.; JIA, W.; CHEN, Y.; VASILAKOS, A. V. Security and privacy for storage and computation in cloud computing. **Information Sciences**, Elsevier, v. 258, p. 371–386, 2014. Citado 2 vezes nas páginas [53](#) e [58](#).
- WEI, Y.; BLAKE, M. B. Proactive virtualized resource management for service workflows in the cloud. **Computing**, Springer, p. 1–16, 2014. Citado na página [13](#).
- WEINGÄRTNER, R.; BRÄSCHER, G. B.; WESTPHALL, C. B. Cloud resource management: A survey on forecasting and profiling models. **Journal of Network and Computer Applications**, Elsevier, v. 47, p. 99–106, 2015. Citado na página [21](#).
- WENG, C.; LUO, Y.; LI, M.; LU, X. A blp-based access control mechanism for the virtual machine system. In: IEEE. **Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for**. [S.l.], 2008. p. 2278–2282. Citado na página [53](#).
- WHITAKER, A.; SHAW, M.; GRIBBLE, S. Denali: A scalable isolation kernel. In: ACM. **Proceedings of the 10th workshop on ACM SIGOPS European workshop**. [S.l.], 2002. p. 10–15. Citado na página [17](#).
- WU, L.; GARG, S. K.; BUYYA, R. Sla-based resource allocation for software as a service provider (saas) in cloud computing environments. In: IEEE. **Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on**. [S.l.], 2011. p. 195–204. Citado na página [71](#).
- XIAO, Z.; XIAO, Y. Security and privacy in cloud computing. **Communications Surveys & Tutorials, IEEE**, IEEE, v. 15, n. 2, p. 843–859, 2013. Citado na página [45](#).
- YAN, L.; RONG, C.; ZHAO, G. Strengthen cloud computing security with federal identity management using hierarchical identity-based cryptography. In: **Cloud Computing**. [S.l.]: Springer, 2009. p. 167–177. Citado na página [51](#).
- YANG, K.; JIA, X. Data storage auditing service in cloud computing: challenges, methods and opportunities. **World Wide Web**, Springer, v. 15, n. 4, p. 409–428, 2012. Citado na página [51](#).
- YANG, X.; NASSER, B.; SURRIDGE, M.; MIDDLETON, S. A business-oriented cloud federation model for real-time online interactive applications. **Future Generation Computer Systems**, Elsevier, 2012. Citado na página [23](#).

- YIN, J.; LU, X.; CHEN, H.; ZHAO, X.; XIONG, N. N. System resource utilization analysis and prediction for cloud based applications under bursty workloads. **Information Sciences**, Elsevier, v. 279, p. 338–357, 2014. Citado na página 95.
- ZAMAN, S.; GROSU, D. Combinatorial auction-based allocation of virtual machine instances in clouds. **Journal of Parallel and Distributed Computing**, Elsevier, v. 73, n. 4, p. 495–508, 2013. Citado na página 18.
- ZHANG, F.; CHEN, J.; CHEN, H.; ZANG, B. Cloudvisor: retrofitting protection of virtual machines in multi-tenant cloud with nested virtualization. In: ACM. **Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles**. [S.l.], 2011. p. 203–216. Citado 2 vezes nas páginas 60 e 62.
- ZHAO, Y.; XIE, Y.; YU, F.; KE, Q.; YU, Y.; CHEN, Y.; GILLUM, E. Botgraph: Large scale spamming botnet detection. In: NSDI. [S.l.: s.n.], 2009. v. 9, p. 321–334. Citado na página 31.
- ZHOU, A. C.; HE, B. Simplified resource provisioning for workflows in iaas clouds. In: IEEE. **Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on**. [S.l.], 2014. p. 650–655. Citado na página 70.
- ZHOU, M.; ZHANG, R.; XIE, W.; QIAN, W.; ZHOU, A. Security and privacy in cloud computing: A survey. In: IEEE. **Semantics Knowledge and Grid (SKG), 2010 Sixth International Conference on**. [S.l.], 2010. p. 105–112. Citado na página 45.
- ZHOU, X.; JIANG, C.-J. Autonomic performance and power control on virtualized servers: Survey, practices, and trends. **Journal of Computer Science and Technology**, Springer, v. 29, n. 4, p. 631–645, 2014. Citado na página 16.
- ZISSIS, D.; LEKKAS, D. Addressing cloud computing security issues. **Future Generation Computer Systems**, Elsevier, v. 28, n. 3, p. 583–592, 2012. Citado 4 vezes nas páginas 27, 29, 30 e 52.