Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

Gerência de Nuvens Computacionais Considerando Diferentes Classes de Serviço

Marcus Williams Aquino de Carvalho

Tese submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Doutor em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Redes de Computadores e Sistemas Distribuídos

Francisco Vilar Brasileiro
(Orientador)

Campina Grande, Paraíba, Brasil

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

C331g Carvalho, Marcus Williams Aquino de.

Gerência de nuvens computacionais considerando diferentes classes de serviço / Marcus Williams Aquino de Carvalho. – Campina Grande, 2016.

143 f. : il.

Tese (Doutorado em Ciência da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2016.

"Orientação: Prof. Dr. Francisco Vilar Brasileiro".

Referências.

1. Computação na Nuvem. 2. Gerência de Recursos. 3. Qualidade de Serviço. 4. Modelagem de Desempenho. I. Brasileiro, Francisco Vilar. II. Título.

CDU 004.7 (043)

"GERÊNCIA DE NUVENS COMPUTACIONAIS CONSIDERANDO DIFERENTES CLASSES DE SERVIÇO"

MARCUS WILLIAMS AQUINO DE CARVALHO

TESE APROVADA EM 01/03/2016

FRANCISCO VILAR BRASILEIRO, Ph.D, UFCG
Orientador(a)

ANDREY ELÍSIO MONTEIRO BRITO, Dr., UFCG
Examinador(a)

JOSÉ ANTÃO BELTRÃO MOURA, Ph.D, UFCG
Examinador(a)

RENATO FONTOURA DE GUSMÃO CERQUEIRA, D.Sc., IBM
Examinador(a)

NABOR DAS CHAGAS MENDONÇA, D.Sc., UNIFOR
Examinador(a)

CAMPINA GRANDE - PB

Resumo

O modelo de nuvens computacionais de infraestrutura como serviço (IaaS, na sigla em inglês) vem crescendo significativamente nos últimos anos. Este aumento em sua adoção trouxe uma grande diversidade de perfis de usuários, com diferentes tipos de aplicação, requisitos e orçamentos. Para satisfazer essas necessidades diversas, provedores de IaaS podem oferecer múltiplas classes de serviço, com diferentes preços e objetivos de nível de serviço (SLOs, na sigla em inglês) definidos para elas. Porém, gerenciar nuvens considerando múltiplas classes não é trivial, pois decisões de gerência de recursos podem gerar impactos diferentes dependendo de como cada classe é afetada. Além disso, a elasticidade da demanda e as incertezas da oferta de recursos típicas deste ambiente tornam ainda mais difícil o cumprimento dos diferentes SLOs mantendo uma alta utilização e um baixo custo. Nesta tese, investigou-se a hipótese de que quando provedores de nuvem de IaaS realizam uma gerência de recursos adequada, oferecendo múltiplas classes de serviço e cumprindo suas metas de qualidade de serviço (QoS, na sigla em inglês), eles obtêm uma alta utilização de seus recursos e aumentam a sua receita. Verificou-se que esta hipótese é verdadeira para os diversos cenários de nuvem avaliados neste trabalho, para os quais foram demonstradas grandes vantagens de se oferecer múltiplas classes de serviço na nuvem. Porém, observou-se que para ter esses benefícios é necessário realizar uma gerência de recursos eficiente, de tal forma que as garantias de QoS para as diferentes classes sejam definidas adequadamente e cumpridas pelo provedor. Desta forma, nesta tese também mostrou-se como provedores de IaaS podem realizar uma gerência de recursos adequada para múltiplas classes de serviço. Para isto, foram propostos e avaliados nesta tese: (1) um método baseado em predição para planejar a capacidade e as garantias de QoS de uma nova classe introduzida em uma nuvem de IaaS existente, com base na capacidade excedente da nuvem; (2) um modelo de controle de admissão baseado em predição, que permite ao provedor oferecer diferentes classes de serviço e cumprir SLOs de disponibilidade de VM para elas; e (3) um modelo analítico de planejamento de capacidade da nuvem que estima métricas de QoS para cada classe oferecida em diferentes cenários, e que busca encontrar a capacidade mínima necessária para cumprir as metas de taxa de admissão e disponibilidade de VM definidas para cada classe.

Abstract

Infrastructure as a Service (IaaS) is a cloud computing model that has been growing significantly in recent years. This increasingly adoption attracted users with different types of applications, requirements and budget to the cloud. To satisfy different users's needs, IaaS providers can offer multiple service classes with different pricing and Service Level Objectives (SLOs) defined for them. However, managing such clouds considering multiple service classes is not trivial, because resource management decisions may have different consequences depending on how each class is affected. Moreover, the demand elasticity and uncertain resource availability typically seen in cloud environments turns even more difficult for providers to fulfill different SLOs while having a high utilization and low infrastructure costs. In this thesis, we investigate the hypothesis that when IaaS cloud providers make an adequate resource management, offering multiple service classes and fulfilling their Quality of Service (QoS) guarantees, they achieve a high resource utilization and increase their revenue. We demonstrate that this hypothesis is true for the many different scenarios evaluated in this work, which shows great advantages on offering multiple service classes in the cloud. However, we also observe that cloud providers need an efficient resource management in order to have these benefits, in a way they can define and fulfill adequate SLOs for different classes. Thus, we also show in this thesis how IaaS providers can make adequate resource management decisions for multiple service classes, by proposing and evaluating: (1) a predictive method to plan the capacity and QoS guarantees of a new service class introduced in an existing IaaS cloud, based on unused resources; (2) a prediction-based admission control model that allows the provider to offer multiple classes and fulfill VM availability SLOs for them; and (3) a capacity planning analytical model that estimates QoS metrics for each class in different scenarios, and aims to find the minimum resource capacity required to fulfill VM availability and admission rate SLOs for each class.

Agradecimentos

Agradeço primeiramente à minha esposa, Raquel, por estar sempre ao meu lado durante toda essa jornada, me dando um suporte enorme em todos os aspectos. Além de tudo, é uma grande inspiração tanto como pessoa quanto profissional para mim. Agradeço também a Guilherme, meu enteado, por trazer alegria à casa nos momentos em que os adultos só pensavam em trabalho. Agradeço aos meus pais, Rebeca e Marcão, pelo esforço que fizeram para me darem a educação que me permitiu chegar onde cheguei. Também agradeço aos meus irmãos, Rafael e Nathália, pois independente da distância sei que posso contar com vocês. Agradeço à minha avó Teresa (*in memorian*) que me mostrou a importância de buscar uma educação de qualidade, me fazendo sair da zona de conforto para estudar em outra cidade, além de ser um grande exemplo de pessoa e profissional juntamente com meu avô Aecio (*in memorian*). Agradeço aos meus avós paternos, Waldemir e Eliza (*in memorian*), pela alegria e orgulho que me transmitiam à cada nova conquista. Agradeço também a outros familiares que de alguma forma me ajudaram durante o doutorado.

Agradeço imensamente ao meu orientador, Fubica, por me guiar tão bem nesse processo de aprendizado. Agradeço também ao professor Walfredo, por todas as discussões técnicas e filosóficas durante meu estágio na Google, e ao professor Daniel Menascé por toda a atenção que teve comigo durante meu estágio de doutorado sanduíche, além da grande contribuição que deu à minha formação. Agradeço aos meus amigos, tanto os mais próximos quanto os fisicamente mais distantes. Aos amigos do LSD/UFCG, pelas discussões técnicas e, não menos importante, pelos momentos de diversão. Agradeço aos colegas professores da UFPB/Rio Tinto, por me darem uma força grande e facilitarem a minha vida enquanto eu me desdobrava para dar aulas e fazer o doutorado ao mesmo tempo. Agradeço aos alunos de Rio Tinto, por darem um sentido a todo o esforço feito no doutorado, me mostrando a utilidade de repassar meu conhecimento à nova geração e ajudar a mudar a realidade de pessoas menos favorecidas, em uma região pouco favorecida, em um estado pouco favorecido, em um país pouco favorecido. Por fim, agradeço ao povo brasileiro por financiar a minha educação, mesmo sem ter as mesmas oportunidades que tive. Espero poder retribuir ao país o suporte que tive durante todos esses anos.

Conteúdo

1 Introdução **1**

1.1 Motivação e relevância 1

1.2 Objetivos 7

1.3 Contribuições 8

1.4 Organização do documento 9

2 Fundamentação teórica **11**

2.1 Computação na nuvem 11

2.2 Gerência de nuvens computacionais de IaaS 14

3 Revisão bibliográfica **17**

3.1 Gerência de recursos de nuvens computacionais 17

3.1.1 Planejamento de capacidade e controle de admissão 18

3.1.2 Gerência de recursos baseada em garantias de QoS 21

3.1.3 Alocação de recursos e balanceamento de carga 23

3.1.4 Gerência de recursos da nuvem na perspectiva do usuário 24

3.2 Caracterização, modelagem e predição da carga de trabalho da nuvem . . . 25

3.2.1 Caracterização e modelagem da carga de trabalho 25

3.2.2 Predição da carga de trabalho 26

3.3 Conclusões do capítulo 28

4 Modelo do sistema e perguntas de pesquisa **29**

4.1 Modelo do provedor de nuvem de IaaS 29

4.2 Perguntas de pesquisa 32

4.2.1 Definição das classes de serviço 35

4.2.2 Planejamento de capacidade 36
4.2.3 Controle de admissão 36

5 Planejando uma nova classe para a nuvem com base em sobras de recursos 38
5.1 Introdução 38
5.2 Descrição do problema 41
5.3 Análise da carga de trabalho da nuvem 44
5.3.1 Análise da utilização dos limites de alocação 44
5.3.2 Análise da capacidade excedente 46
5.4 Modelo de predição 47
5.4.1 Previsão de séries temporais 48
5.4.2 Intervalos de confiança para predições 49
5.4.3 Ciclos de predição 50
5.5 Avaliação 51
5.5.1 Metodologia 51
5.5.2 Resultados das predições 56
5.5.3 Análise do planejamento da reserva 59
5.5.4 Análise da receita potencial 61
5.6 Conclusões do estudo 68

6 Controle de admissão em nuvens de IaaS com múltiplas classes 70
6.1 Introdução 70
6.2 Formalização do problema 72
6.3 Caracterização da carga de trabalho e capacidade da nuvem 76
6.4 Modelo do controle de admissão 84
6.5 Metodologia de avaliação 87
6.5.1 Heurísticas de controle de admissão 87
6.5.2 Ambiente de simulação 88
6.6 Resultados 95
6.6.1 Cenário base 95
6.6.2 Análise de sensibilidade para a capacidade da nuvem 97
6.6.3 Análise de sensibilidade para a carga da nuvem 101

6.6.4 Análise de sensibilidade para SLOs 104
6.7 Conclusões do estudo 108

7 Planejamento de capacidade em nuvens de IaaS com múltiplas classes 110
7.1 Introdução 111
7.2 Formalização do problema 112
7.3 Modelo do planejamento de capacidade da nuvem 114
7.3.1 Aproximações de difusão para filas G/GI/c/K 114
7.3.2 Formulação do modelo 117
7.4 Validação do modelo 121
7.4.1 Instanciando o modelo 121
7.4.2 Resultados 123
7.5 Conclusões do estudo 128

8 Considerações finais e trabalhos futuros 130
8.1 Conclusões 130
8.2 Limitações do trabalho 132
8.3 Trabalhos futuros 133

Lista de Símbolos

AIC – *Akaike Information Criterion*

ARIMA – *Auto Regressive Integrated Moving Average*

CDF – *Cumulative Distribution Function*

CPU – *Central Processing Unit*

DTW – *Dynamic Time Warping*

DVS – *Dynamic Voltage Scaling*

EC2 – *Elastic Computing Cloud*

ETS – *ExponenTial Smoothing*

FDA – *Função de Distribuição Acumulada*

FFT – *Fast Fourier Transform*

GCE – *Google Compute Engine*

IaaS – *Infrastructure as a Service*

MLE – *Maximum Likelihood Estimator*

PaaS – *Platform as a Service*

QoS – *Quality of Service*

SaaS – *Software as a Service*

SLA – *Service Level Agreement*

SLO – *Service Level Objective*

TI – *Tecnologia da Informação*

VM – *Virtual Machine*

Lista de Figuras

1.1 Triângulo da gerência de recursos da nuvem. Provedores podem priorizar apenas dois dos três eixos do triângulo ao oferecer um serviço na nuvem. . 5

2.1 Organização em camadas dos modelos mais populares de nuvens computacionais. 12

4.1 Diagrama de estado para requisições de VM no modelo de nuvem de IaaS. . 30
4.2 Ilustração do mecanismo de controle de admissão baseado em quotas. . . . 33
4.3 Ilustração do problema da gerência de recursos para uma nuvem computacional, mostrando a capacidade da nuvem disponível no tempo e a reserva planejada para diferentes classes de serviço. 34

5.1 Como a nuvem é gerenciada e os diferentes tipos de sobra de recursos. . . . 42
5.2 Função de Distribuição Acumulada (FDA) empírica para a utilização dos limites de alocação de CPU, agrupados para usuários individuais e para todo o cluster, medidos a cada 5 minutos. 45
5.3 Sobras de CPU no tempo, normalizadas pela capacidade total, medidas em intervalos de 5 minutos. 47
5.4 Função de Distribuição Acumulada (FDA) do total de sobras para CPU, memória e disco, medidos em intervalos de 5 minutos. 48
5.5 Ilustração do modelo para uma predição realizada no tempo t para uma janela Δ e para diferentes níveis de confiança. 51
5.6 Função de Distribuição Acumulada (FDA) dos erros das predições para as sobras de CPU em janelas de 6 meses, para 4 técnicas de predição. 57

5.7 Sobras de CPU no tempo observadas e estimadas com a técnica de predição ETS, para diferentes níveis de confiança. 58
5.8 Quantidade de recursos estimada para a classe *Economy* e suas disponibilidades nas janelas de predição de 6 meses. 61
5.9 Comparação das receitas ao oferecer a classe *Economy* e sua disponibilidade para diferentes cenários de nível de confiança e preços para os recursos *Economy*. 66

6.1 Parte do processo de gerência de recursos de nuvens computacionais. 73
6.2 Histograma da capacidade das máquinas físicas, medidas em CPU normalizada. 77
6.3 Função distribuição acumulada (FDA) empírica da capacidade requisitada por cada tarefa para as diferentes classes. 78
6.4 Função distribuição acumulada (FDA) empírica da quantidade de tarefas por job para as diferentes classes. 79
6.5 Função distribuição acumulada (FDA) empírica do tempo de execução de tarefas para as diferentes classes. 80
6.6 Função distribuição acumulada (FDA) empírica do tempo entre chegadas de tarefas para as diferentes classes. 80
6.7 Taxa de chegada de requisições ao longo do tempo para diferentes classes, em total de CPU normalizada requisitada por tarefas submetidas em janelas de 1 hora. 82
6.8 Capacidade requisitada por tarefas ao longo do tempo para diferentes classes, em total de CPU normalizada requisitada em janelas de 5 minutos. 83
6.9 Capacidade disponível no cluster (em CPU normalizada) ao longo do tempo, agregando a capacidade das as máquinas disponíveis em cada intervalo de tempo. 84
6.10 Porcentagem de requisições com SLOs de disponibilidade de VMs cumpridos (acima) e taxa de admissão de requisições (abaixo) para cada classe no cenário base. 96

6.11 Porcentagem de requisições com SLOs de disponibilidade cumpridos (à esquerda) e taxa de admissão (à direita) agregada para todas as classes ao variar a capacidade. 98
6.12 Porcentagem de requisições com SLOs de disponibilidade cumpridos (acima) e taxa de admissão (abaixo) para diferentes classes ao variar a capacidade da nuvem. 100
6.13 Utilização média da nuvem para diferentes capacidades da nuvem. 100
6.14 Eficiência da receita para diferentes capacidades da nuvem. 101
6.15 Porcentagem de requisições com SLOs de disponibilidade cumpridos (à esquerda) e taxa de admissão (à direita) agregada para todas as classes ao variar a carga. 102
6.16 Porcentagem de requisições com SLOs de disponibilidade cumpridos (acima) e taxa de admissão (abaixo) para diferentes classes ao variar a carga da nuvem. 103
6.17 Utilização média da nuvem ao variar a intensidade da carga de trabalho. . . 104
6.18 Eficiência da receita para diferentes cargas da nuvem. 105
6.19 Porcentagem de requisições com SLOs de disponibilidade cumpridos (à esquerda) e taxa de admissão (à direita) ao variar a qualidade dos SLOs das classes. 106
6.20 Porcentagem de requisições com SLOs de disponibilidade cumpridos (à esquerda) e taxa de admissão (à direita) para diferentes classes ao variar a qualidade dos SLOs das classes. 107
6.21 Utilização média da nuvem ao variar a qualidade dos SLOs das classes. . . 107
6.22 Eficiência da receita ao variar a qualidade dos SLOs das classes. 108

7.1 Taxas de admissão observadas em simulações comparadas com valores estimados pelo modelo de planejamento de capacidade, para diferentes classes e capacidades da nuvem. 125
7.2 Taxas de admissão observadas em simulações comparada com valores estimados pelo modelo de planejamento de capacidade, para diferentes classes e intensidades de carga. 126

7.3 Capacidade mínima necessária para cumprir SLOs obtidas em simulações comparadas com os valores estimados pelo modelo do planejamento de capacidade da nuvem. 128

Lista de Tabelas

5.1 Resumo dos cenários de avaliação das predições, com os valores usados para cada parâmetro. 56
5.2 Símbolos usados no modelo de receita potencial do provedor de IaaS ao oferecer diferentes classes de serviço. 62

6.1 Principais símbolos usados no modelo de controle de admissão. 85
6.2 Parâmetros de entrada para o cenário base de simulação. 95
6.3 Parâmetros de entrada para os cenários de análise de sensibilidade para a capacidade da nuvem. 97
6.4 Parâmetros de entrada para os cenários da análise de sensibilidade para a carga de entrada da nuvem. 101
6.5 Parâmetros de entrada para os cenários da análise de sensibilidade para SLOs. 105

Lista de Algoritmos

1 Simula a gerência da nuvem para uma heurística de controle de admissão. . 93
2 Estima a taxa de admissão de requisições para múltiplas classes. 120
3 Busca a menor capacidade necessária para cumprir SLOs de taxa de admissão. 121

Capítulo 1

Introdução

Neste capítulo introdutório, descreve-se o contexto de nuvens computacionais que é objeto de estudo desta tese, e a motivação do trabalho. São destacados os problemas existentes neste ambiente e a relevância de se buscar soluções para endereçá-los. Em seguida, os objetivos gerais e específicos da tese são descritos, definindo a hipótese a ser investigada e resumindo as principais contribuições do trabalho. Por fim, a organização do restante deste documento de tese é apresentada.

1.1 Motivação e relevância

A computação na nuvem é um paradigma cuja adoção vem crescendo consideravelmente nos últimos anos [47; 46]. Um dos modelos mais populares de computação na nuvem é o de IaaS (acrônimo de *Infrastructure as a Service*, em português "infraestrutura como serviço"). Neste modelo, recursos computacionais (e.g., CPU, memória e disco) são providos aos usuários, tipicamente empacotados em instâncias no formato de VMs (acrônimo de *Virtual Machines*, em português "máquinas virtuais") [70] ou *containers* [66]. Usuários da nuvem de IaaS fazem requisições de novas instâncias ao provedor, que eventualmente aloca e executa essas instâncias em suas máquinas físicas. Uma das principais propriedades da nuvem é a sua elasticidade, que permite que usuários escalem dinamicamente suas aplicações, requisitando a execução de novas instâncias (*scale out*) ou finalizando instâncias em execução (*scale down*) sob demanda.

O crescimento do mercado de nuvens computacionais está atraindo usuários de diversos

perfis, que possuem diferentes tipos de aplicação, requisitos e orçamentos [30]. Para dar suporte a esta demanda, provedores de IaaS podem oferecer múltiplas classes de serviço, com diferentes opções de qualidade e tarifação para cada classe oferecida. Desta forma, os usuários podem escolher o serviço que ofereça a combinação de qualidade e custo mais adequada às suas necessidades. Exemplos de classes de serviço oferecidas atualmente incluem as instâncias do tipo *Reserved*, *On-demand* e *Spot* da Amazon EC2, que é a nuvem pública de IaaS mais popular da atualidade [1].

Tipicamente, a gerência de recursos da nuvem é feita com base em metas de QoS (acrônimo de *Quality of Service*, em português "qualidade de serviço"). Estas metas são definidas pelos provedores como SLOs (acrônimo de *Service Level Objectives*, em português "objetivos de nível de serviço"), podendo abranger diversas métricas de QoS. Por exemplo, um provedor pode oferecer as seguintes garantias como SLOs: no mínimo $99{,}9\%$ das requisições de VM serão admitidas para execução (SLO para a taxa de admissão de VMs); uma requisição admitida pelo provedor executará no mínimo durante $99{,}99\%$ do seu tempo de vida (SLO para disponibilidade ou *uptime* de VMs).

A gerência de recursos da nuvem pode ser feita com base em SLOs explícitos ou implícitos. Os SLOs explícitos são aqueles expostos aos usuários, descritos em contratos firmados entre provedor e usuários chamados de SLA (acrônimo de *Service Level Agreement*, em português "acordo de nível de serviço"). Tipicamente, um SLA estabelece as penalidades impostas ao provedor caso os SLOs não sejam cumpridos (i.e., se houver violações de SLO). Por outro lado, os SLOs implícitos são usados apenas internamente pelo provedor para guiar suas decisões de gerência de recursos, não sendo expostos aos usuários nem prevendo penalidades ao provedor caso eles não sejam cumpridos.

Os prejuízos causados ao provedor pela violação de SLOs podem ser classificados em dois tipos: os prejuízos tangíveis, que são os facilmente mensuráveis, como as penalidades previstas no SLA e a diminuição direta na receita ao se negar requisições de serviço aos usuários; e os prejuízos intangíveis, que são os difíceis de mensurar, como o impacto negativo na reputação do provedor ao não oferecer serviços com a qualidade prometida (ou esperada), o que pode reduzir a quantidade de clientes da nuvem e consequentemente diminuir a receita do provedor. Desta forma, é importante que o provedor cumpra corretamente os SLOs estabelecidos, tanto para evitar penalidades quanto para manter uma boa reputação e atrair mais

usuários.

Um dos problemas encontrados atualmente em provedores de IaaS é a falta de SLOs explícitos definidos para os usuários. Por exemplo, os SLAs das nuvens de IaaS da Amazon EC2 [2], Google Compute Engine [8], Microsoft Azure [10] e Rackspace [12], que são alguns dos principais provedores de nuvem pública da atualidade, apresentam SLAs genéricos que oferecem garantias de disponibilidade apenas para a conectividade dos seus serviços. Outras métricas importantes para os usuários não são previstas nos SLAs, como a probabilidade de se obter novas instâncias quando requisitado ou a probabilidade de instâncias em execução falharem. Apesar desses provedores provavelmente terem SLOs implícitos para gerenciar seus serviços, a falta de SLOs explícitos dificulta o planejamento dos usuários para a execução de suas aplicações na nuvem, que dependem dessas garantias para que os requisitos de suas aplicações sejam cumpridos. Um outro problema disso é que os usuários não conseguem diferenciar claramente a QoS das diferentes classes de serviço, por não conhecerem os SLOs implícitos usados pelo provedor. Por exemplo, na nuvem da Amazon EC2 supõe-se que há uma maior probabilidade de se obter instâncias da classe *Reserved* do que da classe *On-demand*, pelo fato da primeira envolver um pagamento antecipado pela reserva de recursos.[1] Porém, como estas garantias não são explicitamente descritas no SLA, os usuários não têm garantias contratuais para o nível de serviço esperado e não conseguem quantificar a diferença da QoS entre as classes.

Definir diferentes classes de serviço e cumprir seus SLOs não é uma tarefa trivial para provedores de IaaS. Um dos motivos é a alta variação na utilização dos recursos neste tipo de ambiente. Por exemplo, uma análise da carga de um *cluster* (ou "aglomerado", em português) na nuvem privada da Google mostrou que a utilização máxima dos recursos é 30% maior do que a utilização média para aplicações em produção [64]. Também há dificuldade causada pela "elasticidade" oferecida pelos provedores de nuvem, que permite que usuários aumentem ou diminuam a quantidade de instâncias ativas quando desejado. Isto faz com que os provedores tenham que se planejar para eventuais "rajadas" (ou *bursts*, em inglês) de requisições de novas instâncias. Além disso, os *clusters* que hospedam os serviços de nuvem

[1] Esta informação consta na propaganda das instâncias *Reserved*, que destaca suas vantagens com relação à confiabilidade das suas instâncias, como descrito neste site: `http://aws.amazon.com/ec2/purchasing-options/reserved-instances/`

geralmente são compostos de máquinas que apresentam uma taxa de falhas significativa, o que fica ainda mais evidente devido à larga escala desses provedores. Por exemplo, em uma análise de falhas nas máquinas em *clusters* da Google realizada em 2013, durante 6 meses de observação, verificou-se que: metade das máquinas falharam pelo menos uma vez no período observado; 5% das máquinas falharam mais de uma vez por mês; 1% das máquinas falharam mais de uma vez por semana; e a disponibilidade média das máquinas foi de 99,84% [15].

Planejar a capacidade da nuvem para atender toda a demanda mesmo em momentos de pico, fazendo uma super-provisão de recursos e oferecendo fortes garantias de elasticidade para todos os usuários, é muito caro e geralmente inviável para o provedor. Uma solução para lidar com picos de demanda causados pela elasticidade é impor um limite de alocação para usuários da nuvem, definindo uma quantidade máxima de recursos que podem ser concedidos simultaneamente a eles [31]. Uma abordagem similar é adotada pela Amazon EC2, onde os usuários obtêm um limite padrão inicial de alocação, podendo futuramente negociar com o provedor seu aumento caso seja necessário [3]. Mesmo assim, ter uma capacidade suficiente para oferecer fortes garantias de que os usuários podem obter recursos até seus limites, simultaneamente, ainda é muito custoso e gera muito desperdício de recursos devido à variação inerente à carga da nuvem.

Contanto que as cargas de trabalho de todos os usuários não atinjam seu pico simultaneamente, os provedores podem combinar instâncias de diferentes usuários e se basear nas estatísticas da carga agregada para prover uma capacidade que seja menor do que a soma dos limites de alocação de todos os usuários, podendo assim reduzir os custos com a sua infraestrutura [59]. Apesar disso, ainda seria necessário deixar alguma capacidade excedente (i.e., com super-provisão) para garantir que requisições de recursos tenham uma baixíssima probabilidade de serem negadas [31], além de ser importante para lidar com tolerância a falhas, tendências de crescimento de mercado e flutuações periódicas da carga que são comuns para diversas aplicações da nuvem (e.g., carga de *e-commerce* durante promoções ou perto de feriados, e carga de aplicações *batch* em horários comerciais).

Uma solução proposta para aumentar a utilização de recursos da nuvem é a de reaproveitar a sua capacidade excedente de forma oportunista, oferecendo novamente as sobras de recursos através de uma classe de serviço sem garantias de QoS [56]. Geralmente, os recursos oferecidos a partir destas sobras são vendidos com preços bem mais baixos, já que não

possuem SLOs definidos. Um problema de classes oportunistas é que nem sempre elas são úteis aos usuários, principalmente para os que exigem um mínimo de garantias para atender aos requisitos de suas aplicações. Além disso, seu preço menor gera uma menor receita para os provedores.

Entender os *trade-offs* associados à qualidade de serviço e os custos de infraestrutura para satisfazer as necessidades dos usuários é essencial. Nesta tese, sugere-se que parte destes *trade-offs* pode ser representado em um *triângulo da gerência de recursos da nuvem*, similar ao triângulo de tempo–custo–desempenho proposto para a gerência de projetos [48]. Neste esquema, ao oferecer um serviço os provedores podem priorizar apenas duas de três opções disponíveis: *elasticidade*, *capacidade* e *desempenho* – sacrificando assim o terceiro atributo. A elasticidade permite aos usuários aumentarem e diminuírem a quantidade de recursos obtidos da nuvem sob demanda; o problema de se oferecer uma grande elasticidade é a alta variação na carga gerada ao provedor. A capacidade da nuvem é a quantidade total de recursos que o provedor possui; um planejamento de capacidade eficiente é necessário para acomodar a demanda e controlar os custos da infraestrutura. O desempenho dos serviços da nuvem é relacionado à QoS obtida pelos usuários, que indica como os SLOs são definidos para diferentes métricas e se essas garantias são cumpridas. O triângulo da gerência de recursos da nuvem sugerido é ilustrado na Figura 1.1.

Figura 1.1: Triângulo da gerência de recursos da nuvem. Provedores podem priorizar apenas dois dos três eixos do triângulo ao oferecer um serviço na nuvem.

Para que sejam oferecidos serviços com foco em desempenho (i.e., com SLOs de alta qualidade) e ao mesmo tempo satisfazer a demanda dos usuários com alta elasticidade, o provedor precisa super-provisionar seus recursos se planejando para os picos de demanda, o que resulta em uma baixa utilização da capacidade e aumenta os custos com a infraestrutura. Por outro lado, se o provedor priorizar uma alta utilização da capacidade para reduzir custos, ele deve escolher entre oferecer serviços com baixo desempenho (e.g., com baixa disponibilidade ao deixar requisições esperando em filas ou alta latência por sobrecarregar o sistema) ou oferecê-los com baixa elasticidade, rejeitando novas requisições de recursos em momentos de alta contenção.

Nesta tese, advoga-se que provedores de IaaS podem oferecer serviços que se adequam às diferentes necessidades dos usuários de forma lucrativa ao oferecer múltiplas classes de serviço, com diferentes níveis de serviço no espectro de elasticidade–capacidade–desempenho. Por exemplo, o provedor poderia oferecer uma classe com nível de produção com SLOs de disponibilidade de alta qualidade e grande elasticidade, o que resultaria em uma baixa utilização da sua capacidade. A capacidade excedente desta classe poderia ser reaproveitada para oferecer uma segunda classe de serviço, que garante SLOs intermediários de disponibilidade e uma menor elasticidade em troca de um menor custo (e.g., para aplicações do tipo *batch* mais flexíveis). Uma terceira classe oportunista ainda poderia ser oferecida a partir das sobras de recursos, mas com SLOs muito fracos e com uma elasticidade limitada, ou até mesmo de forma oportunista sem garantias de QoS. Exemplos de classes oportunistas incluem as instâncias do tipo *spot* da Amazon EC2 [1] e *preemptible* da Google Compute Engine [6], que podem ser interrompidas a qualquer momento e não possuem garantias de QoS. Desta forma, ao combinar múltiplas classes de serviço com diferentes SLOs e preços, provedores podem aumentar sua receita atraindo mais usuários e aumentando sua utilização.

Para que múltiplas classes de serviço possam ser oferecidas e que seus diferentes SLOs sejam cumpridos, é necessária uma gerência de recursos da nuvem eficiente, de tal forma que o custo da infraestrutura seja justificado pela receita (ou utilidade) gerada com os serviços oferecidos.

Neste contexto, pode-se resumir os problemas abordados nesta tese da seguinte forma:

- Os provedores atuais de IaaS não fornecem SLOs explícitos adequados, o que dificulta o planejamento dos usuários na execução de suas aplicações para que seus requisitos

sejam cumpridos dentro do seu orçamento.

- Não é trivial gerenciar nuvens de IaaS com múltiplas classes de serviço e cumprir seus SLOs, de tal forma que se tenha uma alta utilização de sua capacidade. As causas disso incluem a alta instabilidade na oferta e demanda de recursos neste ambiente e os *trade-offs* de qualidade e custo que existem para os serviços da nuvem.

- As soluções atuais de gerência de recursos de nuvem possuem limitações: a super-provisão de recursos é cara para o provedor e gera muito desperdício de recursos; reaproveitar a capacidade excedente oferecendo recursos oportunistas, sem garantias de QoS, gera pouca receita e nem sempre é útil aos usuários.

Para atender a estes problemas, nesta tese são propostos mecanismos que tornam a gerência de nuvens de IaaS mais eficiente. Para isto, pretende-se mostrar como provedores podem oferecer múltiplas classes de serviço e cumprir os diferentes SLOs definidos para elas, buscando obter uma alta utilização da nuvem e aumentar a receita dos provedores.

A relevância deste trabalho pode ser destacada tanto do ponto de vista do provedor de IaaS quanto dos seus usuários. Provedores podem reduzir seus custos e aumentar sua receita oferecendo múltiplas classes de serviço, mas precisam de uma gerência de recursos adequada para definir os SLOs das classes e evitar violações. Define-se *gerência adequada* como aquela que permite ao provedor obter um alto cumprimento dos SLOs estabelecidos para todas as classes e uma alta utilização dos seus recursos. Os usuários também se beneficiam por terem mais opções de serviço, com SLOs bem definidos, podendo assim adequar melhor suas necessidades às diferentes classes de serviço disponíveis e contar com as suas garantias de QoS.

1.2 Objetivos

O objetivo geral deste trabalho é investigar a seguinte hipótese: quando provedores de nuvens computacionais de IaaS realizam uma gerência de recursos adequada, oferecendo múltiplas classes de serviço e cumprindo suas metas de QoS, eles obtêm uma alta utilização de recursos e aumentam a sua receita.

Para que o objetivo geral seja atingido, os seguintes objetivos específicos foram definidos:

1. Formalizar o problema de gerência de recursos da nuvem, identificando as partes do processo de gerência que são mais importantes para solucionar o problema em questão;

2. Identificar métricas de QoS importantes para serem oferecidas aos usuários da nuvem, que permitam a eles planejar a execução de suas aplicações dentro de seus orçamentos e cumprir os seus requisitos;

3. Analisar a carga de trabalho de nuvens existentes, identificando o potencial de se oferecer novas classes de serviço;

4. Propor novas classes de serviço que se adequem bem aos casos de uso da nuvem, complementando as classes oferecidas atualmente;

5. Projetar um mecanismo de gerência de nuvens computacionais que permite ao provedor oferecer múltiplas classes de serviço, buscando maximizar a utilização da nuvem e cumprir os SLOs definidos para as diferentes classes, aumentando assim a sua receita potencial.

1.3 Contribuições

As principais contribuições desta tese são as seguintes:

1. Um método baseado em predição para planejar a capacidade e as garantias de QoS de uma nova classe da nuvem, com base em recursos excedentes advindos de classes de maior prioridade, permitindo ao provedor aumentar a utilização de seus recursos e a sua receita (Capítulo 5);

2. A caracterização de cargas de trabalho e da capacidade disponível na nuvem a partir de rastros de nuvens em produção (Capítulos 5 e 6);

3. A formalização de parte do problema da gerência de recursos de nuvens computacionais com múltiplas classes de serviço, modelado como um problema de otimização (Capítulos 6 e 7);

4. Um modelo de controle de admissão para nuvens com múltiplas classes de serviço, que busca maximizar a taxa de admissão de requisições e cumprir metas de disponibilidade de VMs (Capítulo 6);

5. Um modelo analítico para responder questões do tipo *what-if*, que estima métricas de QoS para cada classe oferecida em diferentes cenários de capacidade, demanda e de SLOs definidos para as classes (Capítulo 7);

6. Um método de planejamento de capacidade da nuvem, que permite ao provedor oferecer garantias para a taxa de admissão e a disponibilidade de VMs para as diferentes classes de serviço, buscando minimizar a quantidade de recursos necessária para cumprir essas garantias e, consequentemente, aumentar a receita do provedor (Capítulo 7).

1.4 Organização do documento

O restante deste documento está organizado da seguinte forma:

- O Capítulo 2 contextualiza o trabalho apresentando a sua fundamentação teórica, definindo os conceitos de computação na nuvem abordados e destacando o foco de estudo desta tese.

- O Capítulo 3 apresenta uma revisão bibliográfica da literatura, descrevendo o estado da arte da área, os trabalhos relacionados mais relevantes e os problemas que ainda estão em aberto.

- O Capítulo 4 descreve o modelo do sistema de nuvem computacional de IaaS considerado neste trabalho e define as questões de pesquisa abordadas nesta tese. O modelo da nuvem descrito neste capítulo serve de base para as soluções propostas nos capítulos seguintes.

- O Capítulo 5 apresenta um estudo inicial de uma nuvem computacional que considera diferentes classes de serviço. Analisa-se o potencial de se oferecer uma nova classe de serviço na nuvem com base em sua capacidade excedente, mostrando como obter garantias de QoS para esta classe e como planejar a sua capacidade de forma eficiente.

Este estudo demonstra os benefícios de se oferecer múltiplas classes na nuvem e serve de motivação para a elaboração de soluções de gerência de recursos da nuvem que considerem múltiplas classes de serviço. Os capítulos seguintes descrevem mecanismos que mostram *como* essa gerência de recursos com diferentes classes de serviço pode ser feita pelo provedor, para que esses benefícios sejam alcançados.

- O Capítulo 6 apresenta um modelo de controle de admissão de requisições de VM, que permite ao provedor da nuvem oferecer múltiplas classes de serviço e cumprir garantias de disponibilidade para cada classe. Propõe-se um modelo baseado em predição de séries temporais para decidir se novas requisições devem ser admitidas ou rejeitadas, buscando maximizar a taxa de admissões e cumprir as garantias de disponibilidade definidas para cada classe. Porém, observou-se que em cenários de alta contenção foi necessário rejeitar muitas requisições para que as garantias de disponibilidade fossem cumpridas. Desta forma, propõe-se no capítulo seguinte um mecanismo que permite ao provedor controlar a taxa de admissão de requisições para também poder oferecer garantias para esta métrica para todas as classes de serviço.

- O Capítulo 7 apresenta um modelo analítico para o planejamento de capacidade da nuvem, que estima métricas de QoS para diferentes cenários e busca encontrar a quantidade mínima de recursos necessária para cumprir SLOs de taxa de admissão de requisições e de disponibilidade de VMs para as diferentes classes oferecidas. Mostrou-se que ao combinar os mecanismos de controle de admissão e planejamento de capacidade propostos nesta tese, provedores da nuvem são capazes de cumprir garantias de disponibilidade para cada classe de serviço, ao mesmo tempo que cumprem garantias de taxa de admissão de requisições e mantêm uma alta utilização dos seus recursos.

- Por fim, o Capítulo 8 conclui a tese apresentando as considerações finais, identificando as limitações do trabalho e listando possíveis trabalhos futuros.

Capítulo 2

Fundamentação teórica

No capítulo anterior, foi apresentada a introdução do trabalho, com a descrição do contexto de nuvens computacionais, alguns problemas existentes atualmente na área e os objetivos deste trabalho para endereçá-los. Neste capítulo, é apresentada a fundamentação teórica abordada nesta tese, dando uma visão geral da área, definindo alguns termos que são usados e destacando qual é o foco do trabalho desta tese.

2.1 Computação na nuvem

A computação na nuvem é um paradigma que consiste no provimento de serviços computacionais em um mercado de *utility*, no qual usuários terceirizam serviços de computação para provedores de serviço, como geralmente é feito com outros serviços como eletricidade e água [79]. Apesar da ideia de computação como *utility* ser antiga, a implementação da ideia na prática só se concretizou há pouco mais de uma década, impulsionada principalmente pelo surgimento de tecnologias de virtualização. Com as nuvens computacionais, os usuários não precisam investir fortemente em uma infraestrutura própria para suprir necessidades relacionadas à Tecnologia da Informação (TI), nem precisam de uma equipe especializada para manter essa infraestrutura. Os provedores de serviço se beneficiam da economia de escala que esta abordagem permite, ao oferecer serviços e compartilhar sua infraestrutura com outros usuários em larga escala [77].

São vários os serviços que podem ser oferecidos pelas nuvens computacionais. É comum definir o provimento de "algo como serviço" pela nuvem com a terminologia ***aaS** (que vem

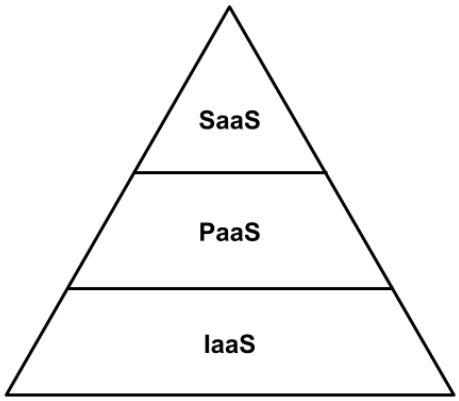

Figura 2.1: Organização em camadas dos modelos mais populares de nuvens computacionais.

do inglês **everything* as a Service*). Os três modelos de serviço de nuvem mais populares podem ser organizados em camadas (Figura 2.1) e são descritos como [73]:

- **IaaS** *(Infrastructure as a Service)*: neste modelo de "infraestrutura como serviço", os provedores de serviço oferecem recursos computacionais como ciclos de CPU, espaço de memória, armazenamento em disco e banda da rede. Geralmente, o serviço é oferecido no formato de VMs (máquinas virtuais) [70] ou *containers* [66] para que seja garantido o isolamento dos recursos físicos que são compartilhados entre diferentes usuários e evitar que a carga de uma VM afete o desempenho de outra que está alocada na mesma máquina física. Os usuários devem acessar remotamente as VMs requisitadas e gerenciar as aplicações que desejarem executar nelas. Esta é a categoria de mais baixo nível na organização em camadas dos modelos de nuvem. Exemplos de provedores de IaaS incluem Amazon EC2 [4] e Google Compute Engine [7].

- **PaaS** (*Platform as a Service*): neste modelo de "plataforma como serviço", provedores oferecem um ambiente computacional para que aplicações sejam desenvolvidas, implantadas e executadas de forma ágil. Funções como a implantação das aplicações e parte da gerência de sua execução são feitas pelo provedor de nuvem, que oferece abstrações que facilitam o desenvolvimento das mesmas e aliviam a parte operacional para os usuários. Esta é a categoria intermediária na organização em camadas, onde geralmente o serviço da camada inferior (IaaS) é usado para dar suporte a provedores de PaaS. Exemplos de provedores de PaaS incluem Microsoft Azure [9] e Google App

Engine [5].

- **SaaS** (*Software as a Service*): neste modelo de "*software* como serviço", provedores oferecem aplicações através da *Web* para usuários finais. Estas aplicações devem dar suporte a múltiplos usuários simultâneos (*multi-tenancy*) de forma customizada. Este é o serviço de mais alto nível na organização em camadas dos modelos de nuvem, onde geralmente os provedores de SaaS usam serviços de mais baixo nível (PaaS ou até mesmo diretamente o IaaS) para oferecer os seus serviços. Exemplos de provedores de SaaS incluem o Salesforce [13] e outras aplicações comuns como redes sociais e serviços de e-mail.

As nuvens computacionais também podem ser caracterizadas como **públicas** ou **privadas**. As nuvens públicas oferecem seus serviços através da Internet num modelo em que qualquer usuário pode usufruir deste serviço, pagando apenas pelos serviços que utiliza. As nuvens privadas consistem em serviços oferecidos internamente em empresas ou instituições, nas quais apenas usuários que fazem parte delas têm acesso ao serviço [38].

Provedores de nuvem podem oferecer diferentes classes de serviço aos usuários, com diferentes modelos de tarifação e garantias de QoS. Atualmente, três classes de serviço oferecidas em nuvens de IaaS públicas podem ser destacadas:

- ***Reserved***: nesta classe, o usuário paga pela reserva antecipada de uma certa quantidade de recursos, tipicamente por um longo período de tempo (e.g., 1 ou 3 anos), mas paga menos pelo uso das instâncias por ter feito esta reserva. Isto geralmente torna este modelo mais barato para aplicações que possuem uma utilização de recursos alta e constante durante longos períodos. Um exemplo desta classe de serviço é a das instâncias *Reserved* da Amazon EC2 [1].

- ***On-demand***: nesta classe, o usuário paga sob-demanda de acordo com a quantidade e tamanho das instâncias usadas em cada unidade de tempo de cobrança (geralmente feita por hora). Um exemplo de um produto que oferece esta classe de serviço é o das instâncias *On-demand* da Amazon EC2 [1].

- ***Opportunistic***: nesta classe, requisições de VM podem ser interrompidas (preemptadas) a qualquer momento para atender requisições de classes de maior prioridade,

como as classes *Reserved* e *On-demand*. Geralmente seus recursos são oferecidos de modo *best-effort*, ou seja, sem qualquer garantia de QoS. Um exemplo desta classe de serviço é o das instâncias *preemptible* da Google Compute Engine [6] e *spot* da Amazon EC2 [1]. No modelo de *spot*, o usuário dá lances com os preços que pretende pagar pelas instâncias, como em um leilão, e caso o seu lance seja maior ou igual ao preço de referência (*spot price*) definido pelo provedor ao longo do tempo, ele obtém as instâncias requisitadas pagando por elas o preço de referência. O preço médio do *spot* geralmente é menor que o preço *On-demand*, pois o primeiro possui uma maior incerteza na disponibilidade do serviço, pelo fato das instâncias poderem ser interrompidas a qualquer momento durante sua execução sem penalidades para o provedor.

O cenário de estudo deste trabalho é o de nuvens computacionais de IaaS, tanto públicas quanto privadas. A literatura relacionada a este cenário geralmente apresenta duas perspectivas diferentes: parte dela aborda a visão do provedor da nuvem, propondo soluções de como oferecer o serviço de forma adequada; a outra parte aborda a visão dos usuários da nuvem, propondo soluções para que os usuários utilizem os serviços disponíveis na nuvem da melhor forma possível, de acordo com suas necessidades. Neste trabalho, a nuvem é estudada com foco no provedor, buscando soluções para uma gerência eficiente dos recursos de nuvens de IaaS que aumente a sua receita.

2.2 Gerência de nuvens computacionais de IaaS

A gerência de nuvens computacionais de IaaS envolve várias atribuições. A gerência operacional da infraestrutura, que inclui o planejamento de compra, configuração e manutenção dos equipamentos de *hardware*, não será uma atribuição abordada neste trabalho (soluções neste sentido podem ser encontradas na literatura [15]). O foco deste trabalho é na **gerência de recursos** da nuvem, que pode ser dividida em seis categorias[1] [55]:

[1] As categorias foram adaptadas do livro citado, que originalmente define apenas 5 categorias por assumir que a capacidade da nuvem é planejada anteriormente à gerência de recursos. Para este trabalho, foi adicionada também a categoria "planejamento de capacidade", pois acreditamos que esta é uma tarefa essencial para provedores de nuvem e que também faz parte das atribuições de gerência de recursos.

1. **Planejamento de capacidade**: define qual é a capacidade adequada da nuvem para que sua demanda seja atendida de forma eficiente, evitando a super-provisão de recursos e cumprindo as garantias de QoS para as diferentes classes de serviço oferecidas aos usuários. Pode ser feito tanto o planejamento da capacidade total dos recursos da nuvem, quanto da capacidade a ser reservada especificamente para classes de serviço específicas.

2. **Controle de admissão**: estabelece critérios de admissão para requisições feitas ao provedor, de forma que sejam aceitas apenas as requisições que não gerem uma carga que possa afetar negativamente o desempenho das aplicações que já estão em execução, a ponto de causar violações de SLO. Esta tarefa se baseia no planejamento de capacidade realizado, na demanda esperada para cada classe e nos SLOs definidos para elas.

3. **Alocação de capacidade**: faz o mapeamento de instâncias requisitadas para máquinas físicas que possuem capacidade disponível, de acordo com políticas definidas pelo provedor. Alguns critérios comumente utilizados na alocação de capacidade são: evitar que algumas instâncias afetem o desempenho de outras com a sobrecarga de servidores; diminuir a quantidade de servidores ativos necessários para executar a carga de trabalho sem violações de SLO, buscando reduzir custos; ou deixar próximas as instâncias que rodam aplicações dependentes e que possuem fortes requisitos de comunicação.

4. **Balanceamento de carga**: faz o rearranjo das alocações de instâncias em máquinas físicas realizadas previamente, de com acordo políticas definidas pelo provedor. Uma técnica bastante utilizada para este propósito em nuvens computacionais é a migração de VMs. Geralmente, a tarefa de balanceamento de carga possui critérios parecidos com os adotados para a alocação de capacidade.

5. **Otimização de energia**: estabelece critérios para que o consumo de energia da infraestrutura seja minimizado, sem afetar a qualidade do serviço oferecida. Geralmente é feita em conjunto com o balanceamento de carga e alocação de capacidade.

6. **Garantias de QoS**: define as metas de qualidade de serviço que se busca obter para

cada classe de serviço oferecida da nuvem. Para isto, são definidas as métricas de QoS a serem monitoradas e as condições que devem ser satisfeitas para essas métricas. A descrição dessas condições com as metas estabelecidas para as métricas de desempenho é feita com os chamados SLOs.

Entre estas categorias de gerência de recursos de nuvens de IaaS, este trabalho será focado nos problemas relacionados ao planejamento de capacidade, controle de admissão e garantias de QoS. A seguir, é apresentada a revisão de trabalhos relacionados que endereçam principalmente problemas nestas três categorias. Trabalhos de outras categorias que não são foco da tese também são mencionados, os quais podem ser utilizados em conjunto com as propostas deste trabalho para se obter uma solução completa de gerência de recursos em nuvens de IaaS.

Capítulo 3

Revisão bibliográfica

No capítulo anterior, foi apresentada a fundamentação teórica relacionada a nuvens computacionais, definindo os conceitos gerais e destacando as áreas específicas que serão abordadas neste trabalho. Neste capítulo, será apresentada uma revisão da literatura relacionada ao trabalho proposto, descrevendo os problemas abordados, as soluções propostas e os resultados obtidos. A revisão foi dividida em dois temas: (i) gerência de recursos de nuvens computacionais; e (ii) caracterização, modelagem e predição da carga de trabalho da nuvem. Também é feita uma discussão das soluções propostas na literatura e a identificação de problemas ainda em aberto. Nesta tese são propostas novas soluções para alguns destes problemas, buscando avançar o estado da arte na área.

3.1 Gerência de recursos de nuvens computacionais

Vários trabalhos abordam a questão da alocação de recursos em nuvens computacionais, principalmente com relação ao mapeamento de VMs a máquinas físicas e ao balanceamento de carga através de migrações de VMs. Alguns trabalhos também apresentam soluções de planejamento de capacidade e controle de admissão, mas geralmente não são definidas múltiplas classes de serviço com diferentes garantias de QoS.

3.1.1 Planejamento de capacidade e controle de admissão

Com relação ao planejamento de capacidade, algumas propostas foram feitas para planejar a quantidade de servidores necessários para uma certa demanda futura da nuvem, resolvidos geralmente com base em otimização [36; 54], modelos de predição [41; 59] ou modelos analíticos [52; 51; 50; 39; 40].

Espadas et al. definem os problemas de baixa utilização (super-provisão) e saturação (sub-provisão) de recursos em nuvens computacionais, destacando a importância de soluções para lidar com estes problemas [36]. Eles também propõem um modelo de planejamento de capacidade e balanceamento de carga para aplicações SaaS que rodam em nuvens computacionais, visando uma alta utilização dos recursos sem comprometer o desempenho das aplicações. O problema do planejamento de capacidade de recursos é mapeado para o problema da mochila e um algoritmo já existente para resolver este problema é adotado. Para melhor desempenho, eles estabelecem pesos diferentes para aplicações com maior quantidade de usuários ativos e de acordo com suas necessidades de recursos. O balanceamento de carga proposto é baseado em medições de carga de cada nó do sistema, distribuindo a carga de nós sobrecarregados para nós com baixa utilização, o que pode ser feito periodicamente ou disparado por um nó saturado.

Maciel et al. apresentam soluções para gerência de curto [53] e longo [54] prazo com base em recursos disponíveis em uma infraestrutura computacional híbrida, que envolve provedores de nuvem pública, uma nuvem privada e uma grade computacional P2P colaborativa. Foram propostas diferentes heurísticas para alocação de recursos e planejamento de capacidade neste tipo de infraestrutura, algumas delas considerando predições da quantidade de recursos disponíveis pela grade computacional. A avaliação foi realizada analiticamente e com simulações. Os resultados mostraram que algumas heurísticas propostas se destacam com relação a outras, realizando a gerência da nuvem de forma eficiente, aumentando a receita dos provedores e cumprindo os requisitos de tempo de execução das aplicações.

Gmach et al. propõem um método para gerenciar a capacidade de *data-centers*, buscando evitar a super-provisão de recursos, porém cumprindo requisitos de qualidade de serviço para as aplicações [41]. O planejamento da capacidade é realizado com base em medições históricas feitas em janelas deslizantes, que identificam quais foram os períodos de sobrecarga e em quantas unidades de tempo a demanda excedida foi prolongada. Eles também apresentam

um método para predição da carga de trabalho, que identifica padrões como a periodicidade e as tendências de crescimento/diminuição da carga usando funções de periodograma e auto-correlação, agrupando as cargas de acordo com a similaridade nos padrões identificados usando o algoritmo *k-means*. Os padrões de carga estimados para cada grupo são usados na geração de carga sintética, que serve para analisar diferentes cenários de planejamento de capacidade. Os resultados mostram que o planejamento de capacidade baseado em medições de janela deslizante apresenta uma redução significativa na capacidade necessária para alocar a demanda, ao mesmo tempo que os requisitos de QoS continuam sendo cumpridos.

Meng et al. apresentam uma solução para aumentar a utilização de nuvens computacionais, buscando co-alocar máquinas virtuais que possuem cargas complementares, assumindo que os picos de carga de uma VM coincidem com os vales de alguma outra [59]. Eles propõem um modelo de SLO que diferencia VMs com relação às suas necessidades de garantias de QoS. Além disso, eles propõem um método de predição para estimar a capacidade total necessária para a provisão das VMs que foram previamente agregadas com base em sua carga, buscando cumprir o SLA estabelecido para cada VM. Para isto, é usada uma técnica que divide a parte periódica e de tendências da carga da sua parte aleatória . Em seguida, aplica-se a técnica de predição ARMA para a parte aleatória identificada. A distribuição dos erros de predição históricos também é considerada para ajustar predições futuras. A seleção de VMs compatíveis é realizada com base na medida de correlação entre suas cargas, de modo que correlações fortemente negativas indicam prováveis cargas complementares. Os resultados mostram que gerenciar a nuvem combinando VMs com cargas complementares pode reduzir significativamente a quantidade de máquinas físicas necessárias para hospedar as VMs, aumentando a utilização da nuvem e mantendo boas garantias de SLA, apesar do aumento de violações apresentadas.

Um dos problemas não abordados nessas soluções propostas para o planejamento de capacidade é a definição de diferentes classes de serviço a serem oferecidas pela nuvem, com SLOs adequados para elas. A gerência de recursos é feita com base apenas em uma classe de serviço, considerando geralmente um critério geral para todas as aplicações ao realizar o planejamento de capacidade. Isto possivelmente impede que o provedor mantenha uma alta utilização da nuvem sem que gere violações de SLO, ou faz com que as garantias de QoS oferecidas para esta única classe sejam muito fracas. Outra consequência é a falta de

opções disponíveis para os usuários atenderem às suas necessidades.

Khazaei et al. propõem um modelo analítico baseado em filas $M/G/m/m + r$ para estimar métricas de QoS em nuvens computacionais [50; 51; 52]. Este modelo permite ao provedor de nuvem analisar a QoS obtida pelo provedor para diferentes cenários de demanda e capacidade da nuvem, dando indicações para o seu planejamento de capacidade. Ghosh et al. propõem um modelo de otimização estocástico de múltiplos níveis, composto por sub-modelos de disponibilidade de máquinas e desempenho de tarefas para planejar a capacidade da nuvem [39; 40]. Ambos os modelos usam um modelo de cadeias de Markov para obter uma aproximação de métricas de desempenho, que assumem um processo de chegada de requisições seguindo uma distribuição de Poisson, um número fixo de máquinas idênticas e uma única classe de serviço. Estas premissas não são realistas para os ambientes de nuvem analisados nesta tese e podem comprometer os resultados. Além disso, resolver estes modelos analíticos para um sistema que possui uma grande quantidade de servidores ou uma grande quantidade de requisições é impraticável porque eles exigiriam em suas fórmulas o cálculo de fatoriais para grandes números, que é inviável para valores típicos de um ambiente de nuvem de larga escala como o estudado nesta tese. Vale salientar que Ghosh et al. apontaram como importante pesquisa futura o estudo de métodos preditivos de controle de admissão, assunto que foi abordado nesta tese [40].

Alguns trabalhos de controle de admissão encontrados na literatura baseiam-se no uso de predições da demanda para definir quotas de admissão. As quotas são definidas com base na estimativa da probabilidade de sobrecarga do sistema no futuro [71; 72; 29].

Unuvar et al. propõem um modelo estocástico de controle de admissão que admite um certo nível de sobrecarga na alocação de recursos (i.e., *overbooking*) da nuvem [71; 72]. O objetivo é atingir metas de qualidade de serviço relacionadas à probabilidade de se ter uma sobrecarga na utilização dos recursos, sendo capaz de lidar com as incertezas da demanda e manter uma alta utilização do sistema. Para isto, eles propõem um modelo que aproxima os dados de demanda a uma distribuição Beta para estimar a probabilidade de sobrecarga na utilização dos recursos. Os resultados mostraram que a distribuição Beta aproximou bem os dados de utilização de máquinas individuais da nuvem, principalmente quando se tem muitas tarefas alocadas na mesma, tornando o modelo útil para se estimar a probabilidade de sobrecarga da utilização de recursos nestes cenários.

Cherkasova e Phaal apresentam um método de controle de admissão preditivo para aplicações de comércio eletrônico, que ajusta um limiar de admissão com base na variação da carga observada ao longo do tempo [29]. O objetivo é prevenir que um servidor fique sobrecarregado e garantir que sessões de longa duração possam ser finalizadas com um desempenho adequado. Os resultados mostraram que o modelo foi capaz de estimar bem a carga e prover as garantias de QoS necessárias para o sistema.

Os problemas de controle de admissão abordados na literatura geralmente lidam com uma única classe de serviço, ao contrário do trabalho elaborado nesta tese. Alguns trabalhos propõem um modelo de quota (ou limiar) de admissão similar ao proposto nesta tese, mas usam modelos simples de predição para ajustar a quota que não se mostraram muito eficazes no cenário de nuvens computacionais. Nesta tese, são propostos modelos de controle de admissão que usam técnicas de predição mais elaboradas, que são capazes de capturar tendências de crescimento e flutuações sazonais na demanda da nuvem. Também foi proposto na literatura um método de controle de admissão que permite sobrecarga na alocação de recursos, mas que busca não sobrecarregar a utilização de recursos em máquinas individuais para a QoS. Este problema específico de *overbooking* não foi abordado nesta tese, mas pode ser incorporado aos modelos propostos neste trabalho em trabalhos futuros.

3.1.2 Gerência de recursos baseada em garantias de QoS

A necessidade de se fazer a gerência da nuvem baseada em garantias de QoS e prover diferentes níveis de serviço foi discutida em alguns trabalhos, como os descritos a seguir.

Cirne e Frachtenberg discutem os desafios no escalonamento de aplicações em *datacenters* que rodam sistemas web de larga escala (e.g. Google e Facebook), evidenciando as diferenças que existem com relação a sistemas de computação de alto desempenho, que foram o foco dos estudos na área até então [30]. Sistemas web nesses ambientes de nuvem são projetados para lidar com falhas, já que a larga escala as torna bastante frequentes. Essa maior probabilidade de falhas, juntamente com a grande diversidade de requisitos das aplicações, torna necessário fornecer diferentes SLOs para diferentes aplicações. Também se discute que a co-alocação de aplicações com diferentes garantias de QoS permite o aumento da utilização dos recursos da nuvem sem que haja violações de SLO. Neste artigo, os autores também apresentam um modelo hierárquico de falhas de máquinas, baseado na

distribuição de probabilidade binomial, para estimar a probabilidade de falhas de máquinas em datacenters em diversos níveis de agrupamento (e.g. *cluster, switch, rack*). Com base neste modelo de falhas, eles agrupam tarefas de acordo com seus requisitos e separam para cada grupo uma certa quantidade de máquinas extra, para tolerar as falhas que ocorrerem e assim conseguir cumprir os SLOs das aplicações. Outros desafios levantados são com relação ao planejamento de manutenções da infraestrutura, consumo de energia e na elaboração e de como cumprir SLAs probabilísticos para aplicações que toleram interrupções eventuais. Apesar deste trabalho destacar a importância de serem oferecidas diferentes classes de serviço na nuvem para aumentar sua utilização, ao combinar aplicações com diferentes níveis de serviço nas máquinas da nuvem, não foi apresentada uma solução de planejamento de capacidade para que isto seja feito.

Marshall et al. propõem o reaproveitamento da capacidade em excesso da nuvem oferecendo-a de forma oportunista, com recursos que não possuem garantias de QoS e podem sem preemptados a qualquer momento [56]. Os resultados mostraram que esta nova classe de serviço oportunista permitiu o aumento da utilização da nuvem sem que os requisitos de disponibilidade dos usuários não-oportunistas fossem afetados, o que corrobora a motivação deste trabalho. O serviço da Amazon EC2 de *Spot instances* [1] é um produto que adota uma abordagem similar. Apesar desta solução permitir o aumento da utilização, os recursos oferecidos a partir das sobras de recursos não possuem qualquer garantias de QoS definidas, o que torna os recursos pouco úteis para parte dos usuários, principalmente os que precisam de um mínimo conjunto de garantias mesmo que fracas. Esta abordagem também não gera muita receita ao provedor, devido ao baixo preço pelo os quais eles são vendidos para compensar a falta de SLOs.

Apesar de alguns destes trabalhos também fazerem a gerência de recursos da nuvem baseada em garantias de QoS, um dos problemas encontrados é que não há uma quantidade razoável de classes de serviço sendo oferecidas e algumas vezes métricas importantes para os usuários são omitidas. Geralmente, a gerência é feita considerando apenas uma única classe de serviço para todas as aplicações [59; 69]. Outros trabalhos destacam os benefícios de se oferecer mais níveis de serviço, mas apenas oferecem duas classes possíveis (com ou sem SLOs) [56], ou apenas mencionam a importância de se ter diferentes classes, mas não apresentam uma solução de como fazer isso [27; 30]. Na proposta apresentada neste trabalho,

acredita-se que oferecer mais classes de serviço permite o aumento da utilização e da receita dos provedores, além de dar mais opções de serviço para que os usuários se adequem a seus planos.

3.1.3 Alocação de recursos e balanceamento de carga

Alguns trabalhos endereçam aspectos de mais curto prazo na gerência de nuvem, como a alocação de recursos a partir do mapeamento de máquinas físicas para requisições de VMs e do balanceamento de carga para realizar ajustes de alocação para que os SLOs sejam cumpridos. As soluções de alocação de recursos e balanceamento de carga propostas foram feitas combinando VMs com cargas complementares, a partir da identificação de cargas com padrões similares com base em predições [59; 69], com heurísticas de otimização [27; 53], ou de migrações de VMs para resolver conflitos de desempenho [69; 49; 36].

Kesavan et al. propõem uma solução para o balanceamento de carga em provedores de computação na nuvem, para evitar que um aumento inesperado na carga cause sobrecarga nos recursos e degrade o desempenho das aplicações [49]. O sistema proposto é gerenciado de forma hierárquica, particionando o datacenter em níveis lógicos e tomando decisões periódicas ou reativas. Ele se baseia em migrações de máquinas virtuais, levando em consideração aspectos práticos encontrados neste tipo de solução, como o desperdício de tempo causado pelas migrações e suas prováveis falhas. A avaliação mostra que o sistema faz o balanceamento de carga de forma eficiente, reduzindo a quantidade de migrações desnecessárias e mantendo o bom desempenho do sistema, quando comparado com outras soluções.

Shen et al. propõem um método de alocação de recursos baseado em predições usando um modelo baseado na técnica de transformada de Fourier (FFT), fazendo posteriormente uma correção dos erros de predição com base em uma margem de segurança adicional para evitar violações de SLO [69]. Posteriormente, são realizadas migrações de VMs para resolver conflitos entre aplicações que possam afetar o desempenho das mesmas. Uma técnica de ajuste dinâmico de voltagem (DVS, do inglês *Dynamic Voltage Scaling*) das máquinas também é usada para reduzir os custo de energia da nuvem quando possível, de forma que os SLOs das aplicações não sejam violados. A avaliação foi feita com carga de trabalho sintética, com base em cargas de aplicações web, Hadoop MapReduce e de fluxo de eventos. Os resultados mostraram que a alocação de recursos baseada no modelo de predição proposto

apresenta uma redução significativa no consumo de energia, sem que houvesse aumento significativo nas violações de SLO.

Casalicchio et al. apresentam um modelo para a alocação de recursos em um provedor de nuvem computacional de IaaS, visando maximizar a receita do provedor e cumprir os requisitos de disponibilidade para diferentes classes de aplicações [27]. O problema é formalizado em um modelo de otimização e uma heurística baseada em *hill climbing* é proposta para solucionar o problema. O modelo considera o custo de realizar migrações, as penalidades ao não cumprir os SLAs e o custo de utilizar provedores externos (públicos) para a execução das VMs. Os resultados mostram que, comparada a uma heurística gulosa de alocação *best-fit*, a solução proposta aumentou a receita do provedor e evitou violações de SLA para os cenários avaliados com cargas geradas sinteticamente.

Estes trabalhos buscam resolver problemas de curto prazo, que podem surgir inesperadamente ao longo do tempo, e assumem que o planejamento de capacidade da nuvem já foi realizado. Como a proposta feita para este trabalho aborda problemas de longo prazo, como a definição de classes de serviço, o planejamento de capacidade e critérios para controle de admissão, os trabalhos de alocação de recursos e balanceamento de carga não substituem o trabalho proposto, mas podem ser usados em conjunto para oferecer uma solução completa para a gerência de recursos da nuvem.

3.1.4 Gerência de recursos da nuvem na perspectiva do usuário

Outros trabalhos exploraram a gerência de aplicações da nuvem na perspectiva dos usuários, buscando obter o melhor balanço entre custo e desempenho para satisfazer seus requisitos ao usar os serviços oferecidos por provedores de IaaS.

Cunha et al. apresentaram um ambiente para a especificação e execução de testes de desempenho em múltiplos provedores de nuvem [33; 32]. O objetivo foi de tornar mais fácil para o usuário o planejamento e a alocação de aplicações de SaaS, oferecendo uma ferramenta para que os usuários identifiquem qual o provedor, qual o tipo de instância e qual a quantidade de instâncias necessárias para a execução de suas aplicações, de acordo com testes de desempenho especificados e executados através da ferramenta, dos requisitos de desempenho de suas aplicações e os seus orçamentos.

Candeia et al. apresentam heurísticas para o planejamento de capacidade na execução de

aplicações que oferecem SaaS através de nuvens computacionais de IaaS [20]. As heurísticas apresentadas são dirigidas a negócio, buscando maximizar a receita gerada pela aplicação SaaS, que depende do seu desempenho obtido da nuvem e do custo com a infraestrutura usada na execução. Eles mostraram que as heurísticas foram eficientes em planejar a capacidade das aplicações de SaaS em nuvens de IaaS, aumentando a receita dos provedores de SaaS.

Sampaio e Lopes mostraram como estratégias de provisão dinâmica propostas na literatura podem usufruir de serviços oferecidos por provedores de IaaS, que permitem a mudança automática de escala (*auto-scaling*) das aplicações baseado em métricas de desempenho [68]. Para isto, são discutidos os aspectos práticos para que essas estratégias sejam adequadas aos serviços dos provedores. A avaliação é realizada com a implementação de políticas de provisão dinâmica usando os serviços de *auto-scaling* oferecidos pelos provedores de IaaS, analisando os custos de se aplicar as políticas na prática e a plausibilidade de se aplicar a teoria a partir dos produtos existentes atualmente na prática.

Estes trabalhos apresentam soluções para que os usuários de nuvens de IaaS obtenham um melhor desempenho na sua execução. No trabalho de tese proposto, pretende-se abordar a questão de gerência de recursos na perspectiva do provedor de IaaS, que deve considerar outras métricas de QoS, outros tipos de custo e um planejamento de capacidade que inclui a definição de classes de serviço a serem oferecidas para usuários.

3.2 Caracterização, modelagem e predição da carga de trabalho da nuvem

Alguns trabalhos abordaram a caracterização e modelagem da carga de trabalho da nuvem, além das configurações e disponibilidade de suas máquinas.

3.2.1 Caracterização e modelagem da carga de trabalho

Reiss et al. destacaram a heterogeneidade e dinamicidade inerentes dos recursos da nuvem e de sua carga de trabalho, mostrando a alta variação de configurações de máquinas dentro de um mesmo *cluster* e de diferentes formatos de requisições de recursos feitas pelos usuários

(e.g. diferentes proporções para os tamanhos de CPU, memória e disco requisitados), além de diversos requisitos específicos exigidos em cada requisição [64]. Esta alta variação na oferta e demanda de recursos pode ter um grande impacto na gerência de recursos oferecidos na nuvem através de diferentes classes de serviço, tornando-a uma tarefa não trivial e que exige métodos eficientes para que as garantias de QoS sejam cumpridas para as diversas classes de serviço.

Em outros trabalhos foi explorada a classificação de requisições de recursos de acordo com a quantidade de recursos requisitados e a duração do período no qual foram utilizados [60]. Também foram modelados, para alocações individuais de usuários, como o consumo de recursos é feito no tempo [81]. Além disso, foram comparadas as cargas de trabalho de grades computacionais e nuvens, destacando suas diferenças com relação às características das tarefas submetidas e à carga das máquinas [34].

Um dos problemas encontrados nestes trabalhos é que a duração dos rastros de carga de trabalho analisados foram de no máximo 1 mês. Apesar destas análises serem úteis para enteder os ambientes de nuvem, é importante que aspectos de longo prazo na carga de trabalho da nuvem também sejam analisados, para que potenciais soluções para a gerência de longo prazo da nuvem sejam identificadas.

3.2.2 Predição da carga de trabalho

Vários trabalhos abordaram o problema da predição de carga de trabalho em nuvens computacionais [62; 41; 59; 35; 42; 69], na maioria dos casos para auxiliar nas decisões de alocação de recursos e planejamento de capacidade. No contexto de grades computacionais, foram propostos modelos de predição para estimar a QoS obtida por usuários em uma grade P2P [26], além da modelagem da carga de trabalho para geração de carga sintética [22; 21].

Di et al. apresentam um modelo de predição da carga de trabalho da nuvem baseado em um modelo *Bayesiano*, no qual a utilização das máquinas da nuvem foi dividida em faixas discretas de valores, em que cada faixa representa um estado no modelo probabilístico proposto [35]. Os possíveis valores de utilização dos recursos foram divididos em 50 estados para treinar o modelo e realizar as predições. A avaliação foi realizada usando dados de 1 mês de rastros de máquinas dos *clusters* da Google, para os quais os resultados mostraram

uma boa acurácia nas predições da carga de máquinas individuais desta nuvem.

Gong et al. propõem um modelo de predição para a carga de trabalho da nuvem usando um modelo discreto de estados, com base em cadeias de Markov [42]. O modelo é treinado calculando a probabilidade de transição entre os estados. Para identificar padrões na carga, foram usadas as técnicas de *Fast Fourier Transform* (FFT) em combinação com o *Dynamic Time Warping* (DTW) [67] para identificar cargas com padrões similares. A avaliação foi feita com base em uma carga sintética, gerada a partir de modelos baseados na carga histórica do servidor web ClarkNet e da carga do site oficial da copa do mundo de 98. Os resultados mostraram que a acurácia do modelo foi superior a de técnicas de séries temporais comumente usadas.

Morais et al. propõem um método de predição de carga de trabalho da nuvem que consiste na combinação de diversas técnicas de predição de séries temporais [62; 61]. Com base em alguns critérios, o melhor preditor é escolhido de acordo com as características da carga, podendo se adaptar para diferentes ambientes ou mudanças nos padrões da carga no tempo. As predições guiam uma abordagem proativa de gerência, que é combinada também com uma abordagem reativa para resolver problemas de cargas inesperadas que não foram previstas, com o ajuste das predições baseado nos erros de predição obtidos anteriormente. A avaliação foi realizada usando rastros de carga de servidores da Hewlett-Packard (HP) com duração de 1 mês cada. Os resultados mostraram bons resultados das predições e os benefícios de se gerenciar os recursos da nuvem com base na combinação de modelos de predição, apresentando redução de custos e poucas violações de SLO.

Um dos problemas é que estas soluções de predição para a nuvem apresentam horizontes de predição que são considerados de curto prazo (e.g. algumas horas ou dias no futuro), além de estimarem a carga apenas para um momento específico no futuro, ao invés de estimarem o comportamento da carga ao longo de uma janela de predição. Os métodos propostos buscam detectar padrões na carga e a periodicidade de horários de pico ou de baixa demanda em ciclos diários. Apesar de apresentarem bons resultados, algumas atribuições de provedores da nuvem também exigem predições de longo prazo para a carga de trabalho, como o planejamento da capacidade da nuvem e do tamanho de suas classes de serviço. As predições para janelas com horizontes maiores de predição geram uma maior incerteza nas estimativas, o que deve ser considerado nos métodos de predição propostos para estes cenários, que ainda

não foram explorados no contexto de nuvens computacionais.

3.3 Conclusões do capítulo

Como foi visto nesta revisão bibliográfica, várias soluções para a gerência de recursos em nuvens computacionais já foram propostas. Porém, os trabalhos geralmente apresentam soluções para uma única classe de serviço na nuvem, tendo apenas um critério para tomar as decisões de gerência. Além disso, muitos trabalhos abordam o problema da alocação de recursos a curto prazo, assumindo que um planejamento de capacidade já foi realizado anteriormente. Outros trabalhos consideram aspectos de QoS na gerência de recursos, mas as poucas opções de classes de serviço oferecidas podem reduzir a receita dos provedores e os benefícios pros usuários. Os trabalhos de caracterização da nuvem analisam dados para curtos períodos de tempo, sendo importante também haver análises para identificar aspectos de longo prazo. Além disso, as abordagens de predição de carga propostas para a nuvem também fazem estimativas de curto prazo, que podem não ser úteis para um planejamento de capacidade de longo prazo, tendo em vista a maior incerteza nas predições para este cenário.

Tendo em vista os problemas ainda em aberto na gerência de nuvens computacionais de IaaS, neste trabalho são propostas soluções para o planejamento de capacidade e controle de admissão de provedores de nuvens computacionais de IaaS, considerando diferentes classes de serviço com SLOs bem definidos e adequados para os usuários da nuvem. Com esta revisão bibliográfica, verificamos que o problema tratado nesta tese não foi endereçado previamente na literatura, o que torna o trabalho relevante para avançar o estado da arte na área.

Capítulo 4

Modelo do sistema e perguntas de pesquisa

No capítulo anterior, foi apresentada a revisão da bibliografia relacionada ao trabalho desta tese. Neste capítulo, o modelo do sistema de gerência de nuvem computacional usado neste trabalho é descrito, com a definição da terminologia adotada. Em seguida, as perguntas de pesquisa abordadas nesta tese são apresentadas.

4.1 Modelo do provedor de nuvem de IaaS

O sistema alvo de estudos desta tese é o de nuvens computacionais de IaaS, que se aplica para provedores de nuvens públicas ou privadas. O modelo do sistema usado neste trabalho é descrito a seguir.

Neste modelo, considera-se a gerência de recursos para um **provedor** de nuvem de IaaS que possui um conjunto de **máquinas** físicas organizadas em um *cluster* ou *data-center*. Cada máquina tem uma certa **capacidade** para cada tipo de **recurso**, como CPU (em número de *cores*), memória (em *gigabytes*) e disco (em *gigabytes*). A **capacidade nominal** da nuvem é a capacidade total que o provedor possui para um recurso, agregada para todas as suas máquinas. A **capacidade disponível** da nuvem é a sua capacidade total considerando apenas as máquinas que estão disponíveis para os usuários; o seu valor pode variar ao longo do tempo por causa de falhas e manutenção das máquinas. O provedor oferece diferentes **tipos de instância** que representam a combinação das capacidades para diferentes tipos de recurso

requisitadas pelos usuários, geralmente oferecidas na forma de VMs (máquinas virtuais) ou *containers*[1]. Ou seja, diferentes tipos de instância se referem a diferentes **tamanhos de VM**. Normalmente, provedores de nuvem oferecem apenas alguns tamanhos de VM pré-definidos. Exemplos de tipos de instância disponíveis no provedor Amazon EC2 são: *m3.medium* com 1 CPU-core, 3,75GB de memória e 4GB de disco; *m3.large* com 2 CPU-cores, 7,5GB de memória e 32GB de disco; e *m3.xlarge* com 4 CPU-cores, 15GB de memória e 2 discos de 40GB cada [1].

Os usuários da nuvem fazem **requisições** de VM para o provedor ao longo do tempo, as quais são associadas a um tipo de instância. A Figura 4.1 mostra os possíveis estados de uma requisição de VM e as transições entre eles.

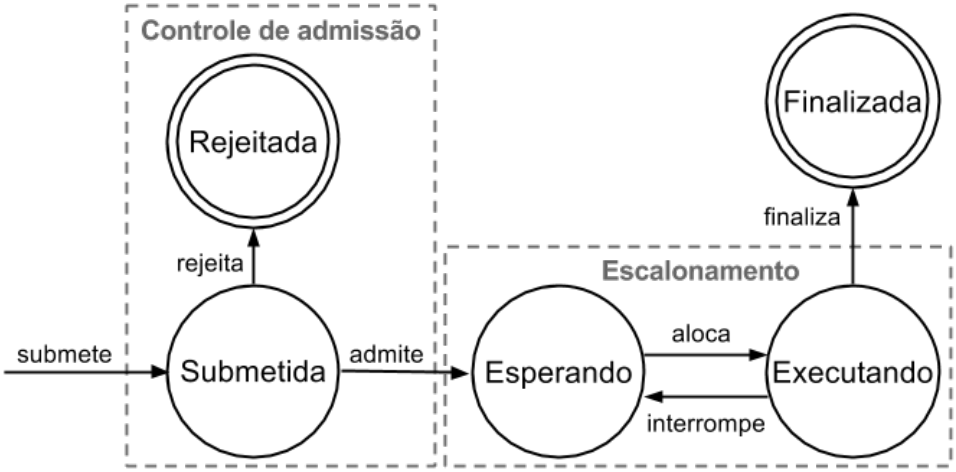

Figura 4.1: Diagrama de estado para requisições de VM no modelo de nuvem de IaaS.

Uma nova requisição de VM submetida por um usuário é rejeitada ou admitida pelo provedor na fase de controle de admissão. Requisições *rejeitadas* são descartadas pelo provedor (e.g., quando o sistema está sobrecarregado) e não são consideradas para alocação. Requisições admitidas passam para a fase de escalonamento e ficam inicialmente no estado *esperando*. Uma requisição passa do estado *esperando* para o estado *executando* quando o provedor aloca a VM requisitada em uma máquina disponível e inicia sua execução; a alocação só pode ser feita se a máquina física escolhida tiver capacidade disponível suficiente para acomodar a VM com sua capacidade requisitada, levando em consideração o tamanho das VMs já alocadas na mesma máquina. Uma requisição de VM no estado *executando* pode

[1]Os termos “instância”, “VM” e “container” são usados, neste trabalho, de forma intercambiável.

ser interrompida a qualquer momento, voltando ao estado *esperando* se a máquina que hospeda a VM falhar ou se o escalonador decidir preemptar a VM para alocar requisições de maior prioridade. Uma requisição passa para o estado *finalizada* quando o tempo de serviço associado à VM requisitada é satisfeito e ela termina sua execução.

O provedor pode oferecer múltiplas **classes de serviço** na nuvem, com diferentes preços e garantias de QoS (*Quality of Service*). Cada classe[2] geralmente possui um SLA (*Service Level Agreement*) que define suas metas de QoS, chamadas de SLOs (*Service Level Objectives*), que o provedor promete cumprir para requisições de VM daquela classe. O SLA estabelece penalidades impostas ao provedor caso haja alguma **violação de SLO** (i.e., se a meta definida não for cumprida).

Nesta tese, considerou-se as seguinte métricas de QoS para as diferentes classes de serviço:

- ***VM Availability***: ou disponibilidade de VM, é a porcentagem de tempo que uma requisição de VM passou no estado *executando*, desde o seu tempo de submissão até o tempo em que ela foi finalizada.

- ***Admission rate***: ou taxa de admissão, é a porcentagem de VMs submetidas à nuvem que foram admitidas pelo controle de admissão.

Com base nessas métricas de QoS, definiu-se os seguintes SLOs para cada classe:

- ***VM Availability SLO***: ou SLO de disponibilidade de VM, é o valor mínimo para a métrica de disponibilidade de VM aceito para requisições admitidas em uma classe, de tal forma que valores observados menores que esta meta resultam em violações deste SLO.

- ***Admissibility SLO***: ou SLO de admissibilidade, é o valor mínimo para a métrica de taxa de admissão aceito para requisições submetidas em uma classe, de tal forma que valores observados menores que esta meta resultam em violações deste SLO.

Essas duas métricas foram escolhidas para a definição de SLOs da nuvem pois elas capturam bem os requisitos de grande parte das aplicações de nuvem, como serviços com interação

[2]O termo "classe" é usado neste trabalho para denotar "classe de serviço" de forma simplificada.

de usuários e aplicações do tipo *batch* [30]. A partir do *VM Availability SLO*, é possível calcular metas de *uptime* (ou tempo total que o sistema esteve disponível e ativo) que são úteis para serviços interativos que rodam por longos períodos de tempo (e.g., servidores web), assim como metas de tempo de resposta e de vazão para aplicações *batch* e serviços não interativos que possuem prazos (*deadlines*) para finalizar sua execução. O *Admissibility SLO* é útil para aplicações que precisam da elasticidade da nuvem para definir metas de escalabilidade, como serviços que precisam escalar seus componentes para atender um pico de demanda (e.g., replicar servidores *web*) ou aplicações *batch* que paralelizam a sua execução para aumentar seu desempenho (e.g., aplicações *MapReduce* ou *bag-of-tasks*).

O provedor pode adotar um mecanismo de controle de admissão para rejeitar requisições durante picos de demanda, para não afetar o cumprimento dos SLOs de requisições já admitidas. Mecanismos baseados em **quota** definem um limite para a quantidade de recursos que podem ser requisitados em cada classe ao longo do tempo, rejeitando requisições de VM caso a quota para a sua classe seja excedida.

A Figura 4.2 ilustra o mecanismo de controle de admissão baseado em quotas. Setas direcionadas para cima representam a chegada de novas requisições de VM, enquanto as direcionadas para baixo representam requisições finalizadas. A quantidade de VMs esperando para executar impacta diretamente a métrica de *VM availability*, enquanto a quantidade de requisições rejeitadas pelo controle de admissão está relacionada à metrica de *admission rate*. Na prática, ao longo do tempo a capacidade disponível na nuvem para as diferentes classes pode mudar, assim como o provedor pode mudar as quotas associadas a essas classes para tentar cumprir os SLOs oferecidos.

4.2 Perguntas de pesquisa

Após a descrição do modelo do sistema a ser estudado, pode-se descrever com mais detalhes as perguntas de pesquisa endereçadas neste trabalho.

A principal pergunta de pesquisa que esta tese visa responder é:

- **Como provedores de IaaS podem oferecer diferentes classes de serviço, com SLOs adequados para os usuários, evitando violações de SLO e aumentando sua receita?**

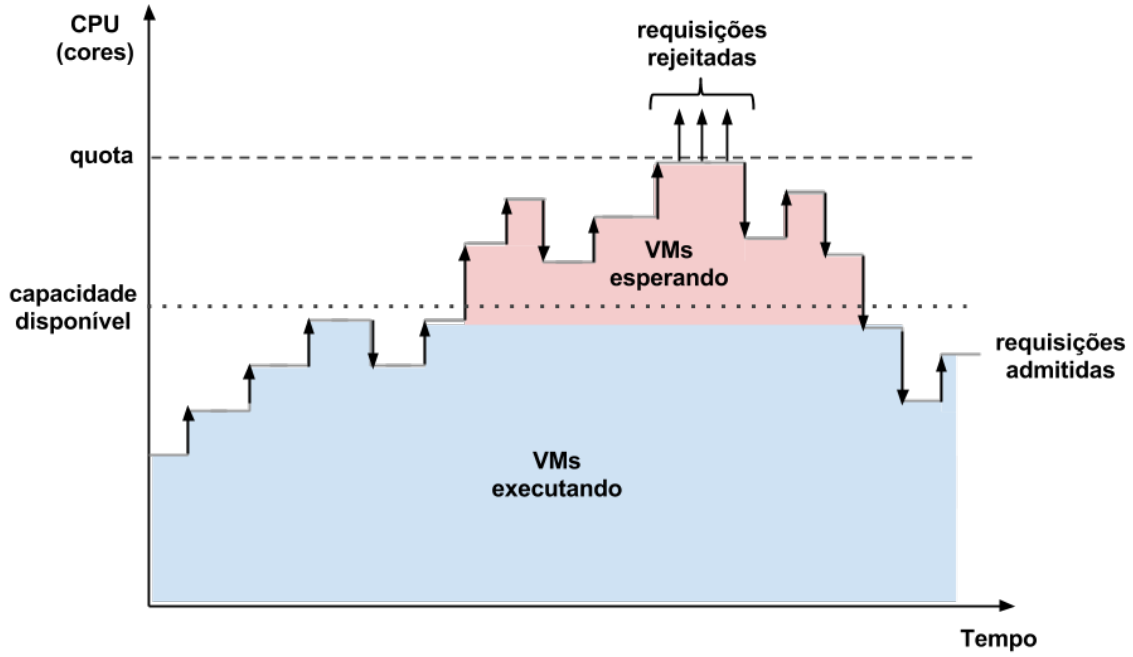

Figura 4.2: Ilustração do mecanismo de controle de admissão baseado em quotas.

Para ilustrar este problema, considere o cenário fictício de um provedor de nuvem computacional de IaaS mostrado na Figura 4.3. A linha sólida (vermelha) representa no eixo-y o percentual da capacidade da nuvem disponível ao longo do tempo, enquanto as linhas tracejadas representam a capacidade reservada para cada classe de serviço. O tempo é representado no eixo-x. Pode-se observar que nem sempre a capacidade total da nuvem (100%) está disponível, o que tipicamente é causado por falhas e manutenção de máquinas. Neste exemplo, em média 80% da capacidade total esteve disponível. Considerando um cenário mais simples em que apenas uma classe é oferecida e toda a capacidade da nuvem é reservada para ela ($reserva = 100\%$), a disponibilidade dos recursos oferecidos para esta classe seria de 80%, que é considerado um valor baixo de QoS para alguns serviços que rodam na nuvem. Para aumentar o nível de disponibilidade para 97%, que ainda pode ser insuficiente para alguns serviços, apenas 70% da capacidade total deveria ser oferecida para esta classe ($reserva = 70\%$), o que resultaria em uma baixa utilização de recursos indesejável para o provedor. Para ter uma disponibilidade próxima a 100%, onde a capacidade disponível na nuvem seria suficiente para acomodar toda a capacidade reservada para a classe durante todo o tempo, apenas metade da capacidade da nuvem deveria ser oferecida ($reserva = 50\%$),

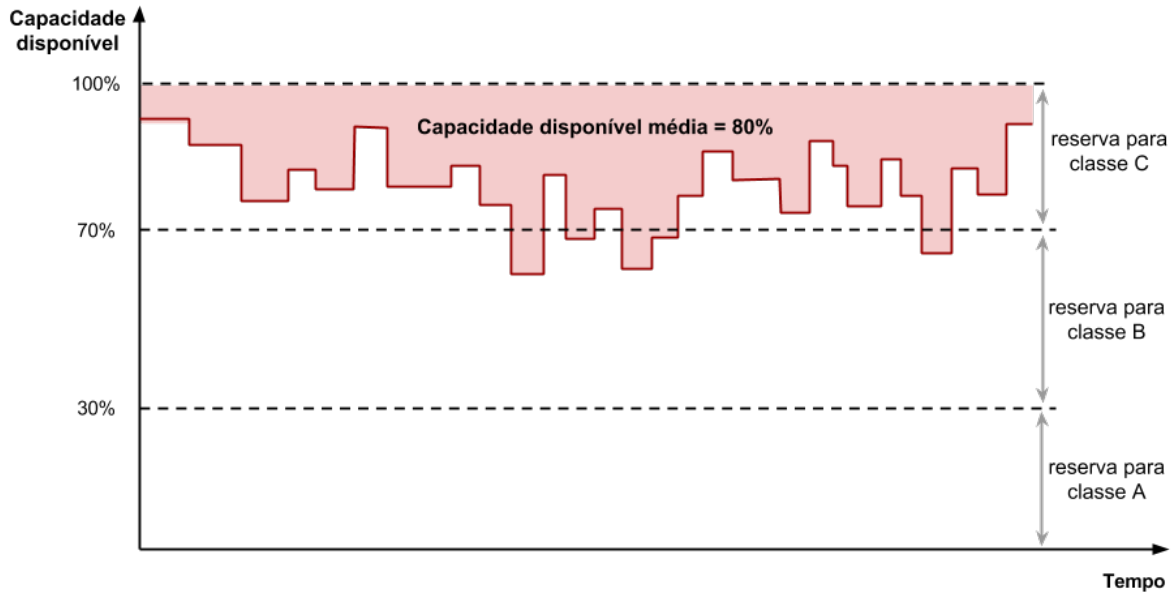

Figura 4.3: Ilustração do problema da gerência de recursos para uma nuvem computacional, mostrando a capacidade da nuvem disponível no tempo e a reserva planejada para diferentes classes de serviço.

enquanto o restante dos recursos seriam usados apenas para tolerância a falhas.

Uma forma de aumentar a utilização da nuvem sem que haja violações de SLO é definindo diferentes classes de serviço. Neste caso, o provedor pode oferecer diferentes níveis de QoS e reservar uma certa quantidade de recursos para cada classe, definindo uma ordem de prioridade para a alocação de VMs de cada classe. No exemplo da Figura 4.3, o provedor oferece três classes de serviço: a *classe A* tem uma reserva de 30% da capacidade total e possui a maior prioridade; a *classe B* tem uma reserva de 40% da capacidade total e possui uma prioridade intermediária; e a *classe C* tem uma reserva de 30% da capacidade total e possui a menor prioridade. Os recursos da nuvem são alocados de acordo com a prioridade de cada classe, fazendo com que a indisponibilidade de recursos afete mais as classes de prioridades mais baixas. Neste exemplo, com a variação na disponibilidade total de recursos da nuvem, a disponibilidade de recursos obtida para as classes A, B e C seria de 100%, 95% e 40%, respectivamente. Note que apesar da disponibilidade total da nuvem ser de 80%, a ordem de prioridade e a reserva definida para as diferentes classes fizeram com que elas obtivessem diferentes qualidades de serviço.

Com base neste exemplo, pode-se destacar alguns benefícios de se oferecer diferentes classes de serviço:

- O provedor da nuvem pode obter uma alta utilização ao mesmo tempo que oferece parte dos recursos através de serviços com fortes garantias de QoS, ao definir prioridades e reservas de recurso adequadas para as classes. Isto permite que a capacidade da nuvem seja preenchida gradativamente, da classe que exige garantias mais fortes até as menos exigentes, fazendo com que as a variação da disponibilidade de máquinas da nuvem afete menos as classes de maior prioridade.

- O provedor pode aumentar sua receita mantendo uma alta utilização de recursos da nuvem e definindo preços adequados para cada classe, podendo cobrar mais pelos recursos quanto mais fortes forem as garantias de QoS das classes às quais eles pertencem.

- Os usuários terão mais opções de serviço na nuvem, podendo escolher a classe mais adequada para atender os requisitos de suas aplicações e o seu orçamento.

Para investigar se estes benefícios serão de fato obtidos, foram abordados problemas específicos de gerência de recursos da nuvem relacionados às seguintes categorias: definição de classes de serviço a serem oferecidas pela nuvem e suas garantias de QoS; planejamento de capacidade da nuvem; e controle de admissão de requisições feitas por usuários, buscando cumprir os SLOs das diferentes classes oferecidas. A seguir são apresentadas as perguntas de pesquisa para cada categoria endereçada nesta tese.

4.2.1 Definição das classes de serviço

Um dos problemas abordados é a definição de classes de serviço que possuam garantias de QoS adequadas para os usuários. Atualmente, as classes de serviço oferecidas pelos provedores de IaaS não tornam explícitos alguns SLOs importantes para usuários. Além disso, há poucas opções de serviço disponíveis, o que dificulta o planejamento dos usuários tendo em vista a diversidade nos requisitos de suas aplicações e nos seus orçamentos.

Desta forma, uma pergunta de pesquisa endereçada neste trabalho é:

- **Como definir as diferentes classes de serviço a serem oferecidas na nuvem, com métricas e níveis de serviço adequados para os usuários?**

Isto envolve identificar métricas de QoS relevantes para os usuários, além de analisar as consequências de se oferecer níveis diferentes de SLO (mais fortes ou mais fracos) para as diferentes classes. Esta pergunta é endereçada inicialmente no Capítulo 5, com uma análise de uma nova classe de serviço introduzida numa nuvem existente com garantias de disponibilidade para seus recursos. Nos Capítulos 6 e 7 ela é aprofundada ao se propor e analisar modelos de controle de admissão e planejamento de capacidade para múltiplas classes de serviço, com diferentes garantias de admissibilidade e de disponibilidade de VMs para elas.

4.2.2 Planejamento de capacidade

Uma outro problema abordado é o do planejamento de capacidade do provedor, que consiste em decidir a quantidade total de recursos necessários para oferecer as diferentes classes de serviços e cumprir os SLOs definidos para cada uma.

A pergunta relacionada ao planejamento de capacidade endereçada nesta tese é:

- **Como planejar a capacidade da nuvem, de tal forma a cumprir os SLOs oferecidos para as diferentes classes de serviço com o mínimo possível de recursos?**

Isto envolve a elaboração de um modelo que estime a demanda de recursos para cada classe, as falhas das máquinas da nuvem, além das métricas de QoS a serem obtidas para diferentes capacidades e cenários da nuvem. Esta pergunta é tratada no Capítulo 7, que apresenta um modelo analítico para o planejamento de capacidade de nuvens de IaaS com múltiplas classes de serviço.

4.2.3 Controle de admissão

Com base no planejamento de capacidade realizado para a nuvem, o provedor precisa definir métodos para o controle de admissão de requisições de VM. Com isto, o provedor deve buscar cumprir os SLOs das requisições admitidas e, ao mesmo tempo, maximizar a quantidade de requisições admitidas para obter uma alta utilização e receita.

Com relação ao controle de admissão, a seguinte questão de pesquisa é abordada:

- **Como controlar a admissão de novas requisições de VM para cada classe de serviço, buscando maximizar a quantidade de admissões e cumprir os SLOs das requisições admitidas?**

Esta pergunta é abordada no Capítulo 6, onde é proposto um modelo de controle de admissão para provedores de IaaS com diferentes classes de serviço.

Todas essas questões são endereçadas nesta tese buscando a eficiência na gerência não só de classes individuais, mas a eficiência global do provedor considerando a coexistência das várias classes oferecidas na nuvem, visando aumentar sua utilização e receita.

Capítulo 5

Planejando uma nova classe para a nuvem com base em sobras de recursos

No capítulo anterior, o modelo do sistema de nuvem adotado nesta tese foi apresentado, com o levantamento de questões de pesquisa abordadas. Neste capítulo[1], mostra-se como uma nova classe de serviço pode ser introduzida em uma nuvem computacional existente, planejando a quantidade de recursos a serem oferecidos através desta classe e as suas garantias de disponibilidade.

5.1 Introdução

Embora se venda a ideia de que nuvens de IaaS oferecem uma "elasticidade infinita" de recursos aos usuários, claramente isto não é o caso, já que os recursos físicos do provedor são finitos. Na prática, os provedores impõem um limite de alocação para cada usuário, que define a quantidade máxima de recursos que pode ser concedida ao usuário simultaneamente na nuvem. Geralmente, os usuários obtêm um limite padrão no início, podendo negociar com o provedor seu aumento no futuro, caso seja necessário.

Planejar a capacidade do provedor da nuvem para que os usuários tenham fortes garantias de obter recursos até seus limites, todos de uma só vez, é muito caro e gera muito desperdício, já que para a maioria das aplicações a média da utilização de recursos da nuvem é bem

[1]Este capítulo é baseado no artigo publicado no *ACM Symposium on Cloud Computing (SoCC)* [23], feito em colaboração com Walfredo Cirne e John Wilkes da Google durante estágio realizado na empresa.

menor do que o máximo requisitado e a variação na quantidade de requisições recebidas é alta devido aos chamados *bursts* de demanda [64]. Alguns usuários podem querer pagar caro para ter recursos dedicados e sempre disponíveis, mas a maioria pode aceitar garantias menos fortes em troca de menores preços.

Neste contexto, este estudo aborda o problema da super-provisão de recursos em nuvens computacionais, que tipicamente é resultado de um planejamento de capacidade da nuvem visando atender à demanda de pico, resultando em sobras de recursos em grande parte do tempo. Discute-se como o provedor pode introduzir uma nova classe de serviço na nuvem, reaproveitando parte da capacidade excedente e oferecendo novamente estes recursos com bons SLOs de disponibilidade, mantendo uma alta utilização da nuvem e permitindo um aumento na receita do provedor.

Uma abordagem proposta anteriormente para aumentar a utilização da nuvem foi de oferecer os recursos em excesso (i.e., que não estão sendo usados em um certo momento) de forma oportunista, ou seja, sem qualquer garantia de QoS para eles [56]. Infelizmente, esta ausência de SLOs restringe as aplicações que podem se beneficiar desses recursos reaproveitados. Por isto, recursos oportunistas geralmente são vendidos a preços bem menores que o de outras classes. Um exemplo de produto que oferece recursos oportunistas desta forma é o modelo de *Spot instances* da Amazon EC2 [1], que tem um preço médio aproximadamente 70% menor do que o preço das instâncias não oportunistas do modelo *on-demand* [23].

Recursos oportunistas podem ser aceitáveis em alguns casos, por exemplo para algumas aplicações do tipo *batch* que podem fazer *checkpoints* periódicos e retomar sua execução a partir deles, ou que podem ser reiniciadas e atrasar suas execuções sem maiores problemas. Porém, para outras aplicações é necessário um mínimo de garantia para que se tenha noção de um prazo de finalização de sua execução ou do seu *uptime*. Além disso, uma parte significativa das aplicações que executam atualmente em nuvens são serviços que ficam ativos continuamente por semanas ou até meses [64; 30]. Geralmente, estes serviços dão suporte a aplicações *web* interativas e precisam de garantias de QoS que cubram longos períodos de tempo. Sem SLOs de longo prazo definidos para a disponibilidade, estes usuários não podem ter certeza de que terão recursos disponíveis quando necessário. Por exemplo, uma aplicação de *e-commerce* pode precisar garantir que ela poderá obter recursos necessários para atender a um aumento futuro esperado na carga após uma campanha publicitária.

Com isto, um dos objetivos deste estudo é melhorar as garantias de disponibilidade para sobras de recursos que são reaproveitadas, de forma que elas se tornem úteis para aplicações típicas da nuvem que executam serviços por longos períodos de tempo – especialmente aquelas que podem tolerar disponibilidades um pouco menores que as metas comuns em troca de um menor custo – e que, com isso, o provedor possa obter uma maior receita para estes recursos, ao invés de simplesmente ofertá-los de forma oportunista.

Neste estudo, mostra-se como um provedor de nuvem pode oferecer fortes garantias de disponibilidade e de longo prazo para parte da capacidade em excesso reaproveitada, através de uma nova classe de serviço chamada *Economy On-demand*. Embora os recursos dessa classe possam ser preemptados se necessário, por serem reaproveitados de sobras pertencentes a outras classes de serviço, o provedor deve planejar esta classe de tal forma que a chance disso acontecer seja pequena o suficiente para torná-la útil mesmo por períodos de tempo de vários meses. Para isto, são feitas predições da quantidade de sobras de recursos da nuvem que muito provavelmente não serão usados durante um certo período do futuro, baseadas em medições históricas dos padrões de uso da nuvem. Além disso, intervalos de confiança são calculados para lidar com a incerteza das predições, fazendo estimativas mais conservadoras quando necessário. O uso de diferentes níveis de confiança permite explorar os *trade-offs* entre o risco de se ter violações de SLO e de se oferecer uma maior quantidade de recursos através desta nova classe.

O método proposto para planejar a nova classe foi avaliado usando rastros de carga de trabalho de 6 *clusters* em produção da Google, coletados para um período de 45 meses. Os resultados mostram que é possível reaproveitar até 17% da capacidade total da nuvem através da nova classe de serviço com garantia de disponibilidade de 98,9% para períodos de 6 meses. Além disso, mostra-se que o provedor de IaaS pode aumentar sua receita em até 60% ao reaproveitar os recursos em excesso desta forma, em comparação a oferecer todos os recursos não usados de forma oportunista, sem SLOs para disponibilidade e consequentemente mais baratos e menos úteis aos usuários.

As principais contribuições deste estudo são as seguintes:

- Análise de cargas de trabalho reais de longa duração para identificar o problema da baixa utilização da nuvem e buscar ideias de soluções para aumentá-la de forma eficiente e lucrativa;

- Definição de uma nova classe econômica de serviço, que se baseia no reaproveitamento de recursos não usados da nuvem, com fortes garantias de disponibilidade definidas para ela;
- Elaboração de um modelo de predição para as sobras de recurso da nuvem, para se estimar a fração de recursos a serem oferecidos pela nova classe de serviço, de tal forma a alcançar um bom equilíbrio entre a quantidade de recursos oferecidos e a qualidade de serviço desta classe através dos SLOs de disponibilidade definidos para ela;
- Instanciação do método proposto usando técnicas de predição bem conhecidas e avaliação do método através de simulações alimentadas por cargas de trabalho de 6 *clusters* de larga escala, que abrangem um longo período de tempo (45 meses cada);
- Demonstração de que a receita do provedor pode ser aumentada com novas classes de serviço, através da modelagem dos resultados obtidos ao se ofertar a nova classe de serviço de acordo com o método proposto.

O restante deste capítulo é organizado da seguinte forma. A Seção 5.2 descreve o problema abordado. A Seção 5.3 analisa a carga de trabalho de uma nuvem computacional em produção e mostra o potencial de se reaproveitar suas sobras de recursos, especialmente se oferecidos novamente com garantias de QoS de longo prazo. A Seção 5.4 descreve o modelo de predição proposto neste estudo, que é usado para estimar a fração de recursos da nuvem que podem ser oferecidos a partir da nova classe com garantias de disponibilidade para eles. A Seção 5.5 avalia o método proposto usando dados reais de 6 *clusters* de larga escala. Por fim, a Seção 5.6 apresenta as conclusões deste estudo.

5.2 Descrição do problema

Neste estudo, pretende-se mostrar como um provedor de IaaS pode oferecer uma nova classe de serviço com garantias de QoS, com base em recursos não usados da nuvem. O modelo do sistema de nuvem de IaaS descrito no Capítulo 4 foi usado como base.

Considerou-se que usuários possuem um **limite de alocação** pré-definido para cada recurso, que é a quantidade máxima deste recurso que pode ser alocada simultaneamente.

Geralmente se aplica um limite padrão para todos os usuários, podendo ser feita uma negociação com o provedor para aumentá-lo, o que normalmente é feito de forma manual através de formulários [31]. Uma requisição de VM é considerada válida se, com a sua alocação, a soma dos recursos alocados para o usuário for menor ou igual ao seu limite. O provedor admite requisições de VM se elas forem válidas e se houver capacidade suficiente na nuvem para a sua alocação no momento requisitado.

Para aumentar a chance de atender requisições, o provedor pode fazer **reservas** de recursos para cada classe de serviço oferecida para tentar garantir que, com alta probabilidade, haverão recursos suficientes para atender usuários de cada classe no futuro e cumprir as garantias de disponibilidade oferecidas.

Em uma gerência de recursos baseada em reservas (ou quotas) para as classes de serviço, a capacidade da nuvem pode não ser totalmente utilizada todo o tempo. As sobras resultantes da capacidade excedente podem vir de diferentes fontes, como mostra a Figura 5.1.

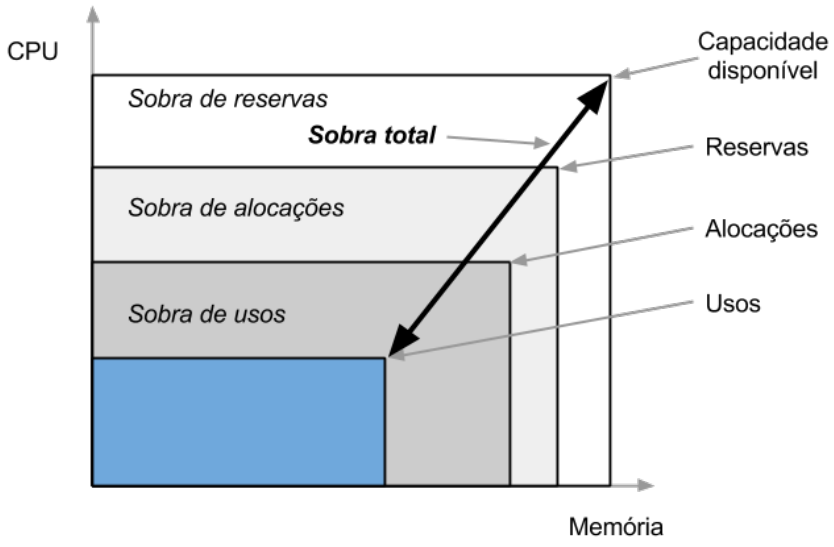

Figura 5.1: Como a nuvem é gerenciada e os diferentes tipos de sobra de recursos.

Os diferentes tipos de sobra são descritos como:

- **Sobra de reservas**: é a sobra de recursos resultante da capacidade da nuvem que ainda não foi reservada para uma classe de serviço – i.e., a capacidade total da nuvem menos a soma das reservas feitas pelo provedor. Por exemplo, são recursos extra que podem ser usados para aumentar a quantidade de recursos reservados para as classes de serviço no futuro ou mantidas para tolerar falhas das máquinas.

- **Sobra de alocações**: é a sobra de recursos resultante da capacidade que já foi reservada para alguma classe ou usuário, mas que ainda não foi alocada para VMs – i.e., a soma das reservas menos a soma dos tamanhos das VM em execução alocadas pelo provedor. Por exemplo, o provedor reserva recursos para uma classe de serviço para cumprir seus SLOs, mas parte desses recursos podem não ser requisitados e de fato atribuídos a VMs em momentos de baixa demanda. Considera-se que uma reserva temporária é criada para alocações que não fazem parte de reservas previamente estabelecidas. Ou seja, o total de alocações nunca excede o total de reservas feitas.

- **Sobra de uso**: é a sobra resultante de recursos que são alocados para uma VM mas não são totalmente usados – i.e., a soma das alocações menos a soma do uso dos recursos. Por exemplo, a sobra resultante da baixa utilização de recursos por uma VM ociosa que não está consumindo toda a capacidade que foi requisitada e alocada para ela.

Também medimos a **sobra total** dos recursos não usados na nuvem. Ela é calculada como a soma dos três tipos de sobra descritos acima, que é igual à capacidade disponível da nuvem menos a soma do uso dos recursos para todas as VMs em execução. O termo **sobra de recursos** é usado neste estudo para denotar a sobra total de recursos, a menos que seja especificado algum tipo particular de sobra.

As sobras de recursos podem ser reaproveitadas pelo provedor, por exemplo oferecendo novamente estes recursos excedentes através de classes de serviço com garantias mais fracas de QoS. O reaproveitamento das *sobras de reservas* e das *sobras de alocações* é facilmente aplicável, pois esta capacidade excedente não está alocada para nenhuma instância. As *sobras de uso* são mais difíceis de serem reaproveitadas, pois estas sobras estão dentro de VMs em execução, exigindo ferramentas que ofereçam suporte específico para este tipo de reaproveitamento [75].

Uma solução adotada para aumentar a utilização da nuvem é oferecer a capacidade excedente sem garantias de QoS, como na classe de serviço *Opportunistic* descrita anteriormente [56]. Neste estudo, propõe-se uma nova classe de serviço chamada *Economy on-demand* (ou apenas *Economy*, por simplicidade), na qual parte da capacidade excedente da nuvem pode ser oferecida novamente com fortes garantias de QoS durante longos períodos de tempo.

Esta nova classe se posiciona abaixo da classe *On-demand*, oferecendo um *admissibility SLO* comparável ao da classe *On-demand*, porém tendo um disponibilidade a longo prazo ligeiramente menor (e.g., 98.9% ao invés de 99.95%). A classe *Economy* pode suprir as necessidades das aplicações (incluindo serviços que executam durante longos períodos) que não podem usar recursos oportunistas devido à sua incerteza, mas que não precisam de garantias tão fortes quanto as oferecidas pelas classes *Reserved* ou *On-demand*. A vantagem é que a classe *Economy* pode ser oferecida com preços bem menores aos usuários, mas com boas garantias de QoS e com benefícios notáveis para a receita do provedor.

A análise da carga da nuvem apresentada na próxima seção mostra que a quantidade de recursos excedentes é significativa em *clusters* de nuvens em produção. Em seguida, é proposto um modelo de predição que estima, para longos períodos de tempo, a capacidade da nuvem não usada e que pode ser oferecida através da classe *Economy*, além do planejamento das garantias de disponibilidade que podem ser definidas para ela.

5.3 Análise da carga de trabalho da nuvem

Nesta seção são apresentadas análises de como os limites de alocação dos usuários são utilizados e como as sobras de recurso variam ao longo do tempo para rastros de 6 *clusters* da nuvem interna da Google. Os dados são de cargas de trabalho observadas no período de dezembro de 2012 a novembro de 2013, medidos em intervalos de 5 minutos.

5.3.1 Análise da utilização dos limites de alocação

Primeiramente, analisou-se o quanto dos limites de alocação dos usuários são realmente alocados, para cada intervalo de 5 minutos observado nos rastros. O termo **utilização dos limites de alocação** é definido como a soma de todas as alocações requisitadas dividida pelos limites de alocação. Esta utilização é calculada tanto para alocações e limites de usuários individuais, como também agregada para todo o *cluster*. Para o agrupamento por usuário, a utilização é calculada somando todas as alocações feitas pelo usuário dividido pelo seu limite de alocação. Para o agrupamento por *cluster*, ela é calculada somando todas as alocações feitas no *cluster* por todos os usuários e dividido pela soma dos limites de alocação de todos os usuários para o *cluster*.

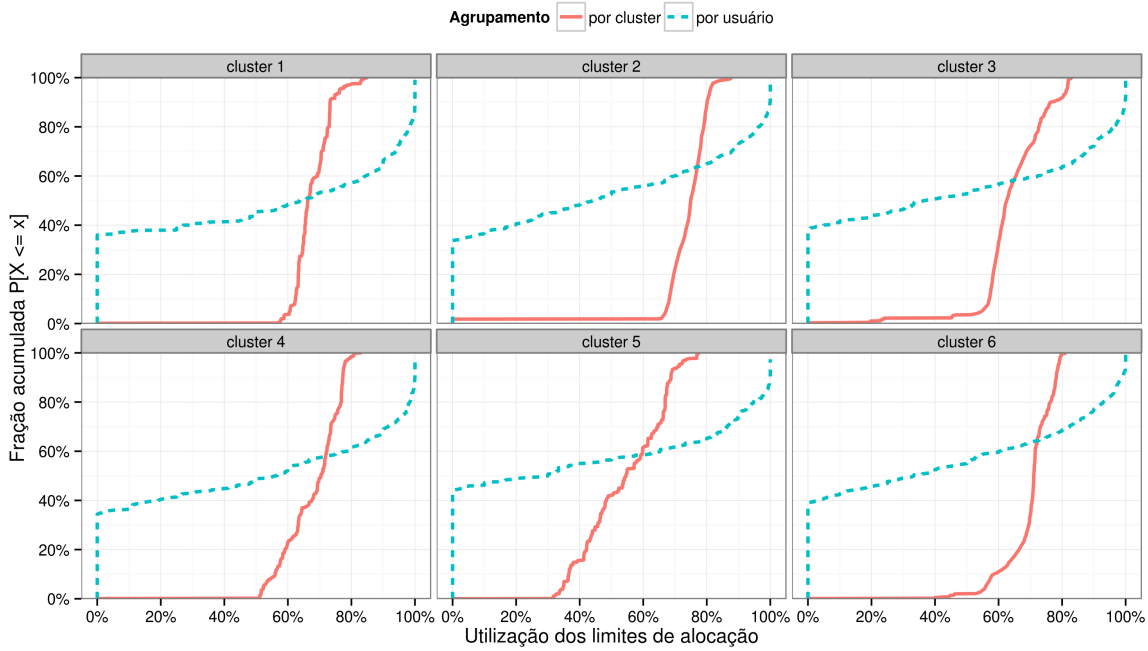

Figura 5.2: Função de Distribuição Acumulada (FDA) empírica para a utilização dos limites de alocação de CPU, agrupados para usuários individuais e para todo o cluster, medidos a cada 5 minutos.

A Figura 5.2 mostra a Função de Distribuição Acumulada (FDA, ou *CDF* em inglês) empírica para a utilização dos limites de alocação de CPU para os *clusters* analisados. A linha tracejada (azul) mostra a FDA da utilização dos limites de alocação de CPU por usuário, para a qual a mediana varia de 30% a 63% para os 6 *clusters*, mas também apresenta alguns valores extremos: em 38% das medições a utilização dos limites foi menor que 1%, enquanto em 15% das medições mais de 99% dos limites foram de fato alocados. Aparentemente, usuários não utilizam completamente os seus limites durante grande parte do tempo, sugerindo que os limites de alocação não são definidos corretamente para eles, ou que há uma grande variação na sua utilização ao longo do tempo, em que altos valores de utilização são atingidos esporadicamente. Este comportamento também pode ser causado pela combinação desses dois fatores.

Para medições agregadas no nível do *cluster* (linha cheia vermelha), a mediana da utilização dos limites de alocação de CPU varia de 55% a 75%, mas percebem-se menos valores

extremos: a utilização é menor que 1% em menos de 1% das observações, além de ser igualmente raro ter valores de utilização maiores que 81%. Com isto, nota-se que agregar as alocações e observá-las para todo o *cluster* diminui a variabilidade da utilização dos limites de alocação, reduzindo o intervalo de valores observados e valores extremos. Isto significa que tomar decisões globais e planejar limites para grupos de usuários é mais eficiente e controlável do que fazer reservas de recursos de acordo com os limites individuais dos usuários. Como é raro a soma das alocações atingir o limite de alocações para todos os usuários simultaneamente (i.e., os picos de demanda não coincidem para todos os usuários), o provedor não precisaria reservar uma quantidade de recursos igual à soma dos limites de alocação de todos usuários para oferecer fortes garantias de QoS para métricas como disponibilidade e taxa de admissão.

5.3.2 Análise da capacidade excedente

Nesta análise, destacamos o tamanho da capacidade não usada da nuvem, causada pela superprovisão de recursos quando um modelo de reserva para classes e limites para usuários é usado. A identificação deste problema é a base das soluções propostas neste trabalho, para que a capacidade da nuvem seja aproveitada de forma mais eficiente.

A Figura 5.3 mostra as sobras de CPU dividas em diferentes tipos de sobra, como descrito na Seção 5.2. Essa quantidade de sobras é normalizada pela capacidade total para este recurso e é apresentada como o percentual dessa capacidade. Em média, as sobras de recursos totalizam 57% da capacidade dos *clusters*, que é um valor bastante significativo. É interessante notar que as sobras das reservas e das alocações possuem uma variação maior ao longo do tempo do que as sobras de uso dos recursos, que são mais bem comportadas.

A Figura 5.4 mostra a FDA empírica do total de sobras para CPU, memória e disco no intervalo de tempo observado. Em 99% das medições, mais de 45% da capacidade de CPU, 43% de memória e 89% de disco não foram usadas. Esta grande quantidade de sobras de recursos é a principal motivação para que estes recursos sejam reaproveitados de forma eficiente.

Neste estudo, foi levantada a hipótese de que oferecer parte dos recursos não usados com boas garantias de QoS através de uma nova classe (*Economy*), ao invés de oferecer toda a sobra de recursos de forma oportunista sem garantia alguma, traz benefícios tanto para os

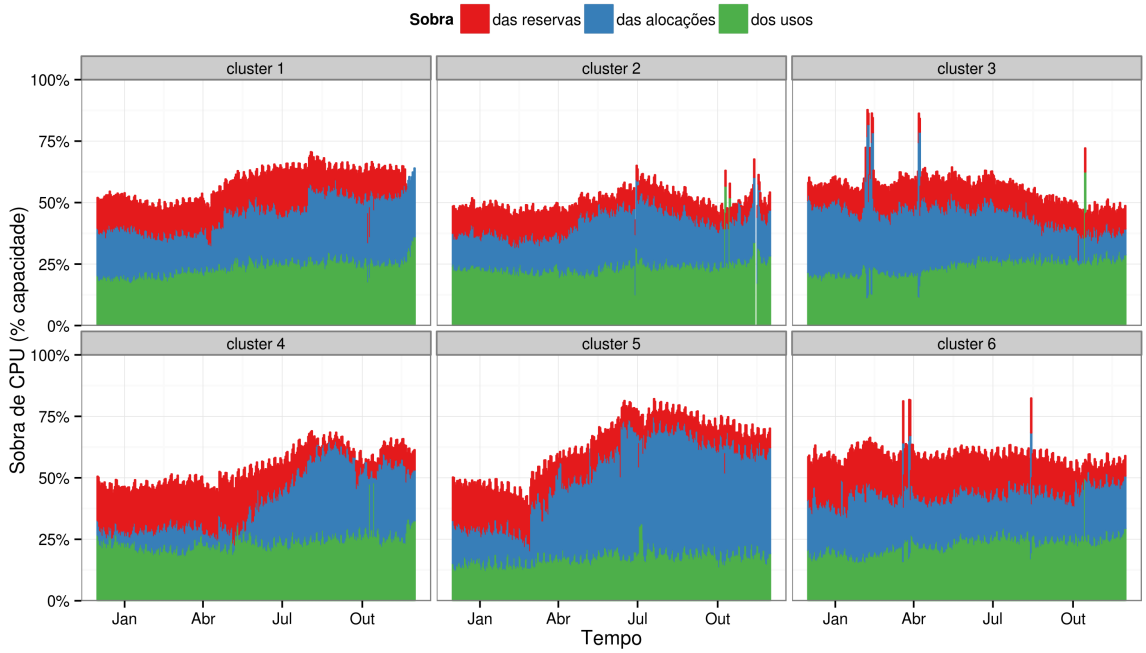

Figura 5.3: Sobras de CPU no tempo, normalizadas pela capacidade total, medidas em intervalos de 5 minutos.

usuários quanto para os provedores da nuvem.

5.4 Modelo de predição

Quanto mais confiáveis as predições para a fração de recursos excedentes e quanto mais provável for deles continuarem disponíveis no futuro, melhor serão os SLOs oferecidos para a classe *Economy*. Boas predições podem resultar em uma maior utilização de recursos, menos violações de SLO e, consequentemente, uma maior receita para o provedor. Para isto, propõe-se um modelo de predição para as sobras de recurso composto de três partes: (i) previsão de séries temporais; (ii) cálculo de intervalos de confiança para predições; e (iii) uso de ciclos de predição. A seguir cada uma dessas partes é descrita.

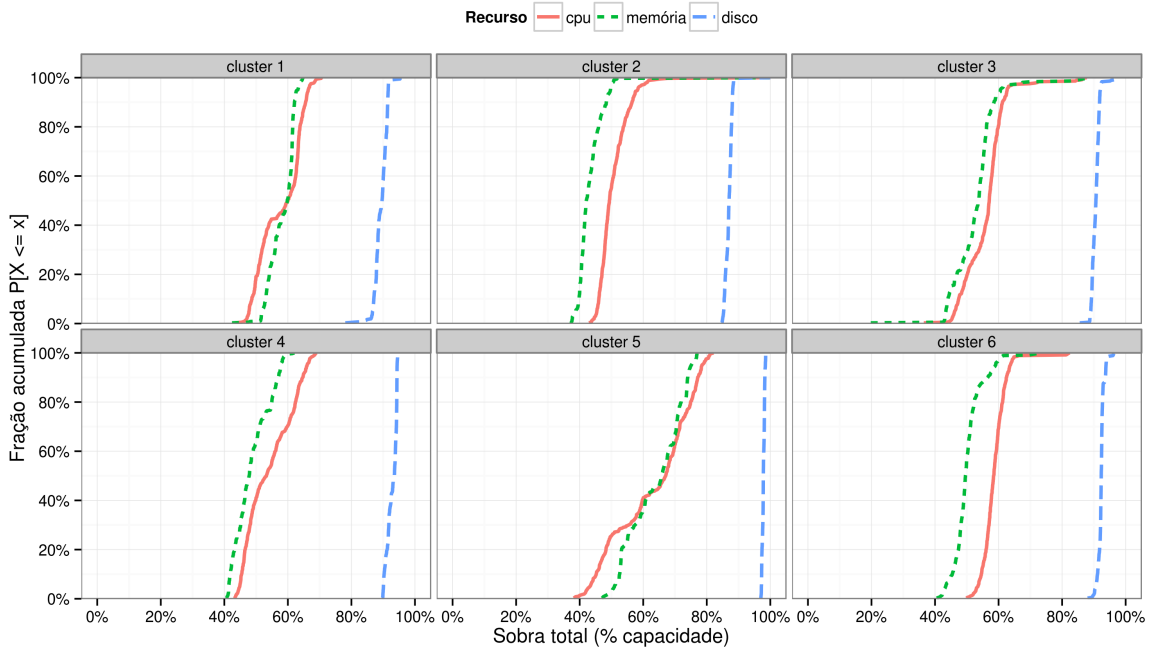

Figura 5.4: Função de Distribuição Acumulada (FDA) do total de sobras para CPU, memória e disco, medidos em intervalos de 5 minutos.

5.4.1 Previsão de séries temporais

O primeiro passo do modelo é estimar a quantidade de sobra de recursos que estará sempre disponível durante períodos de tempo futuros (i.e. o valor mínimo da sobra no período estimado). Neste estudo isto é feito usando modelos de previsão de séries temporais, baseados em dados históricos. O tempo é discretizado no modelo proposto, já que as medições da nuvem geralmente são feitas periodicamente em intervalos bem definidos e são naturalmente discretos. A predição realizada no tempo t para estimar as sobras de recurso $\hat{s}_{t+w}$ em um tempo futuro $t + w$, é feita com base na seguinte equação:

$$\hat{s}_{t+w} = F_w(s_1, \cdots, s_i, \cdots, s_{t-1}), \tag{5.1}$$

onde w é o horizonte da predição e $F_w(.)$ é uma função de predição que usa como entrada as sobras de recurso s_i para cada instante observado i anterior a t e realiza a predição para w unidades de tempo no futuro.

Qualquer técnica de predição (ou até mesmo uma combinação de várias técnicas [62])

pode ser usada neste modelo. A melhor escolha dependerá dos padrões encontrados nas observações e nos critérios estabelecidos por cada provedor.

5.4.2 Intervalos de confiança para predições

Como parte do modelo de predição, também é calculado um intervalo de confiança para as predições, que consiste em uma estimativa da faixa de valores na qual o valor real que está sendo estimado estará contido com uma certa probabilidade (i.e., o nível de confiança). O menor valor contido no intervalo de confiança da predição (i.e., o limite inferior do intervalo de confiança) da quantidade de sobras de recurso é escolhido como o valor estimado, de acordo com o nível de confiança escolhido. Quanto maior o nível de confiança, mais largo será o intervalo de confiança e mais conservadora será a predição – i.e., menores valores serão estimados para as sobras de recurso. Isto permite que diferentes níveis de confiança sejam usados para explorar os *trade-offs* entre reaproveitar uma quantidade maior de sobras de recurso através da classe *Economy* e a disponibilidade a ser obtida para estes recursos. Quanto maior o nível de confiança, menos recursos serão oferecidos, porém menor será o risco da quantidade de sobras estimadas não estarem disponíveis no futuro, o que causaria violações de SLO de disponibilidade.

O cálculo do intervalo de confiança é feito com base na variância dos erros de predição e no nível de confiança. Desta forma, calcula-se o intervalo de confiança da predição para as sobras de recursos no tempo futuro $t + w$ como:

$$\hat{s}_{t+w} \pm \hat{\sigma} \cdot z_q, \tag{5.2}$$

onde $\hat{\sigma}$ é o desvio padrão estimado para os erros das predições; q é o nível de confiança e z_q é o valor relativo ao q-percentil na distribuição normal padrão.

Embora a Equação 5.2 seja paramétrica e assuma que os erros de predição seguem uma distribuição normal, também foi mostrado que ela apresenta bons resultados para dados que não são normalmente distribuídos [28].

5.4.3 Ciclos de predição

Para definir SLOs de disponibilidade para a classe *Economy* que sejam úteis aos usuários, é preciso fazer predições para janelas futuras de longa duração. Porém, como a capacidade e a demanda da nuvem variam ao longo do tempo, há também uma variação na quantidade de sobra de recursos disponíveis. Geralmente, quanto maior a janela de tempo prevista no futuro, maior será a incerteza nas estimativas e, consequentemente, maior deve ser o intervalo de confiança para suas predições – i.e., a quantidade de sobra de recursos estimada deve ser menor, para que seja reduzido o risco de no futuro estarem disponíveis menos recursos que o planejado para a classe *Economy*.

As predições são geradas para uma janela de tempo $\Delta = (t, t + W]$, onde t é o tempo em que a predição é realizada e W é o tamanho da janela de predição. Novas predições são geradas ao fim de cada janela, de tal forma que as predições não se sobrepõem no tempo. Cada predição aplicada para um janela é denominada um *ciclo de predição*. A predição para um ciclo com uma janela Δ é dada por:

$$\hat{s}(\Delta) = \min\ \left(\hat{s}_{t+w} - \hat{\sigma} \cdot z_q\right), \quad \forall w \in [1,W]. \tag{5.3}$$

Note que os limites inferiores dos intervalos de confiança para vários instantes de tempo $t + w$ são usados na Equação 5.3. Além disso, o valor estimado será o valor mínimo das predições feitas dentro da janela, que é equivalente ao menor valor estimado para todos os intervalos de confiança calculados. Esta abordagem é adotada para possibilitar o uso de predições conservadoras, que buscam evitar a indisponibilidade de recursos no futuro de acordo com o nível de confiança escolhido pelo provedor.

O processo geral do modelo de predição é ilustrado na Figura 5.5. Observe que níveis de confiança maiores geram intervalos de confiança maiores, o que diminui o limite inferior deste intervalo e consequentemente gera menores estimativas para as sobras de recurso. As predições sem considerar intervalos de confiança (i.e., para o valor médio) também são apresentadas, indicando um cenário pouco conservador.

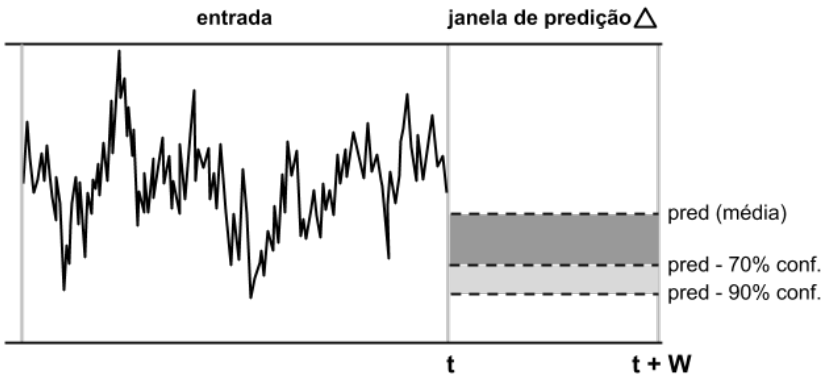

Figura 5.5: Ilustração do modelo para uma predição realizada no tempo t para uma janela Δ e para diferentes níveis de confiança.

5.5 Avaliação

Nesta seção, o modelo de predição proposto é avaliado usando dados de nuvens computacionais de produção. A metodologia de avaliação é descrita, indicando como foi feita a instanciação do modelo. Em seguida, é realizada uma comparação de diferentes técnicas de predição, uma análise do planejamento da nova classe de serviço explorando diferentes cenários de níveis de confiança e, por fim, uma análise da receita potencial do provedor ao oferecer a classe *Economy*, em comparação aos modelos adotados atualmente.

5.5.1 Metodologia

Nesta avaliação, foram usados rastros de carga de trabalho dos mesmos *clusters* da Google analisados na Seção 5.3. Porém, foi usado um período maior de coleta dos dados, contendo cargas de trabalho de abril de 2010 até dezembro de 2013 – um período de quase 4 anos. Usar medições feitas a cada 5 minutos para um período tão grande tornaria o tempo de execução dos experimentos inviável. Então, os dados originais com granularidade de 5 minutos foram transformados em valores diários, calculando a quantidade mínima da sobra de recursos observada em cada dia para cada *cluster*. Estas medições conservadoras representam uma avaliação do pior cenário para prover recursos na classe *Economy*, em que se assume que a menor quantidade de sobras de recursos observada durante um dia seria a mesma para o dia inteiro no qual ela ocorreu.

Apenas dados de sobras de recursos para CPU foram usados neste estudo. Uma avaliação

preliminar realizada para as sobras de memória indicou padrões similares, o que provavelmente daria resultados similares ao mostrados para CPU. Sobras para espaço em disco não foram realizadas, já que este recurso não se mostrou um gargalo nos *clusters* analisados (vide Figura 5.4).

Com base na experiência com os usuários da nuvem privada da Google, foi escolhido para este trabalho um tamanho de janela de predição de 6 meses[2], que permite que os usuários da nuvem façam planos de longo prazo com base na quantidade de recursos oferecidos na classe *Economy* e seus SLOs definidos para longos períodos.

Para cada iteração, os dados são divididos em conjunto de treino e conjunto de teste, onde o tamanho do período de teste Δ é sempre de 6 meses e o conjunto de treino contém todos os dados coletados antes do início do período de teste. As predições são executadas para todas as janelas de 6 meses observadas nos rastros, sem sobreposição de janelas, com um conjunto inicial de treino de 9 meses. O conjunto de treino é usado como entrada para as técnicas de predição de séries temporais e o conjunto de teste é usado para avaliar as predições. Após cada ciclo de predição, a janela é avançada em 6 meses e o conjunto de teste anterior é incluído no conjunto de treino seguinte. Os 45 meses de dados analisados dão um conjunto de treino inicial de 9 meses e 6 ciclos de predição com conjuntos de teste de 6 meses cada, exceto para o *cluster* 3, que possui uma menor amostra e permitiu a avaliação em apenas 4 ciclos de predição.

Neste estudo, foram avaliadas quatro técnicas de predição de séries temporais, descritas a seguir:

- **Mean**: a predição é feita com o cálculo da média aritmética dos valores de entrada da série temporal. Esta técnica geralmente é muito otimista, apresentando altos valores estimados para as sobras de recurso quando há algum pico nas observações.

- **Minimum**: a predição é feita com o cálculo do mínimo dos valores de entrada da série temporal. Esta técnica geralmente é muito conservadora, apresentando baixos valores estimados quando há pelo menos um valor muito baixo (vale) nas observações.

[2]O modelo proposto também se aplica para períodos de tempos menores, mas não satisfaria as necessidades dos usuários no cenário em que o estudo foi aplicado. Muito provavelmente, os resultados seriam ainda melhores para janelas menores, já que haveria menor incerteza nas predições.

- **Auto Regressive Integrated Moving Average** (ARIMA): é uma técnica de predição de séries temporais composta por três componentes [28]:

  1. Um componente de auto-regressão $AR(p)$, baseado na regressão linear das últimas p observações.
  2. Um componente de diferenciação que remove inclinações (ou tendências para dados não estacionários) dos dados, calculando a diferença entre observações consecutivas, aplicadas d vezes.
  3. Um componente de média móvel $MA(q)$, do inglês *moving average*, baseado em um modelo de regressão aplicado para os últimos q erros de predição.

  O critério de seleção AIC [18] foi usado para selecionar o modelo *ARIMA(p, d, q)* mais adequado para cada predição: ele estima a probabilidade dos dados originarem de cada um dos modelos usando estimadores de máxima verossimilhança (MLE, do inglês *Maximum Likelihood Estimators*), penalizando modelos que possuam uma quantidade muito grande de parâmetros [28].

- **Exponential Smoothing** (ETS): é uma técnica de predição de séries temporais que combina modelos baseados na média ponderada dos dados de entrada, com pesos decaindo exponencialmente para observações mais antigas. O modelo ETS possui três componentes: correção de Erros (E), Tendência (T) e Sazonalidade (S). Cada componente pode ser de um certo tipo: Nenhum, Aditivo, Multiplicativo, ou outras variações. As diferentes combinações de tipos para os três componentes resultam em métodos diferentes para o ETS, em que cada método possui um conjunto de parâmetros a serem estimados. O critério de seleção AIC também foi usado para encontrar a melhor combinação de tipos e parâmetros do ETS [43].

O pacote *forecast* do sistema *R* [63] possui implementações para as técnicas ARIMA e ETS [44], que foram usadas nesta avaliação. As outras duas técnicas foram implementadas usando funções básicas do *R*.

Foram analisados diferentes cenários para os intervalos de confiança das predições para cada técnica, variando o nível de confiança de 50% a 90%, com incrementos de 10%. Estes valores se mostraram suficientes, pois notou-se que níveis fora desta faixa apresentariam

estimativas ou muito altas ou muito baixas, resultando em uma disponibilidade muito baixa ou uma baixa quantidade de recursos oferecidos para a classe *Economy*.

Para calcular os intervalos de confiança para cada técnica, deve-se estimar o desvio padrão para os erros de predição ($\hat{\sigma}$), como mostra a Equação 5.2. Uma abordagem genérica para estimar o desvio $\hat{\sigma}$ é testar o modelo de predição para os mesmos dados usados no treinamento do modelo e calcular o desvio padrão dos erros de predição para estes dados. Como este método nem sempre traz uma boa precisão, existem fórmulas específicas para estimar a variância dos erros de predição para algumas técnicas de predição, que podem gerar melhores estimativas. A fórmula proposta por Box-Jenkins foi usada para estimar a variância dos erros de predição da técnica *ARIMA* [16], enquanto a fórmula proposta por Hyndman et al. [45] foi usada para a técnica *ETS*. Para o método *Mean*, $\hat{\sigma}$ foi estimado calculando o desvio padrão dos dados de entrada da série temporal. O intervalo de confiança para o método *Minimum* não foi calculado, pois as predições deste método já são muito conservadoras por padrão. Para realizar estes cálculos, também foi usada a implementação destas fórmulas contidas no pacote *forecast* do ambiente *R* [44].

As quatro técnicas de predição descritas são aplicadas usando o conjunto de treino como entrada e os resultados são avaliados comparando-os com o conjunto de teste. Os erros de predição são obtidos calculando a estimativa realizada no tempo t para a sobra mínima de recursos na janela de predição $\Delta = (t, t + W]$, para $W = 6$ meses, calculado como

$$\xi_{t+w} = s_{t+w} - \hat{s}(\Delta), \quad \forall w \in [1,W], \tag{5.4}$$

onde s_{t+w} é o valor real observado no tempo $t + w$ e $\hat{s}(\Delta)$ é o valor mínimo estimado para a janela de predição Δ calculado com a Equação 5.3. Note que para uma janela de predição Δ serão feitas predições para W instantes de tempo, mas apenas o menor deles é usado como estimativa. Como na janela existem W valores observados, serão calculados W valores de erro de predição para cada janela.

Erros positivos e negativos são analisados separadamente na avaliação, já que eles afetam o provedor de nuvem de forma diferente. Erros positivos significam que a sobra de recursos estimada foi menor do que as observações reais (i.e., foi subestimada). Neste caso, há um desperdício de recursos que poderiam ser reaproveitados, mas não resulta em grandes problemas; não há violações de SLO, apenas uma redução na possível receita gerada com a classe

Economy. Erros negativos significam que a sobra estimada foi maior do que a observada na realidade (i.e., foi superestimada). Neste caso, há uma escassez de recursos; se as sobras estivessem sendo reaproveitadas por usuários da classe *Economy*, em algum momento esses recursos seriam perdidos, o que poderia causar violações de SLO e resultar em penalidades para o provedor da nuvem. Durante o texto os termos **desperdício** e **escassez** de recursos serão usados para indicar erros positivos e negativos, respectivamente.

Como base de comparação para as predições com diferentes níveis de confiança, também foi avaliada a predição que estima a **média** da sobra de recursos, ou seja, sem considerar intervalos de confiança (i.e., com um nível de confiança de 0%). Além disso, um preditor **oráculo** também foi desenvolvido, que conhece os valores futuros da série temporal e gera a predição que resulta no maior valor possível que não resultará em nenhuma escassez de recursos (i.e., o menor valor de sobra de recurso observado na janela de predição, sem violar SLOs). Como na prática os dados futuros não são conhecidos pelo provedor no momento da predição, o preditor oráculo não pode ser aplicado na prática. Como existe uma variação diária na quantidade de sobra de recursos e o oráculo gera apenas uma estimativa para toda a janela de predição, o oráculo ainda poderá apresentar desperdício de recursos (i.e., erros positivos).

Os dados de séries temporais usados nas predições são normalizados para ajudar na comparação dos resultados entre *clusters* de tamanhos diferentes e para não expor dados confidenciais. Para cada predição aplicada no tempo t, os valores históricos das sobras relativas de recursos são usados como entrada, que são calculadas dividindo cada medida de sobra de recursos no conjunto de treino pela capacidade total de recursos na nuvem observada no momento t em que a predição é realizada. A sobra relativa é usada como dado de entrada durante esta avaliação, incluindo o cálculo dos erros de predição da Equação 5.4. Por simplicidade, na avaliação a "sobra relativa" é chamada apenas de "sobra de recursos". Note que se a capacidade do *cluster* for aumentada durante uma janela de predição e a capacidade excedente se tornar muito maior do que a observada no momento da predição, a sobra relativa pode ultrapassar o valor de 100%, ou seja, a sobra futura após o aumento da capacidade da nuvem pode se tornar maior do que a capacidade total observada no momento em que a predição foi realizada.

A Tabela 5.1 mostra um resumo dos cenários usados na avaliação das predições, apre-

sentando os parâmetros variados com os valores usados para cada um deles.

Parâmetro	**Valores**
Técnicas de predição	*Mean, Minimum, ARIMA, ETS*
Níveis de confiança	0% (média), 50%, 60%, 70%, 80%, 90%
Base de dados	Rastros de 6 clusters em produção da Google
Dados de entrada	Sobra total de CPU (relativa à capacidade no momento da predição)
Período dos dados	9 meses iniciais para treinamento + 6 janelas para teste = 45 meses

Tabela 5.1: Resumo dos cenários de avaliação das predições, com os valores usados para cada parâmetro.

5.5.2 Resultados das predições

A Figura 5.6 mostra a FDA empírica dos erros diários das predições para as sobras de CPU, feitas para janelas de 6 meses para todos os 6 clusters e para todas as quatro técnicas avaliadas. As linhas pretas cheias (mais escuras) representam as predições sem considerar intervalos de confiança, que chamamos de predição para a *média*; as linhas azuis cheias (mais claras) representam as predições ao se considerar intervalos de confiança para as predições. Quanto mais claras as linhas, maior o nível de confiança usado. A linha verde tracejada mostra os erros das predições para o preditor oráculo e a linha cinza vertical tracejada no ponto zero divide os erros positivos dos negativos.

Observa-se que quanto maior for o nível de confiança, menores serão os erros de predição tendo uma curva da FDA mais à esquerda – i.e., os valores estimados vão diminuindo, então se tem mais desperdício (erros positivos) e menos escassez de recursos (erros negativos). Ou seja, quanto maior o nível de confiança, menos recursos serão oferecidos na classe *Economy* e alguns recursos que poderiam ser reaproveitados serão desperdiçados. Porém, serão menores os riscos dos recursos estimados não estarem disponíveis no futuro, o que poderia causar violações de SLO.

Como cada provedor da nuvem pode ter preferências diferentes com relação ao equilíbrio desejado entre desperdício e escassez de recursos que pretendem obter ao oferecer uma classe como a *Economy*, o objetivo deste estudo é tornar essa escolha flexível, indicando uma técnica de predição que permita que o provedor analise os *trade-offs* entre a quantidade

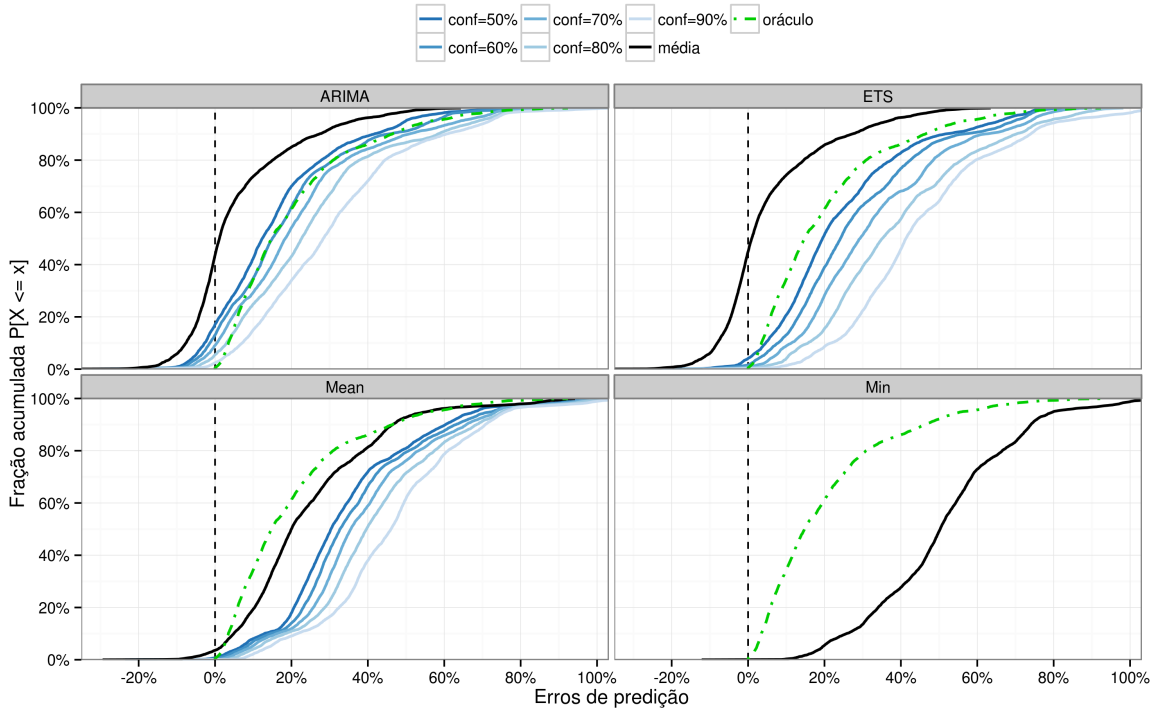

Figura 5.6: Função de Distribuição Acumulada (FDA) dos erros das predições para as sobras de CPU em janelas de 6 meses, para 4 técnicas de predição.

de desperdício e escassez de recursos. Estas foram as conclusões com relação às técnicas avaliadas:

- A técnica *Mean* não apresentou escassez de recursos para nenhum cenário que usa intervalos de confiança, mas todas as predições foram muito conservadoras, apresentando muito desperdício.
- A técnica *Minimum* é bastante conservadora. Ela também não apresenta escassez, mas resulta em muito desperdício.
- A técnica *ARIMA* produziu escassez de recursos mesmo para cenários com níveis de confiança muito conservadores.
- A técnica *ETS* produziu a melhor faixa de opções: a escassez de recursos diminui gradualmente ao aumentar o nível de confiança, chegando ao ponto de quase não haver

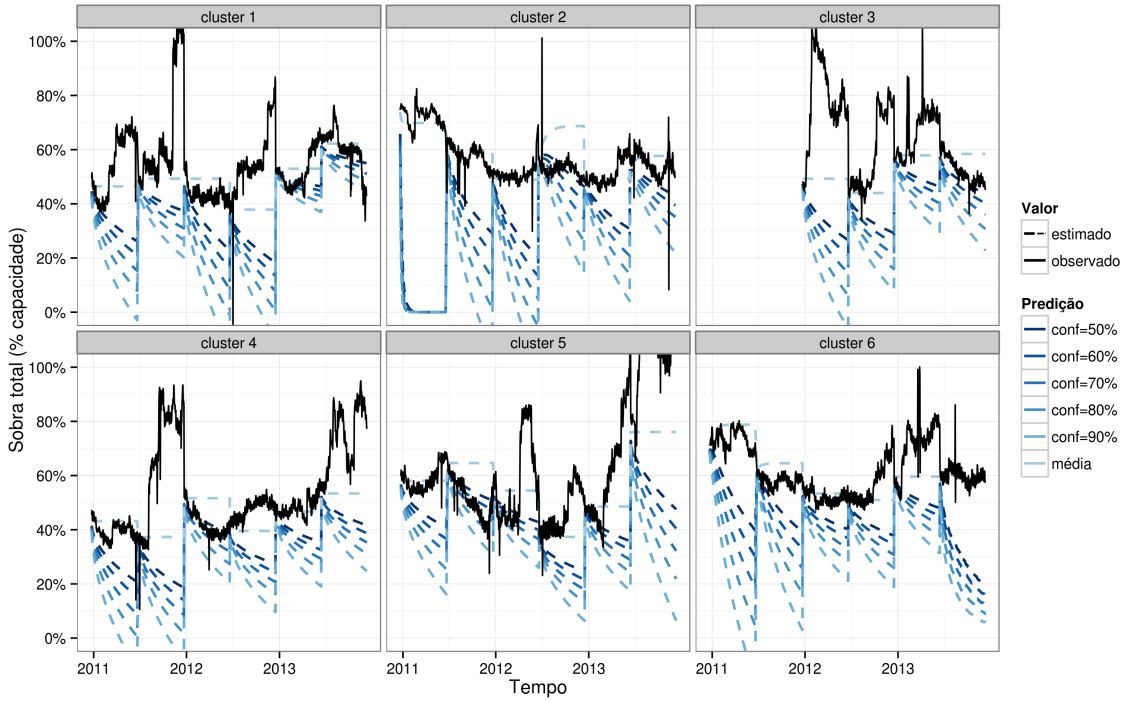

Figura 5.7: Sobras de CPU no tempo observadas e estimadas com a técnica de predição ETS, para diferentes níveis de confiança.

desperdício em cenários mais conservadores, enquanto o desperdício aumenta gradualmente para níveis de confiança maiores.

Com base nestes resultados, a técnica *ETS* foi escolhida para ser usada durante o restante da avaliação, por permitir uma melhor exploração dos *trade-offs* entre escassez e desperdício ao se variar os cenários de níveis de confiança.

A Figura 5.7 mostra a sobra relativa de CPU observada e as estimativas realizadas no tempo para a técnica de predição ETS. A linha cheia (preta) mostra as sobras reais observadas e as linhas tracejadas (azuis) mostram os valores estimados, sendo que linhas mais claras representam maiores níveis de confiança. O cenário sem considerar intervalos de confiança também é mostrado (*média*), representado pela linha tracejada mais clara. Como dito anteriormente, as sobras de CPU apresentadas são normalizadas pela capacidade de cada *cluster* observada no dia em que a predição é aplicada – no último dia do conjunto de treino.

O modelo de predição proposto é intencionalmente conservador para lidar com a variabilidade dos dados e evitar prejuízos maiores causados pela escassez de recursos oferecidos

pela classe *Economy*. Como resultado disso, podemos observar que os intervalos de confiança são maiores e, consequentemente, os valores estimados são menores quando: há uma grande variação nas sobras observadas nos dados de treino; o horizonte da predição é maior; e o conjunto de treino é menor (i.e., o tamanho da amostra é menor, como nos primeiros ciclos de predição).

O método proposto neste trabalho continuaria funcionando se uma técnica de predição diferente resultasse em predições melhores para outras cargas de trabalho. Um provedor pode realizar a mesma avaliação descrita neste trabalho para escolher a técnica que melhor se adéqua para a carga de sua nuvem. Espera-se que os resultados apresentados neste estudo possam ser generalizados para outros ambientes de nuvem, com boas chances de serem robustos e bem representativos, tendo em vista as características das cargas dos *clusters* utilizadas na avaliação neste trabalho, que possuem: longa duração, larga escala, grande diversidade de aplicações e variações de comportamento no tempo (e.g., *bursts* de demanda).

5.5.3 Análise do planejamento da reserva

As predições realizadas na seção anterior foram usadas para fazer o planejamento da reserva da nova classe de serviço da nuvem. A quantidade de recursos a serem oferecidos na classe *Economy* foi definida a partir dos valores estimados das sobras de recurso para janelas de predição de 6 meses.

A métrica **disponibilidade na janela de predição** é definida como sendo a fração de dias em uma janela de predição que não apresenta nenhuma escassez de recursos. Ou seja, é a fração de dias que não apresentam erros negativos de predição e, por isso, a quantidade de sobras de recursos definida como a reserva para a classe *Economy* estará de fato disponível para ser oferecida aos usuários, sem risco de violar SLOs. Esta métrica foi calculada de uma forma bastante conservadora, representando um cenário de pior caso. Na prática, se em algum momento houver uma quantidade menor de sobra de recursos do que a reserva definida para a classe *Economy*, isto só resultaria em uma indisponibilidade visível caso estas sobras estivessem sendo de fato requisitadas e usadas por algum usuário. Além disso, um dia inteiro é considerado como indisponível mesmo que em uma única janela de 5 minutos houvesse alguma escassez de recursos, por causa da granularidade dos dados. Por causa da falta de dados com menor granularidade, preferiu-se usar uma medida conservadora para

uma análise de pior caso. Se uma escassez de recursos em curtos períodos de tempo (e.g., alguns minutos) for aceitável para algumas aplicações, avaliar com uma menor granularidade nos dados e no cálculo da métrica usada certamente resultaria em maiores disponibilidades nas janelas de predição, tornando a classe *Economy* ainda mais atrativa.

O nível de confiança afeta a quantidade de recursos a serem oferecidos pela classe *Economy*, assim como a disponibilidade na janela de predição para estes recursos. A Figura 5.8 mostra como se dá o balanço entre o tamanho médio da classe *Economy* estimado pela técnica ETS (i.e., sua estimativa média para as sobras de recurso) e a disponibilidade na janela de predição obtida, para predições com diferentes níveis de confiança. A linha tracejada vertical indica a média das estimativas do preditor oráculo, que representa o maior tamanho para a classe *Economy* que pode oferecer 100% de disponibilidade na janela de predição. Em alguns casos, as predições da técnica ETS estimam tamanhos maiores que a do oráculo, resultando em uma menor disponibilidade. As linhas apresentadas em torno dos pontos, em forma de barras de erro, representam o desvio padrão para as predições realizadas para cada janela.

No cenário mais conservador avaliado (nível de confiança de 90%), a disponibilidade na janela de predição ficou entre 98,9% e 100% para os 6 *clusters* e a quantidade de recursos oferecidos na classe *Economy* foram de 6,7 a 17,3% da capacidade do *cluster*. No cenário menos conservador (nível de confiança de 50%), a disponibilidade foi inferior (entre 88,7% e 100%), mas a quantidade de recursos que poderiam ser oferecidos pela classe *Economy* aumentou para a faixa de 28,4% a 40,4%. O preditor oráculo (inviável na prática) estimou que em média 30,7% a 52,1% da capacidade dos *clusters* poderia ser reaproveitada pela classe *Economy* sem que houvessem violações de disponibilidade.

Para os *clusters* 2, 3 e 6, a disponibilidade ficou próxima de 100% para todos os níveis de confiança. Este resultado reflete o conservadorismo adotado para o modelo de predição, além de uma variabilidade moderada observada nos dados de treino em alguns casos.

Ao comparar os resultados para diferentes níveis de confiança, provedores da nuvem podem escolher um valor adequado que satisfaça às suas necessidades, balanceando a força dos SLOs de disponibilidade e a quantidade de recursos a serem oferecidos na classe *Economy*.

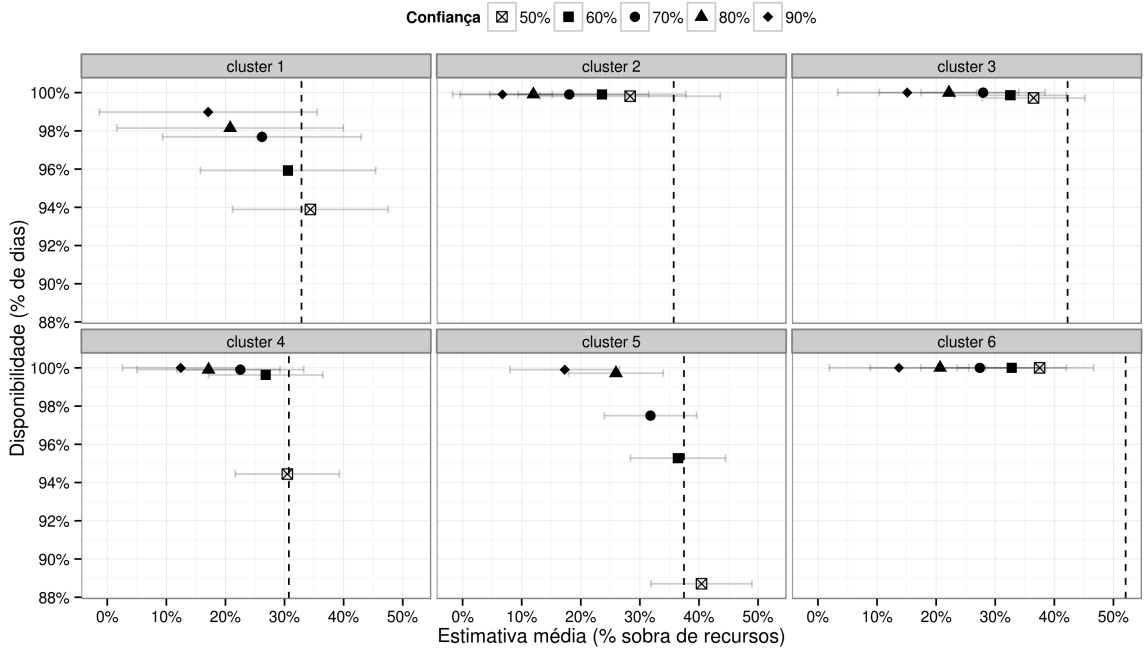

Figura 5.8: Quantidade de recursos estimada para a classe *Economy* e suas disponibilidades nas janelas de predição de 6 meses.

5.5.4 Análise da receita potencial

Seria mais benéfico para o provedor de nuvem oferecer parte das sobras dos recursos através da classe *Economy*, com garantias de disponibilidade, ou oferecer toda a sobra na classe *Opportunistic*, sem garantia alguma? Esta seção explora diversos cenários para responder esta questão com relação à receita do provedor. A ideia é analisar a possível melhora na receita do provedor ao oferecer a nova classe. Os símbolos usados no modelo de receita potencial apresentado nesta seção são descritos na Tabela 5.2.

Inicialmente, assume-se que todas as sobras de recursos que são oferecidas pelo provedor são consumidas pelos usuários. Em seguida, discute-se o que pode tornar esta suposição inválida na prática e quais seriam as consequências.

Para comparar as receitas totais, calcula-se primeiro a receita G_r gerada pelos serviços oferecidos pela classe r como:

$$G_r = C \cdot T \cdot f_r \cdot p_r,$$

Símbolo	Descrição
r	Classe de serviço oferecida na nuvem
e	Identifica uma medida que se refere à classe *Economy*
op	Identifica uma medida que se refere à classe *Opportunistic*
G_r	Receita gerada dos serviços oferecidos pela classe r
C	Capacidade total dos recursos do provedor da nuvem
T	Duração do período (em unidades de tempo) de base para o cálculo da receita
f_r	Fração da capacidade total alocada para a classe r
f_{sobra}	Fração da capacidade que está sobrando e pode ser reaproveitada
p_r	Preço que o usuário paga por recursos da classe r por unidade de tempo
X_r	Penalidade paga pelo provedor ao violar o SLO da classe r
γ_r	Disponibilidade medida para a classe r
P_r	Receita que o provedor obtém ao oferecer novamente as sobras de recursos usando a classe r
Θ	Proporção da receita – receita da classe *Economy* divido pela receita da classe *Opportunistic*

Tabela 5.2: Símbolos usados no modelo de receita potencial do provedor de IaaS ao oferecer diferentes classes de serviço.

onde C é a capacidade total dos recursos do provedor da nuvem, f_r é a fração da capacidade total reservada para a classe r, p_r é o preço dos recursos por unidade de tempo para a classe r e T é a duração do período (em unidades de tempo) para o qual a receita está sendo calculada.

Se o provedor não oferecer a classe *Economy*, toda a sobra de recursos será oferecida através da classe *Opportunistic*. A receita gerada para o provedor ao se reaproveitar a capacidade em excesso da nuvem usando apenas a classe *Opportunistic* é dada por:

$$G_{op} = C \cdot T \cdot f_{sobra} \cdot p_{op},$$

onde f_{sobra} é a fração de capacidade que está sobrando e pode ser reaproveitada, e p_{op} é a média do preço dos recursos da classe *Opportunistic* durante o período medido.

Quando parte da sobra de recursos é oferecida através da classe *Economy*, a sobra de recursos restante é oferecida na classe *Opportunistic*. Desta forma, calcula-se a receita G_e do provedor gerada pelo reaproveitamento da capacidade em excesso quando a classe *Economy* é usada, como:

$$G_e = C \cdot T \cdot [f_e \cdot p_e + (f_{sobra} - f_e) \cdot p_{op}]$$

onde f_e é a fração da capacidade total alocada para a classe *Economy* e $(f_{sobra} - f_e)$ é o restante da sobra de recursos que será oferecida pela classe *Opportunistic*.

Ao definir garantias de QoS para os recursos através de SLOs de disponibilidade, o provedor paga penalidades quando esses SLOs não são cumpridos. Nesta análise, a penalidade do provedor foi definida de acordo com o SLA oferecido na nuvem da Amazon EC2: se a média da disponibilidade mensal do serviço for menor que 99%, o servidor paga ao usuário uma penalidade (através de créditos para o usuário) equivalente a 30% da conta do usuário no mês [2].[3] O impacto das penalidades para o provedor podem ser pouco significativas, podendo até criar um incentivo para o provedor descumprir SLOs ocasionalmente. Porém, outros fatores como a reputação do provedor também serão importantes para sua receita a longo prazo.

A penalidade X_r para a classe r é definida como uma fração da receita do provedor gerada a partir das sobras de recursos no período medido. A penalidade para a classe *Opportunistic* é sempre zero ($X_{op} = 0$), já que o SLA para esta classe não possui garantias de disponibilidade. A penalidade para violações de disponibilidade para a classe *Economy* é definida como:

$$X_e = \begin{cases} 30\%, & \text{se } \gamma_e \leq 99\% \\ 0, & \text{caso contrário} \end{cases}$$

onde γ_e é a disponibilidade (*availability*) observada para a classe *Economy*.

A receita total P_r que o provedor obtém com o reaproveitamento da sobra de recursos ao oferecer a classe r consiste na receita gerada pelo serviço menos as penalidades, definida como:

$$P_r = G_r \cdot (1 - X_r)$$

Duas suposições bastante conservadoras foram feitas devido à limitação dos dados disponíveis para esta análise:

1. A granularidade dos dados usados na avaliação é de uma observação por dia, então é assumido um cenário de pior caso em que um único momento de indisponibilidade observado em um dia faz com que o dia inteiro seja considerado como indisponível.

[3] As penalidades oferecidas por outros provedores atuais são similares.

Então, uma única indisponibilidade ocorrida durante um mês resulta em 1 dia em 30 indisponível (i.e., 96,7% dos dias disponíveis), que já é uma disponibilidade menor do que o limite de 99% definido no SLO. Ou seja, um único momento de indisponibilidade no mês já causaria penalidades para o provedor para recursos da classe *Economy*.

2. A quantidade de usuários afetados por cada período de indisponibilidade não é conhecida, então é assumido que todos os usuários são afetados em todos os períodos de indisponibilidade. Isto é equivalente a aplicar a penalidade proporcionalmente a toda a receita gerada pela classe *Economy* em caso de violações de disponibilidade.

Na prática, a disponibilidade real será significativamente maior do que a medida nesta análise e as penalidades apenas seriam pagas para um conjunto menor de usuários, que foram realmente afetados pela falta de recursos. Desta forma, se dados com menor granularidade fossem usados, esta análise muito provavelmente apresentaria maior receita para a classe *Economy*.

A comparação da receita que o provedor obteria com a capacidade em excesso usando apenas a classe *Opportunistic* com a receita quando a classe *Economy* também é usada pode ser feita calculando a proporção da receita Θ para os dois cenários:

$$\begin{aligned} \Theta &= \frac{P_e}{P_{op}} \\ \Theta &= \frac{[f_e \cdot p_e + (f_{sobra} - f_e) \cdot p_{op}] \cdot (1 - X_e)}{f_{sobra} \cdot p_{op}} \end{aligned}$$

Note que a proporção da receita Θ é independente da capacidade C e das unidades de tempo T do período medido, pois essas variáveis são canceladas ao dividir uma receita pela outra.

A análise da receita potencial da classe *Economy* é baseada nos resultados das predições apresentados anteriormente. São usadas as frações da capacidade estimadas para serem oferecidas na classe *Economy* e a disponibilidade resultante para diferentes níveis de confiança ao usar o método ETS, como mostrado na Figura 5.8). Também são explorados diferentes opções de preço para os recursos oferecidos na classe *Economy*, mostrando em que casos o uso desta nova classe pode gerar maior receita do que oferecer sobras de recurso apenas de forma oportunista, sem garantias de disponibilidade.

Para definir os preços dos recursos da classe *Opportunistic*, foi usado o histórico de preços de recursos da classe *Spot Instances* da Amazon EC2, que oferece recursos de forma oportunista. Os preços foram medidos de dezembro de 2013 a março de 2014 para diferentes tipos de instância, com os quais foi calculado o **preço relativo dos recursos**: o preço médio das instâncias *Spot* dividido pelo preço (constante) de uma instância similar na classe *On-demand*. Foram coletados preços para 4 tipos de instância de propósito geral da Amazon EC2 (*m1.small, m1.medium, m1.large, m1.xlarge*) para um *cluster* na região *us-west-1*. Nesses dados, a média dos preços para instâncias *Spot* varia entre 15,4% e 59,9% do preço *On-demand*, para os 4 tipos de instância analisados. Considerando a média de todos os tipos de instância agregados, a média do preço relativo dos recursos foi de 31,8%, que foi o valor usado nesta avaliação como preço relativo dos recursos da classe *Opportunistic* (i.e., $p_{op} = 31{,}8\%$)

Como neste estudo pretende-se identificar preços razoáveis para os recursos da classe *Economy* com garantias de QoS, foram explorados diferentes preços relativos de recurso para esta classe, variando p_e de 30% a 100% do preço *On-demand*. Valores menores ou maiores não foram explorados; é razoável assumir que o preço dos recursos oferecidos na classe *Economy* devem pelo menos valer o mesmo que o preço médio dos recursos da classe *Opportunistic* ($p_e \geq p_{op}$) por eles possuirem garantias de QoS, mas não devem ser mais caros que os recursos *On-demand* ($p_e \leq 1$), já que recursos reaproveitados das capacidade em excesso provavelmente possuirão disponibilidade inferior.

A Figura 5.9 mostra a proporção da receita Θ e a média da disponibilidade quando diferentes combinações de nível de confiança e preço são usadas para recursos da classe *Economy*. As diferentes formas e cores dos pontos representam diferentes níveis de confiança. Os diferentes tamanhos dos pontos representam diferentes preços para os recursos *Economy*, relativos aos preços *On-demand*. A linha tracejada mais à esquerda apresenta o ponto em que oferecer a classe *Economy* se torna mais lucrativo do que oferecer apenas a classe *Opportunistic* ($\Theta = 1$), de tal forma que os pontos à direita desta linha representam cenários em que é mais lucrativo oferecer a classe *Economy*. A linha tracejada mais à direita mostra o valor máximo possível para a proporção da receita, que seria obtido usando o preditor oráculo e definindo o preço dos recursos *Economy* igual ao preço dos recursos *On-demand* – um preço relativo igual a 1.

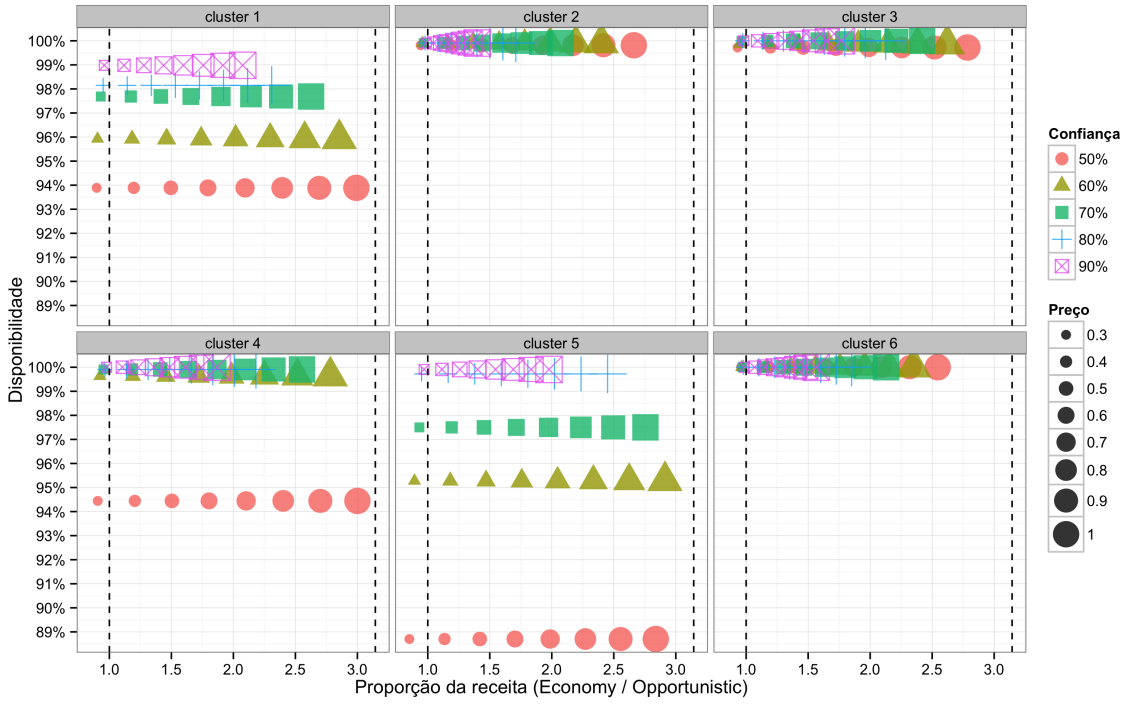

Figura 5.9: Comparação das receitas ao oferecer a classe *Economy* e sua disponibilidade para diferentes cenários de nível de confiança e preços para os recursos *Economy*.

Como discutido anteriormente, quanto maior o nível de confiança (i.e., predições mais conservadoras), maior tende a ser a disponibilidade, o que também tornam menores as penalidades pagas pelo provedor. Porém, o aumento do nível de confiança resulta em menos recursos sendo oferecidos na classe *Economy*, o que pode diminuir a receita gerada para os recursos reaproveitados da capacidade excedente. Portanto, o provedor precisa analisar esses *trade-offs* e escolher um nível de confiança que resultará em um bom balanço entre a disponibilidade e a quantidade de recursos a serem oferecidos na classe *Economy*, que poderá resultar em bons SLOs, baixas penalidades e consequentemente gerar boas receitas para os serviços baseados na sobra de recursos.

Observa-se que a proporção da receita aumenta quando o preço dos recursos oferecidos na classe *Economy* é aumentado. Como esperado, a receita da classe *Economy* é menor que a receita da classe *Opportunistic* quando seu preço relativo é definido como $p_e = 30\%$, que é ligeiramente menor do que o preço da classe oportunista $p_{op} = 31.8\%$. Porém, para quase todos os outros cenários de preços, oferecer parte dos recursos na classe *Economy* gera uma

receita total maior, mesmo pagando penalidades no caso de violações de SLO.

Para as predições mais conservadoras (i.e., nível de confiança de 90%), a disponibilidade média ficou entre 98,9% e 100%. Assumindo que 70% do preço *On-demand* é um valor aceitável para o preço dos recursos da classe *Economy* com este nível de disponibilidade, a receita ao oferecer a classe *Economy* com este cenário seria de 22% a 60% maior do que a receita obtida ao oferecer apenas a classe *Opportunistic* para as sobras de recurso, mesmo pagando penalidades em casos de violação de SLO. Para as predições menos conservadoras (i.e., nível de confiança de 50%), a média da disponibilidade ficou entre 88,7% e 100%. Se for considerado para os recursos *Economy* um preço de 50% do *On-demand* com este SLO de disponibilidade, o aumento na receita ao oferecer a classe *Economy* seria de 41% a 50%, comparando com o uso apenas da classe *Opportunistic* para a capacidade em excesso.

Para estes resultados, assumiu-se que a sobra de recursos seria totalmente reaproveitada e consumida por usuários, independentemente de seu preço e disponibilidade. Na prática, preços altos, SLOs fracos e/ou excesso de violações de SLO poderiam afetar a demanda para estes recursos. Nesta análise, não se sabe qual seria o impacto de cada um destes fatores para a demanda da nuvem, pois não haviam dados suficientes para tirar conclusões do quanto os usuários estariam dispostos a pagar para diferentes níveis de SLO de disponibilidade, ou quantas violações um usuário poderia tolerar até mudar de provedor. Porém, é razoável assumir que haveria uma boa demanda para a classe *Economy* se o preço dos seus recursos fosse menor do que os *On-demand* e que bons SLOs fossem oferecidos. Qualquer capacidade não vendida pode ser sempre oferecida através da classe *Opportunistic*, então há poucas desvantagens para o provedor em oferecer a classe *Economy*, contanto que suas garantias de disponibilidade sejam boas o suficiente para atrair usuários e que suas violações sejam evitadas. Como foi observado anteriormente, os cálculos para as penalidades são extremamente conservadores, já que consideram um cenário de pior caso, então a proposta desta nova classe parece realizável, com potenciais benefícios tanto para o provedor quanto para os usuários.

Nos capítulos seguintes são apresentadas estratégias de controle de admissão e planejamento de capacidade que permitem ao provedor oferecer diferentes classes de serviço e cumprir seus SLOs, fazendo o planejamento para todas as classes de serviço oferecidas pelo provedor da nuvem.

5.6 Conclusões do estudo

A elasticidade oferecida pela computação na nuvem apresenta vários desafios para o provedor. Para oferecer garantias fortes de disponibilidade, provedores precisam fazer superprovisão de recursos, que resulta em uma baixa utilização de sua capacidade e aumenta os custos de sua infraestrutura. Uma forma comum de melhorar isto e aumentar a utilização da nuvem é oferecer as sobras de recursos de forma oportunista (i.e., sem qualquer garantia de QoS). Porém, este tipo de recurso não é adequado para vários casos de uso da nuvem, especialmente para serviços que executam por longos períodos de tempo e precisam de alguma garantia mínima de disponibilidade. Além disso, recursos oportunistas geralmente são oferecidos com preços bem mais baixos para compensar a ausência de garantias, trazendo poucos ganhos ao provedor.

Um forma de tornar a capacidade em excesso da nuvem mais útil e lucrativa é oferecendo parte dessas sobras de recursos com SLOs de disponibilidade para múltiplos meses. Neste estudo foi mostrado como isso pode ser feito, definindo uma nova classe de serviço chamada *Economy*. Foram usadas predições baseadas em técnicas de previsão de séries temporais para planejar a capacidade a ser oferecida através desta nova classe, estimando a quantidade de sobras de recurso que estarão disponíveis em longas janelas de tempo e que podem ser oferecidas novamente com garantias de longo prazo para disponibilidade. Foram usados intervalos de confiança para quantificar os riscos de violações de SLO e para se analisar os *trade-offs* entre este risco e a quantidade de recursos a serem oferecidos por esta classe.

A abordagem proposta foi avaliada usando dados que compreendem um período de quase 4 anos obtidos de 6 *clusters* da nuvem privada de produção da Google, abrangendo dezenas de milhares de máquinas. Acreditamos que a carga de trabalho analisada é representativa para uma grande variedade de nuvens computacionais, tendo em vista o longo período de tempo (não visto em outros estudos de nuvem); a larga escala dos *clusters*; o uso de diferentes *clusters*; e a diversidade das suas cargas – cada *cluster* executou milhares de aplicações e os *clusters* tinham padrões de carga significativamente diferentes.

Os resultados mostraram que mesmo com uma abordagem conservadora, 6.7 – 17.3% dos recursos poderiam ser oferecidos pela classe *Economy* com uma disponibilidade de 98,9% ou maior. Também foi visto que um provedor pode aumentar sua receita ao vender parte

das sobras de recursos através da classe *Economy*. Nos exemplos analisados, o aumento na receita foi de até 60%, mesmo após o pagamento de penalidades por conta de violações de SLO.

Por fim, oferecer uma nova opção de classe de serviço pode ser muito benéfico também para os usuários da nuvem: eles podem ter boas garantias de disponibilidade e de longo prazo para planejar a execução das suas aplicações e cumprir seus requisitos dentro do orçamento, pagando menos do que pagariam para recursos das classes *Reserved* e *On-demand* e obtendo melhores garantias do que a classe *Opportunistic*.

Capítulo 6

Controle de admissão em nuvens de IaaS com múltiplas classes

No capítulo anterior, mostrou-se como uma nova classe de serviço pode ser introduzida em uma nuvem existente, a partir de sobras de recursos não usados e com metas de disponibilidade para esta classe. O estudo evidenciou as vantagens de se oferecer múltiplas classes com garantias de QoS, possibilitando o aumento da utilização e da receita do provedor de nuvem. Neste capítulo[1], dá-se mais um passo em direção a uma gerência de recursos da nuvem geral e flexível, ao planejar não apenas uma nova classe mas um ambiente completo de nuvem de IaaS com diferentes classes de serviço. Para isto, propõe-se um modelo de controle de admissão para nuvens de IaaS com múltiplas classes, que permite ao provedor definir e cumprir SLOs de disponibilidade de VM para todas as classes oferecidas.

6.1 Introdução

Este estudo aborda o problema do controle de admissão em nuvens computacionais de IaaS. Com mecanismos de controle de admissão, o provedor pode reduzir violações de SLO de disponibilidade de VMs rejeitando novas requisições de VM quando a capacidade da nuvem

[1]Este capítulo é baseado no artigo publicado na *IEEE International Conference on Cloud Computing Technology and Science (CloudCom'15)* [24] e no artigo submetido para o periódico *IEEE Transactions on Parallel and Distributed Systems (TPDS)* que ainda em fase de revisão, mas foi disponibilizado como *Technical Report* [25].

não for suficiente para acomodar a demanda. Assume-se que rejeitar novas requisições é menos prejudicial do que comprometer a QoS das requisições já admitidas e violar seus SLOs. Um desafio maior para o provedor se dá quando ele oferece múltiplas classes de serviço, pois decisões de controle de admissão podem ter consequências diferentes dependendo das classes que elas afetam.

Neste contexto, um problema importante para provedores de IaaS é o de como tomar decisões eficientes de controle de admissão ao oferecer múltiplas classes. Nesta tese é proposto um modelo baseado em predições para solucionar este problema, buscando maximizar a taxa de admissão de requisições e cumprir os SLOs de disponibilidade de VM definidos para cada classe. Diferentes técnicas de predição foram usadas para prever a capacidade disponível para cada classe no futuro, usando séries temporais históricas como entrada. Com base nas predições, o método proposto define dinamicamente quotas para cada classe, limitando a quantidade de recursos que podem ser requisitados, de tal forma que novas requisições são rejeitadas se a quota referente à sua classe for excedida.

As principais contribuições deste estudo são:

- A caracterização da carga de trabalho e da disponibilidade de recursos de uma nuvem computacional em produção;
- A formalização de parte do problema de gerência de recursos da nuvem, formulado como um modelo de otimização;
- A elaboração de um novo modelo de controle de admissão baseado em predição para nuvens de IaaS com múltiplas classes;
- A avaliação do modelo proposto, instanciando-o com diferentes métodos de predição e simulando diferentes cenários usando rastros de uma nuvem em produção;
- A demonstração de que heurísticas preditivas de controle de admissão são menos sensíveis a mudanças no ambiente da nuvem e geram poucas violações de SLO mesmo em cenários de alta contenção, nos quais uma heurística gulosa e outra que não usa controle de admissão apresentam quantidades de violações significativamente maiores.

O restante deste capítulo está estruturado da seguinte forma. A Seção 6.2 formaliza parte do problema da gerência de recursos da nuvem como um modelo de otimização. A Seção 6.3 apresenta a caracterização da carga de trabalho e capacidade disponível na nuvem, com base em rastros de nuvem em produção. A Seção 6.4 apresenta o modelo de controle de admissão baseado em predição proposto nesta tese. A Seção 6.5 descreve como o modelo foi instanciado e define o ambiente de simulação usado na avaliação. A Seção 6.6 apresenta a avaliação das heurísticas que implementam o modelo comparando-as com outras heurísticas. Por fim, a Seção 6.7 apresenta as conclusões do estudo.

6.2 Formalização do problema

Nesta tese são abordados problemas relacionados à gerência de recursos de nuvens computacionais de IaaS, com foco em decisões de planejamento de capacidade e controle de admissão. Nesta seção, esses problemas são descritos com mais detalhes e formalizados como um modelo de otimização, com foco em métodos de controle de admissão.

A Figura 6.1 mostra parte do processo de gerência de recursos estudado. Na fase de planejamento de capacidade, o provedor decide a capacidade da nuvem necessária para executar a carga de trabalho dos usuários, de tal forma que os SLOs definidos para cada classe sejam cumpridos e os custos sejam minimizados. Na fase de controle de admissão, o provedor decide quais requisições serão rejeitadas, buscando admitir o máximo de requisições possíveis de forma a garantir que seus SLOs sejam cumpridos. Na fase de escalonamento, o provedor define quais requisições de VM estarão executando e em quais máquinas elas serão alocadas [58].

A avaliação de decisões de gerência de recursos é feita para um **período de observação**, que é dividido em intervalos de tempo de duração finita chamados de **épocas** [57]. Uma nova época é iniciada quando pelo menos um desses eventos ocorrem: (1) a chegada de uma nova requisição de VM; (2) a finalização de uma requisição de VM; ou (3) a mudança na capacidade disponível da nuvem, que ocorre quando máquinas ficam indisponíveis (e.g., por falhas ou manutenção) ou voltam a ficar disponíveis (e.g., após reparos ou atualizações).

A sequência de épocas em um período de observação é definida como $\langle E_1, \cdots, E_i, \cdots, E_N \rangle$, onde N é a quantidade de épocas neste período. Sejam $b(E_i)$

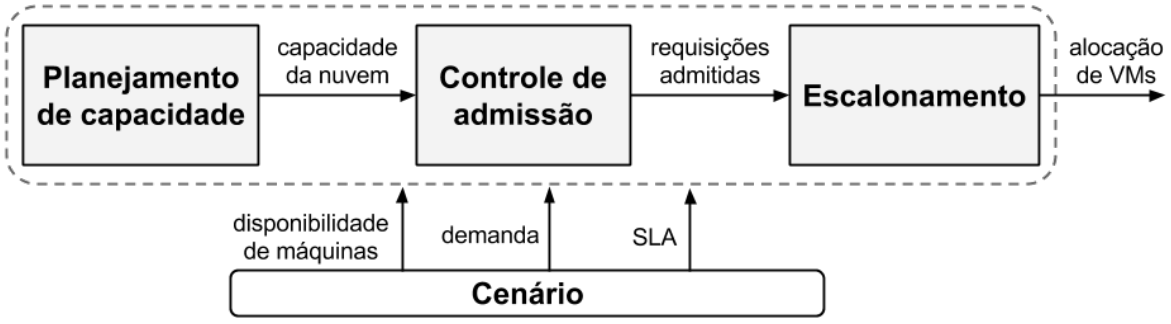

Figura 6.1: Parte do processo de gerência de recursos de nuvens computacionais.

e $e(E_i)$ respectivamente os registros de tempo do começo e do fim da época E_i, define-se o período de observação como sendo o intervalo de tempo $[b(E_1), e(E_N)]$. Define-se também $\Delta_i = e(E_i) - b(E_i)$ como a duração da época E_i e $\Delta = e(E_N) - b(E_1)$ como a duração total do período de observação, em unidades de tempo.

Neste trabalho, o único tipo de recurso considerado nas decisões de gerência de recursos foi a CPU, medida em número de *cores* (núcleos). Esta simplificação foi feita com base em estudos que analisaram a carga de trabalho da nuvem e mostraram que CPU é o recurso que possui a maior utilização, sendo identificado como o gargalo do sistema [64; 23]. De toda forma, outros tipos de recurso como memória e disco podem ser considerados em trabalhos futuros, fazendo com que o provedor aloque VMs se e somente se houver disponível uma máquina com capacidade suficiente para acomodar a VM para todos os tipos de recurso.

A seguinte notação também foi usada na formalização do problema:

- H: quantidade de máquinas físicas pertencentes ao provedor. O índice h é usado para representar uma máquina da nuvem ($1 \leq h \leq H,\ h \in \mathbb{N}$).

- R: quantidade de classes oferecidas pelo provedor. O índice r é usado para representar uma classe da nuvem ($1 \leq r \leq R,\ r \in \mathbb{N}$).

- $W_r = \{V_{1r}, \cdots, V_{jr}, \cdots, V_{|W_r|r}\}$: conjunto da carga de trabalho para a classe r durante o período de observação, onde V_{jr} representa a j-ésima requisição de VM para a classe r.

- $b(V_{jr})$: registro de tempo em que a j-ésima requisição de VM da classe r foi subme-

tida.

- $e(V_{jr})$: registro de tempo em que a execução da j-ésima requisição de VM da classe r foi finalizada.

- D_{jr}: capacidade requisitada pela j-ésima requisição de VM da classe r (i.e., o tamanho da VM), em *CPU-cores*.

- S_{jr}: tempo de serviço da j-ésima requisição de VM da classe r, em unidades de tempo.

- C_h: capacidade da máquina h, em *CPU-cores*. A capacidade nominal da nuvem, agregada para todas as máquinas, é definida por $\mathcal{C} = \sum_{h=1}^{H} C_h$.

- $A_{ih} \in \{0, 1\}$: disponibilidade da máquina h durante a época E_i, que indica se a máquina esteve disponível (valor 1) ou indisponível (valor 0) durante esta época. A disponibilidade média da máquina h é definida por $A_h = \sum_{i=1}^{N} A_{ih} \cdot \Delta_i / \Delta$.

- $x_{jr} \in \{0, 1\}$: admissão da j-ésima requisição de VM da classe r, que indica se a requisição foi admitida (valor 1) ou rejeitada (valor 0) no controle de admissão.

- $y_{ijrh} \in \{0, 1\}$: alocação da j-ésima requisição de VM da classe r na máquina h durante a época E_i, que indica se a requisição foi alocada (valor 1) ou não (valor 0) na máquina h durante a época E_i.

- g_{jr}: receita obtida por unidade de tempo com a execução da j-ésima requisição de VM da classe r.

- X_{jr}: penalidade imposta ao provedor por violar o SLO de disponibilidade de VM da j-ésima requisição de VM da classe r.

- γ_{jr}: disponibilidade de VM observada para a j-ésima requisição de VM da classe r. O SLO de disponibilidade de VM (i.e., o menor valor aceito) para a classe r é representado por γ_r^{min}.

- θ_r: taxa de admissão de VMs observada para a classe r. O SLO de admissibilidade (i.e., o menor valor aceito) para a classe r é representado por θ_r^{min}.

As decisões de gerência de recursos da nuvem são formalizadas como um problema de otimização. O seu objetivo é maximizar a receita P do provedor, sujeito ao cumprimento dos SLOs de disponibilidade de VMs para todas as classes.

As variáveis de decisão para cada fase do processo de gerência de recursos são as seguintes:

- **Controle de admissão**: variáveis binárias de admissão x_{jr} para cada requisição j e cada classe r.
- **Escalonamento**: variáveis binárias de alocação y_{ijrh} para cada requisição j da classe r na máquina h durante a época E_i.

Um **cenário de decisão** representa o conjunto de valores de entrada usados em uma solução para um certo período de observação, definido como

$$\Psi = \langle N, R, H, C_h, A_{ih}, W_r, D_{jr}, S_{jr}, \gamma_r^{min}, \theta_r^{min} \rangle.$$

Note que as variáveis referentes ao planejamento de capacidade são consideradas neste modelo como parâmetros de entrada, que são: a quantidade H de máquinas disponíveis na nuvem, a capacidade C_h de cada máquina e a disponibilidade de cada máquina A_{ih}. No Capítulo 7 essas variáveis passam a ser variáveis de decisão no modelo que também inclui decisões de planejamento de capacidade.

Desta forma, o problema de otimização para a gerência da nuvem é formulado como:

Dado Ψ,

maximizar

$$P = \sum_{r=1}^{R} \sum_{j=1}^{|W_r|} \left(\sum_{h=1}^{H} \sum_{i=1}^{N} x_{jr} \cdot y_{ijrh} \cdot g_{jr} \cdot \Delta_i - X_{jr} \right), \tag{6.1}$$

sujeito a

$$\sum_{r=1}^{R} \sum_{j=1}^{|W_r|} y_{ijrh} \cdot D_{jr} \leq C_h \cdot A_{ih}, \quad \forall i, h \tag{6.2}$$

$$y_{ijrh} \leq x_{jr}, \quad \forall i, j, r, h \tag{6.3}$$

$$\gamma_{jr} \geq \gamma_r^{min}, \ \ \forall j, r \tag{6.4}$$

A função objetivo (Equação 6.1) a ser maximizada representa a receita P obtida pelo provedor da nuvem, que é a soma da receita obtida ao admitir requisições e executar VMs menos as penalidades impostas pela violação de SLOs de disponibilidade de VM. A restrição de capacidade (Equação 6.2) estabelece que a capacidade alocada para todas as VMs em uma máquina não pode ser maior que a capacidade da máquina, e que máquinas indisponíveis para o provedor não devem possuir VMs alocadas. A restrição de admissão (Equação 6.3) define que uma VM só pode ser alocada se e somente se sua requisição for admitida na fase de controle de admissão. Finalmente, a restrição de SLA estabelece que as disponibilidades das VMs (Equação 6.4) não podem ser menores que o valor mínimo aceitável (i.e., o SLO) definido para cada classe.

Infelizmente, este problema de otimização é um exemplo de programação linear inteira conhecido por ser NP-completo. Além disso, para solucionar esta otimização seria necessário conhecimento exato de dados futuros da demanda e oferta de recursos da nuvem, que é algo não realista para ser assumido. Desta forma, nesta tese são propostos modelos que buscam resolver os problemas de controle de admissão e planejamento de capacidade em provedores de nuvem de IaaS com base em premissas realistas, baseado em dados históricos observados no sistema.

6.3 Caracterização da carga de trabalho e capacidade da nuvem

Para entender o ambiente de nuvem computacional em estudo e elaborar soluções adequadas para ele, analisou-se nesta seção os rastros de carga de trabalho de uma nuvem produção. Para isto, foram usados dados públicos de carga de trabalho de um *cluster* da Google [65]. Esses rastros possuem dados de um cluster com 12.583 máquinas físicas e mais de 25 milhões de tarefas submetidas durante 29 dias em maio de 2011. Estes rastros também foram usados na avaliação de métodos propostos nesta tese. Outras análises destes dados podem ser encontradas na literatura [64; 14].

Os valores da capacidade disponível na nuvem e da capacidade requisitada pelas tarefas submetidas ao sistema são normalizadas pelo tamanho da maior máquina disponível no *cluster*. Ou seja, uma unidade de CPU normalizada é equivalente ao valor da capacidade de CPU

da maior máquina física do *cluster* da Google. Este valor específico não foi divulgado, mas pode-se fazer alguma estimativa com base nos dados normalizados.

A Figura 6.2 mostra o histograma da capacidade das máquinas físicas da nuvem, contendo no eixo-x a capacidade de CPU normalizada da máquina e no eixo-y a quantidade de máquinas da nuvem que possuem capacidade x. Observa-se que de um total de 12.583 máquinas, a grande maioria (93%) possui capacidade de CPU normalizada de $0{,}5$, enquanto 6% possui a maior capacidade (igual a 1) e apenas 1% possui a menor capacidade que é equivalente a $0{,}25$ da maior máquina da nuvem.

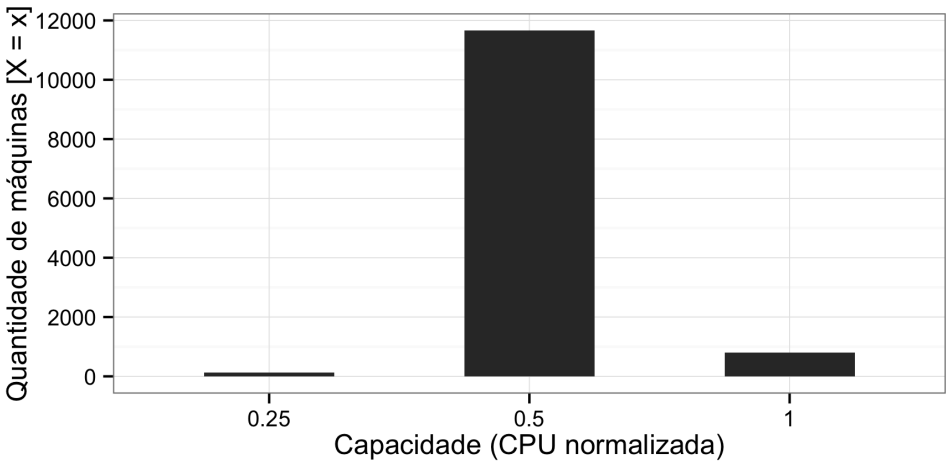

Figura 6.2: Histograma da capacidade das máquinas físicas, medidas em CPU normalizada.

Apesar de não ser necessário para as avaliações deste trabalho, pode-se estimar o tamanho da maior máquina da nuvem com base em configurações de *clusters* da época da coleta dos rastros, feita em 2011. Uma estimativa razoável seria uma capacidade 64 CPU-cores para as maiores máquinas, tendo então a grande maioria das máquinas com 32 CPU-cores e outras poucas máquinas com 16 CPU-cores. Porém, nas análises e avaliações deste trabalho serão usados valores normalizados de capacidade disponível e requisitada, da forma que está apresentada nos rastros.

Baseado na descrição dos rastros em [65], foram definidas três classes de tarefas de acordo com as suas prioridades: classe "prod" para tarefas de produção com alta prioridade ($2 \leq prioridade \leq 8$); classe "batch" para tarefas com prioridade intermediária ($9 \leq prioridade \leq 11$); e classe "free" para tarefas com baixa prioridade ($0 \leq priority \leq 1$).

A Figura 6.3 mostra a função distribuição acumulada (FDA) empírica da capacidade

requisitada por cada tarefa para as diferentes classes, medida em CPU normalizada. Na avaliação deste trabalho, a capacidade requisitada por tarefa é equivalente ao tamanho das VMs. A classe de maior prioridade (*prod*) possui os maiores tamanho de VMs requisitadas com uma mediana de $0{,}0625$, seguida da classe intermediária (*batch*) com mediana de $0{,}03125$ e, com os menores tamanho de VMs, a classe de menor prioridade (*free*) com mediana $0{,}01250$ para a CPU normalizada. O tamanho de VM máximo foi de $0{,}5$ em CPU normalizada. Considerando por exemplo que a maior máquina da nuvem possui 64 CPU-cores, as medianas da capacidade requisitada por tarefa para as classes *prod*, *batch* e *free* seriam de 4, 2 e $0{,}8$ CPU-cores, respectivamente, enquanto a capacidade máxima requisitada seria de 32 CPU-cores.

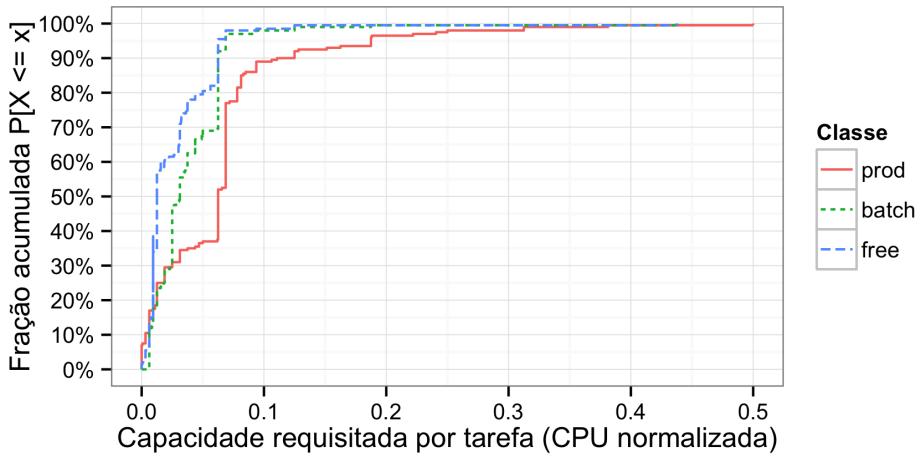

Figura 6.3: Função distribuição acumulada (FDA) empírica da capacidade requisitada por cada tarefa para as diferentes classes.

No sistema da Google onde os rastros foram coletados, os usuários submetem *jobs* agrupados como um conjunto de tarefas [74]. As tarefas de um mesmo job são submetidas no mesmo instante de tempo e geralmente requisitam a mesma capacidade de recursos, mas podem ser alocadas em máquinas diferentes. A Figura 6.4 mostra a FDA empírica da quantidade de tarefas por job para as diferentes classes, com o eixo-x em escala logarítmica. A grande maioria dos jobs submetidos em todas as classes possuem apenas uma tarefa. Aproximadamente 87% dos jobs das classes *prod* e *free* possuem apenas uma tarefa, enquanto a classe *batch* tende a ter uma maior quantidade de jobs por tarefa possuindo 65% dos jobs com uma única tarefa e uma quantidade significativa de jobs com dezenas e centenas de tarefas. Uma possível causa dessa maior quantidade de tarefas por job é uma maior presença de

aplicações do tipo *batch* para esta classe, como aplicações *MapReduce* que possuem diversas tarefas executando em paralelo.

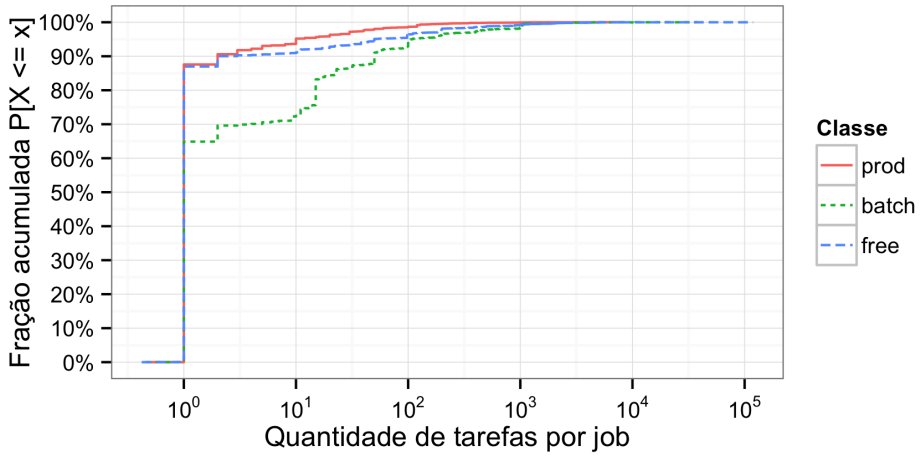

Figura 6.4: Função distribuição acumulada (FDA) empírica da quantidade de tarefas por job para as diferentes classes.

A Figura 6.5 mostra a FDA empírica do tempo de execução de tarefas em minutos para as diferentes classes, com o eixo-x em escala logarítmica. As tarefas da classe de maior prioridade (*prod*) possuem os maiores tempos de execução com uma mediana de 722 minutos (ou $\approx$ 30 horas), seguida da classe *free* com mediana de 9 minutos e, com os menores tempos médios de execução, a classe *batch*) com mediana de 8 minutos. Observa-se também que uma fração significativa de tarefas da classe *prod* ($\approx 15\%$) possuem o valor máximo do tempo de execução, que é limitado pelo tamanho do rastro e representa as tarefas que executaram durante todos os seus 29 dias de abrangência. Um possível motivo dos grandes valores de tempo de execução da classe *prod* é que ela deve concentrar a maior parte dos serviços de longa duração, como servidores web interativos ou serviços que executam permanentemente para gerenciar sistemas, e que geralmente possuem alta prioridade. Por outro lado, as classes *batch* e *free* provavelmente possuem mais aplicações não interativas e que não necessitam de um longo tempo de execução para realizar sua computação.

A Figura 6.6 mostra a FDA empírica do tempo entre chegadas de tarefas para as diferentes classes, com o eixo-x em escala logarítmica. Observa-se que mais da metade das tarefas da classe *batch* chegaram no mesmo instante da tarefa anterior (i.e., tempo entre chegadas igual a zero). Isto acontece por causa da quantidade significativa de jobs com múltiplas ta-

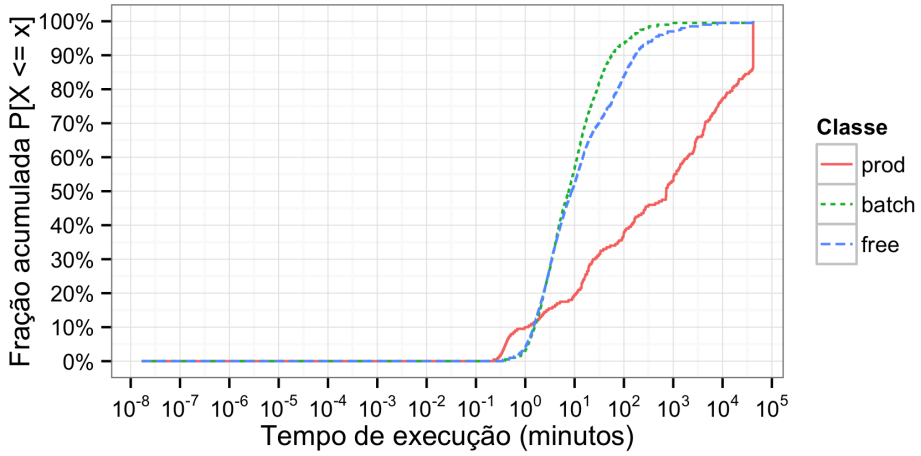

Figura 6.5: Função distribuição acumulada (FDA) empírica do tempo de execução de tarefas para as diferentes classes.

refas, como visto na análise anterior. Apesar das classes *prod* e *free* não possuirem muitos intervalos entre chegadas de tarefas exatamente iguais a zero, elas possuem uma quantidade de valores próximos a zero, que representa uma grande quantidade de jobs submetidos em rajadas (ou *bursts*). A classe *prod* tende a ter os maiores valores de intervalo entre chegadas, com diferenças de segundos ou até minutos (14 minutos no máximo) entre a chegadas consecutivas de tarefas.

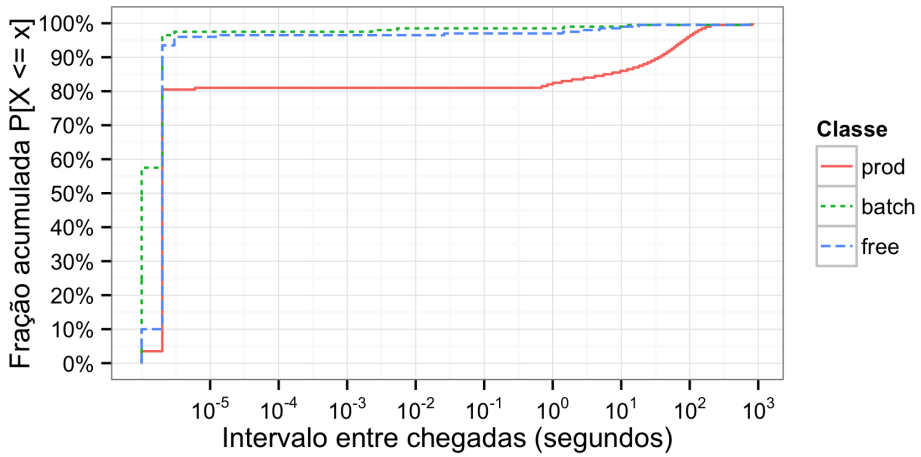

Figura 6.6: Função distribuição acumulada (FDA) empírica do tempo entre chegadas de tarefas para as diferentes classes.

A Figura 6.7 mostra a taxa de chegada de requisições ao longo do tempo para diferentes

classes, agregando a quantidade de CPU normalizada requisitada por todas as tarefas de uma classe submetidas em intervalos de 1 hora. Note que as escalas no eixo-y são diferentes para as diferentes classes. A classe *batch* possui as maiores taxas de chegada, tendo como máximo $3017{,}2$ e mediana $766{,}2$ de CPU normalizada requisitada por tarefas que chegaram durante uma hora; a classe *free* teve máximo $2092{,}8$ e mediana $237{,}8$; e a classe *prod* teve máximo $547{,}8$ e mediana $2{,}5$. Apesar da classe *prod* possuir uma menor taxa de chegada com relação às outras classes, ela tem uma quantidade muito grande de requisições que estavam executando desde antes do tempo de início da coleta dos rastros, que não foram contabilizadas nas estatísticas de taxa de chegada, mas somam um valor total de $4835{,}7$ de CPU normalizada no início do rastro. Grande parte dessas requisições *prod* que foram submetidas antes da coleta do rastro executam por longos períodos de tempo, tornando esta a classe que mais requisita recursos da nuvem ao longo do tempo.

Para analisar a demanda total que cada classe gera para a nuvem ao longo do tempo, foi realizada a simulação do ambiente de nuvem considerando uma capacidade de recursos infinita. Ou seja, cada tarefa submetida será alocada sem espera em fila e executará até completar seu tempo de execução. A capacidade requisitada por cada tarefa e seu tempo de execução foram obtidos dos rastros. A Figura 6.8 mostra a capacidade total de CPU normalizada requisitada para as diferentes classes ao longo do tempo. Os valores para as diferentes classes estão empilhados, de tal forma que o valor acumulado das classes representa a demanda total da nuvem. Nota-se que a fração de capacidade requisitada pela classe de maior prioridade (*prod*) é significativamente maior que a das outras classes. A demanda da classe *prod* também possui uma variação muito baixa, enquanto as classes *batch* e *free* possuem uma alta variância – isto também pôde ser constatado na análise das taxas de chegada. A demanda total, agregando todas as classes, teve um valor médio de CPU normalizada de $\approx 7214{,}8$, enquanto o seu valor máximo foi de $\approx 8462{,}2$. Ou seja, seriam necessárias mais de 8462 máquinas de maior tamanho de CPU contidas no *cluster* da Google para se planejar para a demanda de pico e acomodar toda a carga sem rejeitar requisições, sem tempo de espera e sem sobrecarregar o sistema (e.g., com *overbooking*). A média da capacidade total requisitada pelas classes *prod*, *batch* e *free* foi respectivamente $4995{,}7$, $1045{,}4$ e $1172{,}7$.

Porém, a super-provisão de recursos não é desejada por provedores de nuvem, pois resultam em um grande custo para atender uma demanda de pico esporádica. Por exemplo, a

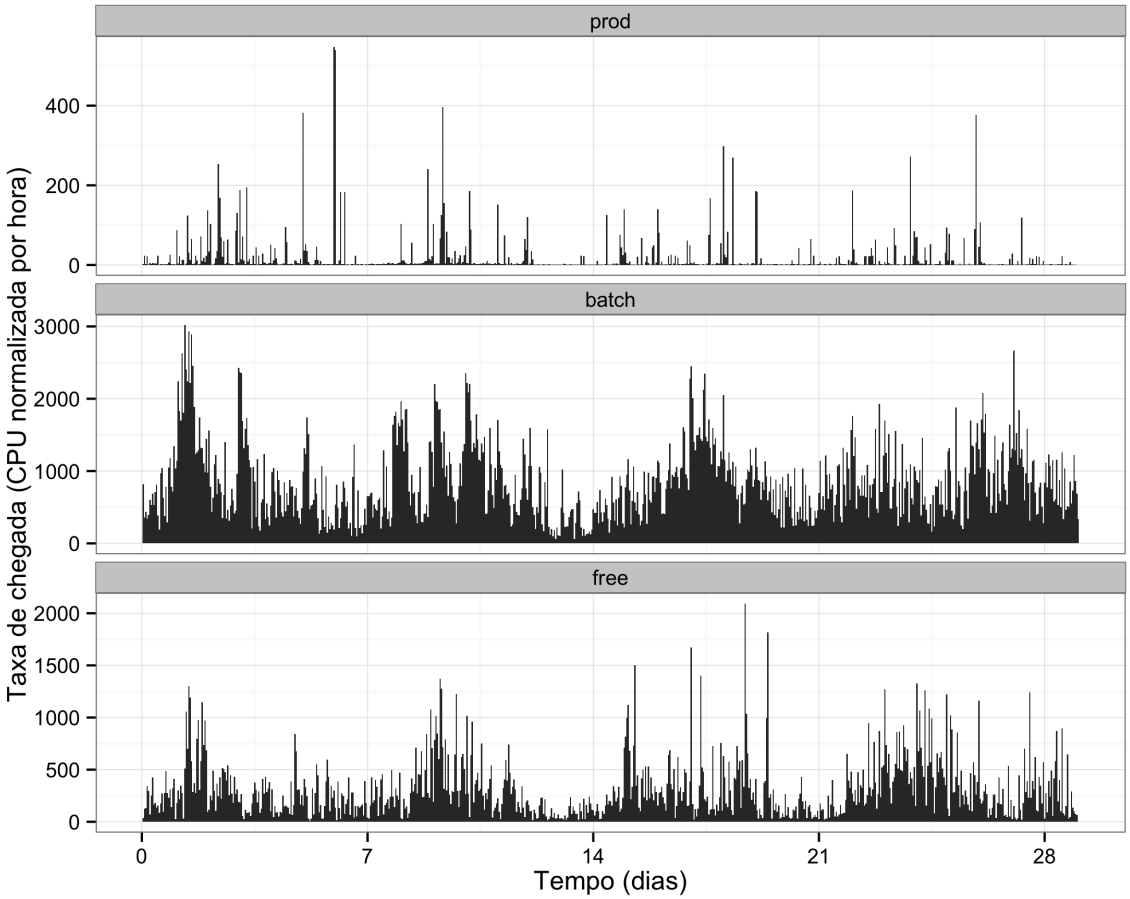

Figura 6.7: Taxa de chegada de requisições ao longo do tempo para diferentes classes, em total de CPU normalizada requisitada por tarefas submetidas em janelas de 1 hora.

capacidade do *cluster* da Google analisado nesta seção não é suficiente para acomodar toda a demanda.

A Figura 6.9 mostra a capacidade total do cluster ao longo do tempo para os rastros analisados. Os valores são calculados agregando a capacidade de todas as máquinas disponíveis em cada intervalo de tempo – o valor varia ao longo do tempo por causa de eventos como falhas e manutenção de máquinas. A soma da capacidade de todas as máquinas registradas nos rastros somam uma capacidade total de CPU normalizada de 6659. Porém, em nenhum momento todas as máquinas estiveram disponíveis: o valor máximo da capacidade total disponível foi de $6619{,}2$ e o valor médio foi de $6584{,}6$, o que representa uma disponibilidade média de CPU de $98{,}88\%$ do total do *cluster*. Note que na última semana do rastro há uma

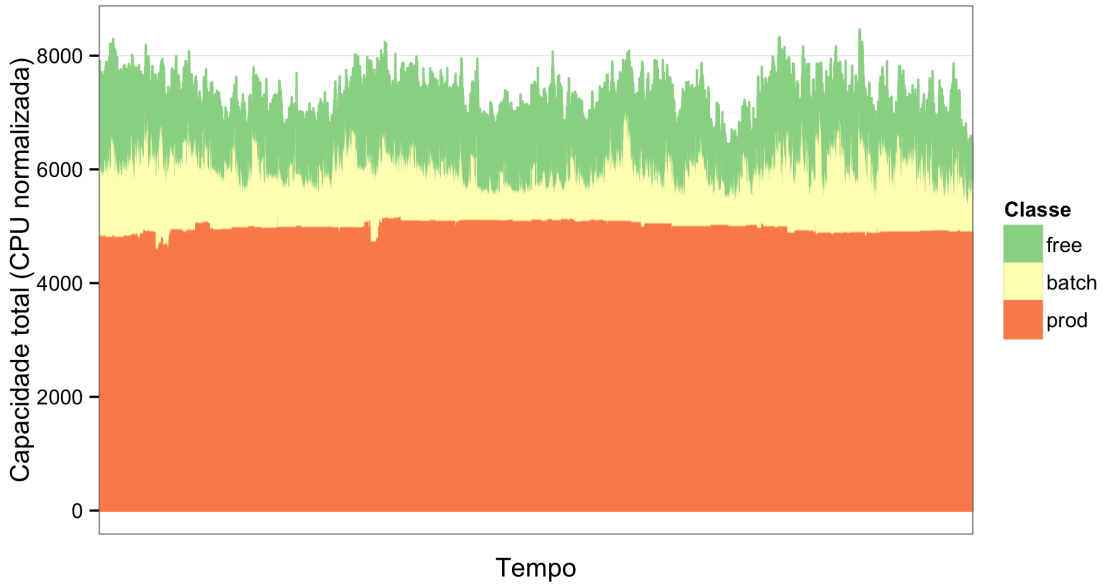

Figura 6.8: Capacidade requisitada por tarefas ao longo do tempo para diferentes classes, em total de CPU normalizada requisitada em janelas de 5 minutos.

grande decaída na capacidade disponível, atingindo um valor mínimo de $6498{,}2$. Um provável motivo para este decaimento seria uma indisponibilidade simultânea de máquinas que compartilham o mesmo *rack*, com um ponto comum de falhas como um roteador ou fonte de energia compartilhado, como também a atualização do sistema para uma parte das máquinas *cluster*.

Ao comparar os dados de demanda de requisições das diferentes classes na Figura 6.8 e da capacidade total do cluster na Figura 6.9, observa-se que a capacidade total não é suficiente para acomodar toda a demanda. Nesta tese busca-se gerenciar nuvens de IaaS de forma eficiente, tendo em vista a variação da demanda e da capacidade disponível da nuvem ao longo do tempo, além dos diferentes SLOs definidos para as diferentes classes oferecidas. Na próxima seção, é apresentado um modelo de controle de admissão baseado em predição que é parte da gerência de recursos da nuvem proposta nesta tese. Em seguida, o modelo é avaliado usando os rastros de carga de trabalho analisados nesta seção, realizando simulações do ambiente de nuvem em diferentes cenários.

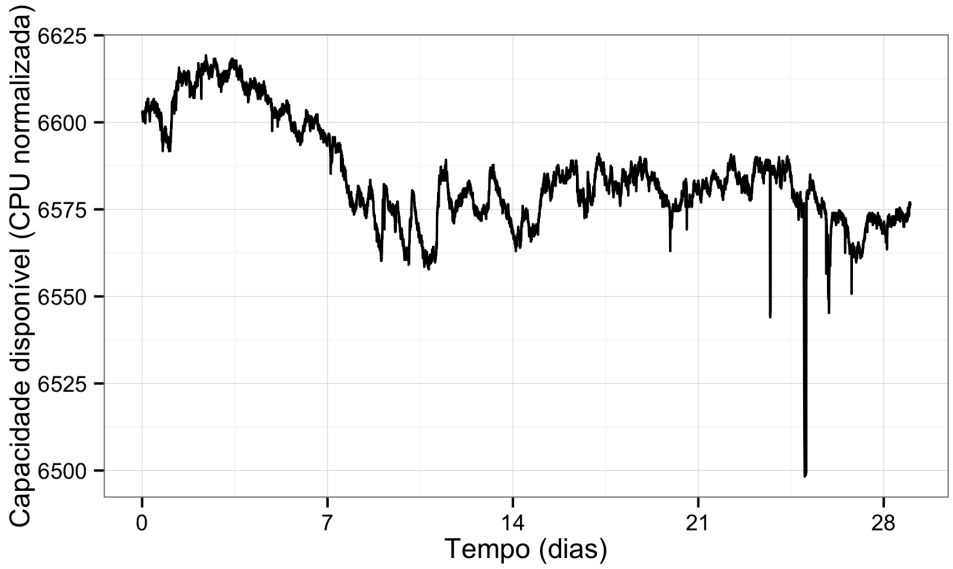

Figura 6.9: Capacidade disponível no cluster (em CPU normalizada) ao longo do tempo, agregando a capacidade das as máquinas disponíveis em cada intervalo de tempo.

6.4 Modelo do controle de admissão

Nesta seção é apresentado o modelo de controle de admissão proposto neste trabalho. O modelo se baseia no sistema de nuvem computacional de IaaS com múltiplas classes, descrito no Capítulo 4. A Tabela 6.1 descreve os principais símbolos usados no modelo de controle de admissão.

Uma abordagem baseada em quotas é adotada, na qual o provedor define dinamicamente uma *quota* para limitar a quantidade de recursos que podem ser alocados para uma classe de serviço. Novas requisições são rejeitadas se a capacidade total requisitada por VMs de uma classe exceder a quota associada à sua classe. Estratégias similares incluem o sistema de gerência de recursos da Google chamado Borg [74] e os limites impostos pelo provedor de nuvem Amazon EC2 para a quantidade de VMs simultâneas que podem ser obtidas por cada usuário [3].

Prioridades são associadas às diferentes classes oferecidas pelo provedor da nuvem, onde $r = 1$ identifica a classe de maior prioridade e $r = R$ a classe de menor prioridade, onde R representa a quantidade de classes oferecidas. Um provedor pode definir explicitamente a ordem de prioridade para as classes, ou elas podem ser inferidas de acordo com os SLOs definidos para cada uma delas. O modelo de controle de admissão proposto neste traba-

Símbolo	**Descrição**
N	quantidade de épocas em um período de observação.
R	quantidade de classes oferecidas pelo provedor.
H	quantidade de máquinas físicas pertencentes ao provedor.
$W_r = \{V_{1r}, \cdots, V_{jr}, \cdots, V_{\|W_r\|r}\}$	conjunto da carga de trabalho para a classe r durante o período de observação, onde V_{jr} representa a j-ésima requisição de VM para a classe r.
A_{ih}	disponibilidade da máquina h durante a época E_i, que indica se a máquina esteve disponível (valor 1) ou indisponível (valor 0) durante esta época.
C_h	capacidade da máquina h, em *CPU-cores*.
D_{jk}	capacidade requisitada pela j-ésima requisição de VM da classe r (i.e., o tamanho da VM), em *CPU-cores*.
x_{jk}	admissão da j-ésima requisição de VM da classe r, que indica se a requisição foi admitida (valor 1) ou rejeitada (valor 0) no controle de admissão.
y_{jkh}	alocação da j-ésima requisição de VM da classe r na máquina h durante a época E_i, que indica se a requisição foi alocada (valor 1) ou não (valor 0) na máquina h durante a época E_i.
c_{ir}	capacidade disponível para a classe r na época E_i
γ_{jr}	disponibilidade de VM observada para a j-ésima requisição de VM da classe r. O SLO de disponibilidade de VM para a classe r é representado por γ_r^{min}.

Tabela 6.1: Principais símbolos usados no modelo de controle de admissão.

lho considera um escalonador preemptivo baseado em prioridades, que aloca primeiramente VMs de maior prioridade, podendo preemptar (i.e., desalocar) VMs de menor prioridade caso necessário. O sistema Borg da Google adota uma política de escalonamento similar [74].

As quotas são definidas dinamicamente para cada classe de acordo com a capacidade da nuvem disponível para a classe e seu SLO referente à métrica de disponibilidade de VMs. A capacidade disponível para uma classe depende não só da capacidade total da nuvem disponível ao longo do tempo, mas também da demanda por recursos de classes de maior prioridade. A classe de maior prioridade é a única que depende apenas da disponibilidade e tamanho das máquinas físicas do provedor da nuvem, não sendo afetada pela demanda de outras classes.

Desta forma, a *capacidade disponível* para a classe r na época E_i é definida como

$$c_{ir} = \sum_{h=1}^{H} A_{ih} \cdot C_h - \sum_{k=1}^{r-1} \sum_{j=1}^{|W_k|} x_{jk} \cdot y_{ijkh} \cdot D_{jk}. \tag{6.5}$$

Com base na Lei de Little [58], pode-se definir a quantidade média de requisições de VM no sistema como

$$L = W \cdot T = \frac{S}{\gamma} \cdot \frac{c}{S} = \frac{c}{\gamma}, \tag{6.6}$$

onde W é o tempo médio de residência de uma VM (i.e., tempo de resposta), T é a vazão média (i.e., quantidade de VMs finalizadas por unidade de tempo), S é o tempo de serviço médio, γ é a média da disponibilidade de VMs e c é a quantidade média de servidores disponíveis no sistema (i.e., a capacidade total disponível).

A Equação 6.6 indica que um maior número L de requisições admitidas implica em uma menor disponibilidade de VM γ para uma mesma capacidade c. Neste modelo, considera-se L como um bom estimador do número máximo de requisições a serem admitidas para cada classe (i.e., a quota), tendo como alvo uma disponibilidade de VMs γ e usando como base uma estimativa da capacidade disponível para a classe c no futuro.

Desta forma, basedo na Equação 6.6, a *quota* para a classe r na época i é definida como

$$K_{ir} = \frac{\hat{c}_{ir}}{\gamma_r^{min}}, \tag{6.7}$$

onde $\hat{c}_{ir}$ é a capacidade disponível para a classe r estimada na época E_i para épocas futuras, e γ_r^{min} é o SLO referente à disponibilidade de VMs para a classe r.

Como a capacidade disponível para cada classe pode variar ao longo do tempo, foram usados métodos de predição para capturar suas variações em tempos futuros. Predições superestimadas da capacidade disponível para classes podem causar violações de SLO de disponibilidade de VM. Para lidar com isto, propõe-se o cálculo de intervalos de confiança para as predições e o uso de estimativas conservadoras utilizando os limites inferiores dos intervalos de confiança das predições como estimativas. Esta mesma abordagem foi usada no planejamento da classe *Economy* no Capítulo 5.

Seja $\mathcal{F}(\vec{c}_{ir}, w)$ uma função de predição de séries temporais, que recebe como entrada dados históricos $\vec{c}_{ir}$ da capacidade disponível para a classe r, observados até o início da época E_i, para prever valores futuros em uma janela de w unidades de tempo à frente. Desta forma, no início da época E_i estima-se a capacidade disponível para a classe r como o limite inferior do intervalo de confiança da predição com variância desconhecida [76], definido como

$$\hat{c}_{ir} = \mathcal{F}(\vec{c}_{ir}, w) - t_{q,n-1} \cdot \sigma \cdot \sqrt{1 + \frac{1}{n}}, \tag{6.8}$$

onde $t_{q,n-1}$ é o q-quantil da distribuição t-Student com $n-1$ graus de liberdade; q é o nível de confiança da predição; σ é o desvio padrão e n o tamanho da amostra dos dados de entrada.

Adotando este modelo de controle de admissão baseado em predições, um provedor de nuvem pode definir quotas ao longo do tempo para cada classe. Qualquer método de predição (ou uma combinação de métodos [62]) pode ser usado. Na seção de avaliação, o modelo é instanciado com diferentes técnicas de predição.

6.5 Metodologia de avaliação

Nesta seção é descrito como o modelo de controle de admissão é instanciado usando diferentes heurísticas, as métricas usadas na avaliação, o procedimento de simulação da gerência de recursos da nuvem e os cenários explorados nas simulações baseadas em rastros de sistemas de nuvem em produção.

6.5.1 Heurísticas de controle de admissão

O modelo de controle de admissão é instanciado com duas heurísticas preditivas e uma gulosa. Elas também são comparadas a uma outra heurística que não usa controle de admissão. As heurísticas usadas na avaliação são as seguintes:

- *pred-cmean*: heurística preditiva que usa o modelo de controle de admissão para definir uma quota a cada classe usando o método de predição *Conservative Mean*.
- *pred-ets*: heurística preditiva que usa o modelo de controle de admissão para definir uma quota a cada classe usando o método de predição *Exponential Smoothing (ETS)*.
- *greedy-quota*: heurística gulosa que usa o modelo de controle de admissão para definir uma quota a cada classe com base no último valor observado da capacidade disponível para a classe.
- *no-adm-control*: heurística simples que não usa controle de admissão, admitindo todos as requisições para todas as classes.

Os métodos de predição usados pelas heurísticas *pred-cmean* e *pred-ets* são:

- *Conservative Mean (CMEAN)*: neste método, calcula-se médias aritméticas para três amostras dos dados de entrada: a *média da última hora* usa dados da hora anterior

à predição para capturar mudanças de curto prazo; a *média do último dia* usa dados das 24 horas anteriores à predição para capturar mudanças de médio prazo; e a *média diária* usa valores históricos da mesma hora do dia da predição (e.g., às 9 da manhã) em dias anteriores, para capturar padrões diários de sazonalidade nos dados, no longo prazo. O valor mínimo das três médias é usado para se obter uma predição conservadora. Desta forma, esta heurística reage rapidamente a quedas repentinas de capacidade disponível, mas aumenta lentamente as predições para picos de capacidade disponível observados.

- *Exponential Smoothing (ETS)*: método de predição que captura tendências de crescimento/decrescimento nos dados, além de padrões de sazonalidade. A descrição deste método pode ser encontrado na Seção 5.5, onde ele também foi usado em avaliação.

Um nível de confiança de $q = 95\%$ foi usado para calcular intervalos de confiança para as predições. Este valor pode ser ajustado pelos provedores da nuvem para obter predições mais (ou menos) conservadoras [23]. Os dados históricos usados como entrada se acumulam com o tempo, tornando o tamanho da amostra n maior ao longo da simulação. Isto não causou problema de desempenho nas avaliações realizadas neste estudo para um período de 1 mês. Porém, para período longos seria mais adequado adotar uma abordagem de janela deslizante para os dados de entrada para evitar problemas de desempenho no processamento das predições.

No início de um período de decisão, o tamanho da amostra para as primeiras predições será muito pequeno, o que pode afetar os resultados das heurísticas preditivas. Para tratar deste problema, adotou-se uma abordagem conservadora durante a primeira hora simulada, definindo a quota para uma classe como a capacidade disponível para ela no momento da predição. Isto significa que nenhuma requisição de VM ficará pendente no momento da sua admissão, porém elas poderão ser interrompidas futuramente devido a falhas de máquinas ou preempções causadas por requisições futuras de VMs para classes com maior prioridade.

6.5.2 Ambiente de simulação

A avaliação foi feita através de simulações baseadas em rastros de sistemas de nuvem, que foram apresentados e analisados na Seção 6.3.

Como dito, os dados referentes à capacidade de recursos (CPU e memória) nos rastros são normalizados pelo tamanho da maior máquina disponível no cluster. Infelizmente, o tamanho dessa máquina específica não foi publicado, não permitindo o cálculo de valores absolutos de capacidade de recursos. Porém, a capacidade de recursos normalizada é suficiente para nossas simulações, pois tanto a capacidade das máquinas como a capacidade requisitada pelas tarefas são normalizadas pelo mesmo fator constante. Com esses dados é viável calcular métricas como a utilização da nuvem e com isto tomar decisões de controle de admissão, planejamento de capacidade e escalonamento.

Cada tarefa submetida nos rastros é considerada como uma requisição de VM na simulação. Atributos do rastro são mapeados para parâmetros de entrada da simulação da seguinte forma:

- *Tempo de submissão de requisições de VM* ($b(V_{jr})$): tempo de submissão de cada tarefa.

- *Classe da requisição de VM* (r): faixa de prioridade associada a cada tarefa, baseado na descrição dos rastros [65]. Como descrito na Seção 6.3, três prioridades ($R = 3$) são associadas às tarefas: "prod" para tarefas de produção com alta prioridade ($2 \leq prioridade \leq 8$); "batch" para tarefas com prioridade intermediária ($9 \leq prioridade \leq 11$); e "free" para tarefas com baixa prioridade ($0 \leq priority \leq 1$).

- *Tamanho da VM* (S_{jr}): capacidade de CPU requisitada por cada tarefa, normalizada pela capacidade da maior máquina do cluster. Para tarefas que mudam a capacidade requisitada ao longo do tempo nos rastros, considerou-se o valor máximo requisitado durante todo o tempo que a tarefa esteve no sistema.

- *Demanda de serviço da VM* (D_{jr}): tempo total que cada tarefa esteve no estado "executando" durante todo o rastro.

- *Capacidade da máquina* (C_h): capacidade de CPU de cada máquina física do cluster contida no rastro em CPU-cores, normalizada pela capacidade da maior máquina do cluster.

- *Disponibilidade da máquina* (A_{ih}): períodos observados no rastro nos quais as máquinas estiveram disponíveis.

Como os eventos de submissão de tarefas e disponibilidade de máquinas ocorrem com muita frequência nesse rastro, os eventos foram agregados em intervalos fixos de 5 minutos. Isto foi feito para reduzir o tempo de simulação e tornar factível a avaliação do rastro completo em diferentes cenários. Portanto, decisões de controle de admissão e escalonamento são realizadas a cada 5 minutos nas simulações, fazendo com que cada época E_i tenha o mesmo tamanho (i.e., $\Delta_i = 5\ minutos, \forall\ i$). Se uma máquina ficar indisponível durante uma dessas janelas, considera-se que esta máquina esteve indisponível durante toda a época. Por outro lado, eventos de submissão de requisições de VMs que ocorrem durante uma época são antecipados para o início da mesma época, enquanto eventos de finalização de requisições de VM são postergados para o início da época seguinte.

A cada nova época, o mecanismo de controle de admissão pode ajustar as quotas definidas para as classes e rejeitar novas requisições se a quota da sua classe for excedida. Como o foco desse estudo é o controle de admissão, alocações de VMs em máquinas físicas específicas (i.e., *VM placement*) não foram feitas nas simulações. O escalonador define as requisições de VM a serem alocadas de acordo as suas prioridades, de tal modo que a capacidade total alocada para VMs não seja maior que a capacidade disponível na nuvem. Uma política de escalonamento preemptiva baseada em prioridades foi usada nas simulações, como descrito na Seção 6.4. Requisições de VM na mesma classe de prioridade são alocadas de acordo com seus tempos de submissão (i.e., *First-Come First-Served*). Uma abordagem de *aggressive backfilling* também é usada nas simulações [37], na qual requisições que estão à frente da fila podem ser puladas se seu tamanho de VM não couber na capacidade disponível da nuvem, dando vez a requisições de VMs menores e de menor prioridade. Embora uma requisição pulada ficará pendente e indisponível, ela será reconsiderada pelo escalonador a cada nova época, sendo possível preemptar VMs de menor prioridade para que ela seja alocada.

Funções lineares simples foram usadas para calcular a receita e as penalidades obtidas pelo provedor ao oferecer as diferentes classes de serviço na nuvem. Essas funções são proporcionais à capacidade requisitada pelas VMs e ao SLO de disponibilidade das VMs para cada classe.

A receita obtida por unidade de tempo com a execução da j-ésima requisição de VM da

classe r é definida por

$$g_{jr} = S_{jr} \cdot \gamma_r^{min}. \tag{6.9}$$

onde S_{jr} é o tamanho da j-ésima VM requisitada na classe r e γ_r^{min} é o SLO de disponibilidade de VM da classe r.

As penalidades pagas pelo provedor são proporcionais ao tempo total em que as VMs estiveram executando ou pendentes. Portanto, define-se a penalidade imposta ao provedor por violar o SLO de disponibilidade de VM da j-ésima requisição da classe r como

$$X_{jr} = \begin{cases} 0, & \text{se} \quad \gamma_{jr} \geq \gamma_r^{min} \\ S_{jr} \cdot \gamma_r^{min} \cdot \left(e(V_{jr}) - b(V_{jr})\right), & \text{caso contrário.} \end{cases} \tag{6.10}$$

Desta forma, pode-se calcular a receita P obtido pelo provedor como a receita obtida para cada VM executada subtraída das penalidades causadas por violações, como apresentado na Equação 6.1.

As heurísticas de controle de admissão foram avaliadas para diferentes métricas, definidas a seguir:

- *Admission rate* – ou taxa de admissão, definida como a porcentagem de requisições admitidas para a classe r:

$$\theta_r = \frac{\sum_{j=1}^{|W_r|} x_{jr}}{|W_r|} \cdot 100\%. \tag{6.11}$$

- *VM availability* – ou disponibilidade de VM, definida como a porcentagem de tempo que uma requisição de VM V_{jr} esteve alocada (i.e., no estado *executando*) desde o seu tempo de sua submissão até o tempo em que foi finalizada:

$$\gamma_{jr} = \frac{\sum_{i=1}^{N} \sum_{h=1}^{H} y_{ijrh} \cdot \Delta_i}{e(V_{jr}) - b(V_{jr})} \cdot 100\%, \tag{6.12}$$

  onde $b(V_{jr})$ e $e(V_{jr})$ são respectivamente os tempos de submissão e finalização da j-ésima requisição da classe r. Considerou-se $b(V_{jr}) = b(E_1)$ para uma requisição V_{jr} se ela foi submetida antes do início do período de observação $b(E_1)$, e também $e(V_{jr}) = e(E_N)$ se ela foi finalizada após o fim do período de observação $e(E_N)$.

- *VM Availability SLO fulfillment* – ou cumprimento de SLO de disponibilidade de VM, definida como a porcentagem de requisições admitidas da classe r para as quais a disponibilidade de VMs foi maior ou igual ao SLO definido para esta classe:

$$\Gamma_r = \frac{|\{V_{jr} \in W_r : \gamma_{jr} \geq \gamma_r^{min} \wedge x_{jr} = 1\}|}{|\{V_{jr} \in W_r : x_{jr} = 1\}|} \cdot 100\%. \tag{6.13}$$

- *Mean cloud utilization* – ou utilização média da nuvem, definida como a porcentagem da capacidade disponível da nuvem usada na alocação de VMs, com a média calculada com base nos valores observados em cada época:

$$\mathcal{U} = \sum_{i=1}^{N} \left(\frac{\sum_{r=1}^{R} \sum_{j=1}^{|W_r|} \sum_{h=1}^{H} y_{ijrh}}{\sum_{h=1}^{H} C_h \cdot A_{ih}} \cdot \frac{\Delta_i}{\Delta} \right) \cdot 100\%. \tag{6.14}$$

- *Revenue efficiency* – ou eficiência da receita, definida como a receita obtida pela heurística m normalizada pela maior receita de todas as heurísticas para o mesmo cenário:

$$\mathcal{P}_m = \frac{P_m}{\max(P_1, \cdots, P_M)} \cdot 100\%, \tag{6.15}$$

onde M é a quantidade de heurísticas avaliadas e P_m é a a receita obtida pela heurística m, calculada com a Equação 6.1. Desta forma, a melhor heurística em um cenário terá uma *revenue efficiency* igual a 100%.

O Algoritmo 1 descreve em alto nível o procedimento de simulação da gerência de recursos da nuvem usado na avaliação. Ele recebe como entrada: a disponibilidade das máquinas ao longo do tempo; a capacidade de cada máquina; a carga de trabalho para cada classe que inclui o tempo de submissão, o tamanho da VM e o tempo de serviço de cada requisição; além da heurística m de controle de admissão a ser simulada. A simulação gera como saída as métricas seguintes: taxa de admissão e cumprimento de SLO de disponibilidade de VM para a classe r; a utilização média da nuvem; e a receita obtida pela heurística m para o cenário avaliado. Os comentários (em verde) explicam os detalhes de cada linha do algoritmo.

O *loop* nas linhas 2–25 itera para cada época do período de observação e realiza o processo de gerência de recursos da nuvem para cada época. Nas linhas 3–6, calcula-se a capacidade total disponível na nuvem, que é também a capacidade disponível para a classe de maior prioridade ($r = 1$). Nas linhas 7–13 são realizados o controle de admissão e a alocação de VMs, iniciando da classe de maior prioridade ($r = 1$) e terminando na classe

Algoritmo 1 Simula a gerência da nuvem para uma heurística de controle de admissão.

function CLOUDRMSIMULATION(A_{ih}, C_h, W_r, m)
  for $i \leftarrow 1, N$ **do** ▷ Iteração para cada época i do período de observação.
    $c_{i1} \leftarrow 0$ ▷ Inicializa a variável que representa a capacidade disponível para a classe $r = 1$
    for $h \leftarrow 1, H$ **do** ▷ Iteração para cada máquina física h do provedor.
      $c_{i1} \leftarrow c_{i1} + A_{ih} * C_h$ ▷ Adiciona a capacidade da máquina h se ela estiver disponível.
    end for
    for $r \leftarrow 1, R$ **do** ▷ Iteração para cada classe r, da classe de maior para a de menor prioridade.
      $K_{ir} \leftarrow$ DEFINEQUOTA($\vec{c}_{ir}, \gamma_r^{min}, m$) ▷ Define quota para r usando a heurística m (Eq. 6.7)
      $W_{ir}^{new} \leftarrow \{V_{jr} \in W_r : b(V_{jr}) = i\}$ ▷ Conjunto de novas requisições submetidas na época i.
      $x_{jr} \leftarrow$ DOADMISSIONCONTROL(W_{ir}^{new}, K_{ir}) ▷ Aplica controle de admissão para a classe r.
      $W_{ir} \leftarrow W_{ir} \cup \{V_{jr} \in W_{ir}^{new} : x_{jr} = 1\}$ ▷ Atualiza requisições atuais com novas admissões.
      $y_{ijrh} \leftarrow$ ALLOCATEVMS(W_{ir}, c_{ir}) ▷ Decide quais requisições serão alocadas na nuvem.
    end for
    $c_{i(r+1)} \leftarrow c_{ir}$ ▷ Inicializa a variável com a capacidade disponível para a classe seguinte.
    $W_{(i+1)r} \leftarrow W_{ir}$ ▷ Inicializa a variável com as requisições da classe r para a época seguinte.
    for each $V_{jr} \in W_{ir}$ **do** ▷ Iteração para cada requisição atual da classe r.
      if $y_{ijrh} = 1$ **then** ▷ Se a requisição j da classe r foi alocada durante a época i.
        $c_{i(r+1)} \leftarrow c_{i(r+1)} - D_{jr}$ ▷ Reduz a capacidade disponível para a classe seguinte.
        if $\sum_{i'=1}^{i} y_{i'jr} * \Delta_{i'} \geq S_{jr}$ **then** ▷ Se o tempo de serviço de V_{jr} foi satisfeito.
          $W_{(i+1)r} \leftarrow W_{(i+1)r} \setminus \{V_{jr}\}$ ▷ Remove V_{jr} das requisições da época seguinte.
          $e(V_{jr}) \leftarrow e(E_i)$ ▷ Define o tempo de finalização de V_{jr} como o fim da época i.
        end if
      end if
    end for
  end for
  for $r \leftarrow 1, R$ **do** ▷ Iteração para cada classe r.
    $\theta_r \leftarrow$ CALCULATEADMISSIONRATE(x_{jr}, W_r) ▷ Eq. 6.11
    for each $V_{jr} \in \{V_{jr} \in W_r : x_{jr} = 1\}$ **do** ▷ Iteração para cada requisição admitida da classe r.
      $\gamma_{jr} \leftarrow$ CALCULATEVMAVAILABILITY($y_{ijrh}, W_r, b(V_{jr}), e(V_{jr})$) ▷ Eq. 6.12
    end for
    $\Gamma_r \leftarrow$ CALCULATESLOFULFILLMENT(x_{jr}, W_r, γ_{jr}) ▷ Eq. 6.13
  end for
  $\mathcal{U} \leftarrow$ CALCULATECLOUDUTILIZATION($y_{ijrh}, W_r, C_h, A_{ih}$) ▷ Eq. 6.14
  $P_m \leftarrow$ CALCULATEREVENUE($x_{jr}, y_{ijrh}, g_{jr}, X_{jr}, W_r$) ▷ Eq. 6.1
  return $\langle \theta_r, \Gamma_r, \mathcal{U}, P_m \rangle$
end function

de menor prioridade ($r = R$). Para cada classe, o algoritmo define a sua quota usando uma heurística m com base no histórico da capacidade disponível para a classe e no seu SLO de disponibilidade de VM; admite ou rejeita novas requisições com base na quota definida; e decide quais VMs serão alocadas na nuvem de acordo com a ordem de chegada, o tamanho da VM e a capacidade disponível para a classe. Nas linhas 14–24, itera-se por todas as requisições atuais de VM da classe r para: atualizar a capacidade disponível para a classe seguinte de menor prioridade, subtraindo a capacidade das VMs alocadas da classe r atual; define o tempo de finalização das requisições de VM que tiveram seus tempos de serviço satisfeitos após a execução na época atual e as remove do conjunto de requisições da época seguinte. Nas linhas 26–32 são calculadas as métricas de saída específicas para cada classe r, enquanto nas linhas 33–34 são calculadas as métricas gerais da nuvem. Por fim, na linha 35 as métricas são retornadas como resultado das simulação para a heurística m e o cenário de nuvem avaliado representado pelo conjunto de valores dos parâmetros de entrada.

Diferentes cenários de nuvem são avaliados ao variar valores para os seguintes parâmetros de entrada da simulação:

- *Capacity size factor*: ou fator de capacidade, definido como um fator multiplicativo aplicado à capacidade de cada máquina observada no rastro original. Por exemplo, um *capacity size factor* igual a 1,1 significa que cada máquina terá 10% mais capacidade do que os valores originais no rastro, aumentando assim a capacidade total da nuvem pelo mesmo fator.

- *Load factor*: ou fator de carga, definido como um fator multiplicativo aplicado à capacidade requisitada por cada VM observada no rastro original. Por exemplo, um *load factor* igual a 1,1 significa que cada VM requisitada terá um tamanho 10% maior que seu valor original no rastro, aumentando assim a demanda total da nuvem pelo mesmo fator.

- *SLO strength*: ou força do SLO, definido como o nível de qualidade associado ao SLO de disponibilidade de VMs oferecido em um cenário. É representado pela combinação dos valores de SLO definidos para cada classe. Por exemplo, um *SLO strength* intermediário poderia ter valores de SLO de $\langle 100\%, 90\%, 50\% \rangle$ definidos respectivamente

para as classes *prod*, *batch* e *free*, enquanto um *SLO strength* muito alto poderia ter valores de $\langle 100\%, 99{,}9\%, 90\% \rangle$ para as mesmas classes.

6.6 Resultados

Primeiramente são apresentados os resultados para um cenário base de simulação e, em seguida, são feitas análises de sensibilidade que exploram diferentes cenários de capacidade da nuvem, fatores de carga e SLOs de disponibilidade de VMs.

6.6.1 Cenário base

Para o cenário base, foram utilizados dados de demanda e capacidade obtidos do rastro, sem modificações. A Tabela 6.2 mostra os valores usados para cada parâmetro.

Parâmetro de entrada	**Valores**
Capacity size factor	1
Load factor	1
SLO strength	$\langle prod = 100\%, batch = 90\%, free = 50\% \rangle$

Tabela 6.2: Parâmetros de entrada para o cenário base de simulação.

A Figura 6.10 mostra as métricas de *SLO fulfillment* (acima) e de *admission rate* (abaixo) para requisições de VMs de cada classe, comparando diferentes heurísticas de controle de admissão para o cenário base. O *SLO fulfillment* para a classe *prod* foi de 100% para todas as heurísticas avaliadas, pois havia capacidade suficiente para alocar todas as requisições desta classe mesmo sem realizar controle de admissão. Para a classe *batch*, as heurísticas preditivas (*pred-cmean* e *pred-ets*) obtiveram os maiores valores de *SLO fulfillment*, enquanto as heurísticas *greedy-quota* e *no-adm-ctrl* tiveram valores um pouco menores. A diferença foi mais significante para a classe *free*, para a qual as heurísticas preditivas obtiveram *SLO fulfillment* mais que o dobro dos valores da *greedy-quota*, enquanto a heurística *no-adm-ctrl* obteve um *SLO fulfillment* muito próximo a zero, violando o SLO para quase todas as requisições de VM.

A taxa de admissão para a heurística *no-adm-ctrl* é sempre 100%, pois ela nunca rejeita requisições. Para a classe *prod*, todas as heurísticas admitem todas as requisições porque há

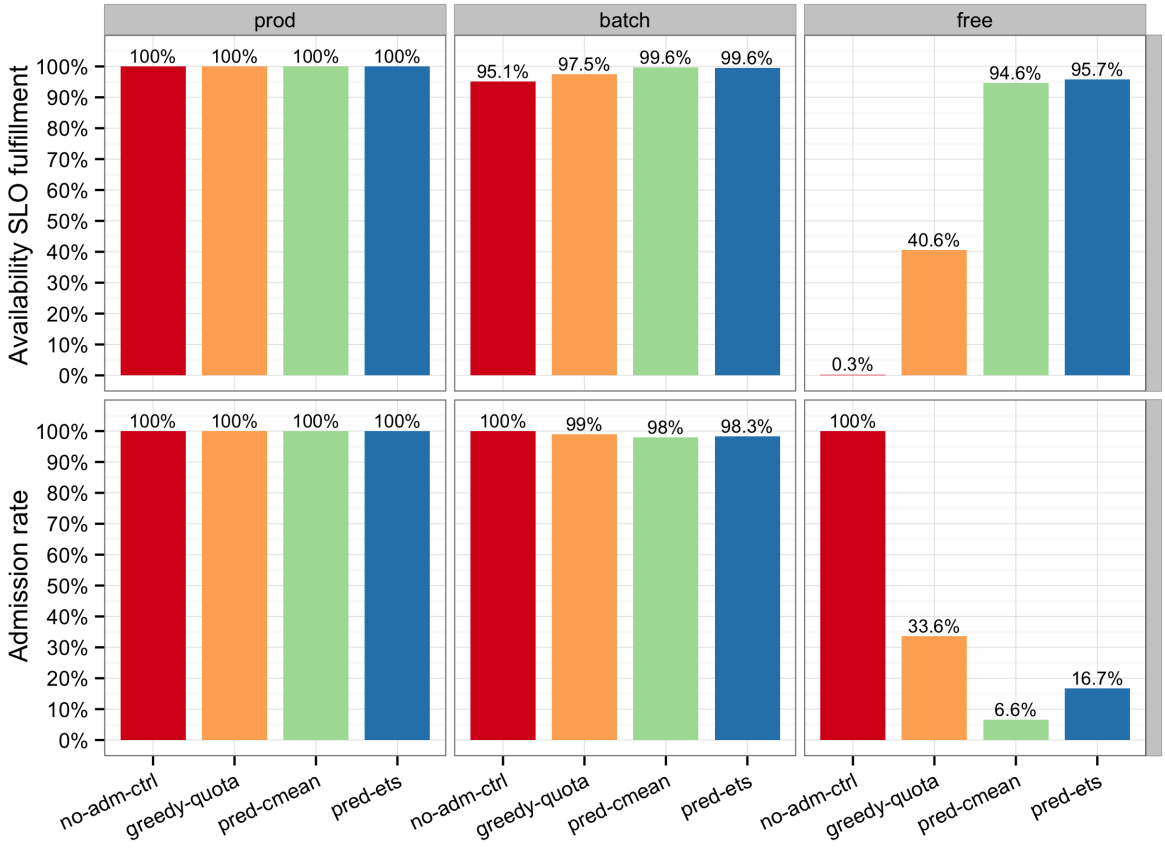

Figura 6.10: Porcentagem de requisições com SLOs de disponibilidade de VMs cumpridos (acima) e taxa de admissão de requisições (abaixo) para cada classe no cenário base.

capacidade suficiente para cumprir o SLO desta classe mesmo sem rejeitar requisições. Para a classe *batch*, a heurística *greedy-quota* teve uma taxa de admissão um pouco maior que as heurísticas preditivas. Para a classe *free*, a heurística *greedy-quota* teve mais que o dobro de admissões da *pred-ets* e um valor mais de cinco vezes maior que o da *pred-cmean*.

Os baixos valores de *admission rate* para os métodos baseados em predição foram necessários para que elas tivessem um alto *SLO fulfillment*, que é considerada a métrica mais importante no contexto estudado. Heurísticas preditivas obtiveram um *SLO fulfillment* maior que a *greedy-quota* porque a estratégia gulosa se baseia apenas na última capacidade disponível observada para definir a quota de cada classe. Esta estratégia simples resulta em capacidades superestimadas com mais frequência, em comparação com métodos de predição mais elaborados que capturam melhor as mudanças de capacidade ao longo do tempo. As heurísticas preditivas também lidam melhor com grandes variâncias ao calcular intervalos

de confiança para as predições e usar seu limite inferior para obter estimativas mais conservadoras, gerando menos violações de SLO. A *pred-ets* apresentou melhores resultados entre as heurísticas preditivas para o cenário base, obtendo valores maiores tanto para *SLO fulfillments* quanto para *admission rate*. O método de predição usado pela *pred-cmean* se mostrou muito conservador ao rejeitar muitas requisições, enquanto o método usado pela *pred-ets* se mostrou mais eficaz, com um bom equilíbrio entre cumprimento de SLOs de disponibilidade de VM e taxa de admissão.

Estes resultados destacam a importância de mecanismos de controle de admissão para atingir metas de disponibilidade de VM para diferentes classes. Observou-se que o *SLO fulfillment* pode ser muito baixo quando o controle de admissão não é usado, especialmente para classes de menor prioridade. O cenário base avaliado apresenta uma baixa contenção de recursos para as classes *prod* e *batch*, resultando em uma alto cumprimento de SLOs e alta taxa de admissão para todas as heurísticas. Por outro lado, a classe *free* de menor prioridade necessitou uma baixa taxa de admissão para ter um alto *SLO fulfillment*, como obtido pelas heurísticas preditivas. Outros cenários de contenção de recursos e SLOs são explorados nas próximas seções.

6.6.2 Análise de sensibilidade para a capacidade da nuvem

A sensibilidade das heurísticas para diferentes capacidades da nuvem foi analisada usando diferentes valores de *capacity size factor*, como mostra a Tabela 6.3. Os valores para os parâmetros de *load factor* e *SLO strength* são os mesmos usados no cenário base.

Parâmetro de entrada	**Valores**
Capacity size factor	$\{0{,}7, 0{,}8, 0{,}9, 1, 1{,}1, 1{,}2, 1{,}3\}$
Load factor	1
Availability SLO strength	$\langle prod = 100\%, batch = 90\%, free = 50\% \rangle$

Tabela 6.3: Parâmetros de entrada para os cenários de análise de sensibilidade para a capacidade da nuvem.

A Figura 6.11 mostra as métricas de *SLO fulfillment* (à esquerda) e *admission rate* (à direita), agregadas para todas as classes em diferentes cenários de capacidade da nuvem. O *SLO fulfillment* tende a diminuir ao se diminuir a capacidade da nuvem para as heurísticas

no-adm-ctrl e *greedy-quota*, que obtiveram respectivamente $0{,}2\%$ e 56% de requisições com SLOs cumpridos no pior caso. Por outro lado, as heurísticas preditivas não foram afetadas significativamente ao se variar a capacidade da nuvem, obtendo no pior caso um *SLO fulfillment* de 91%.

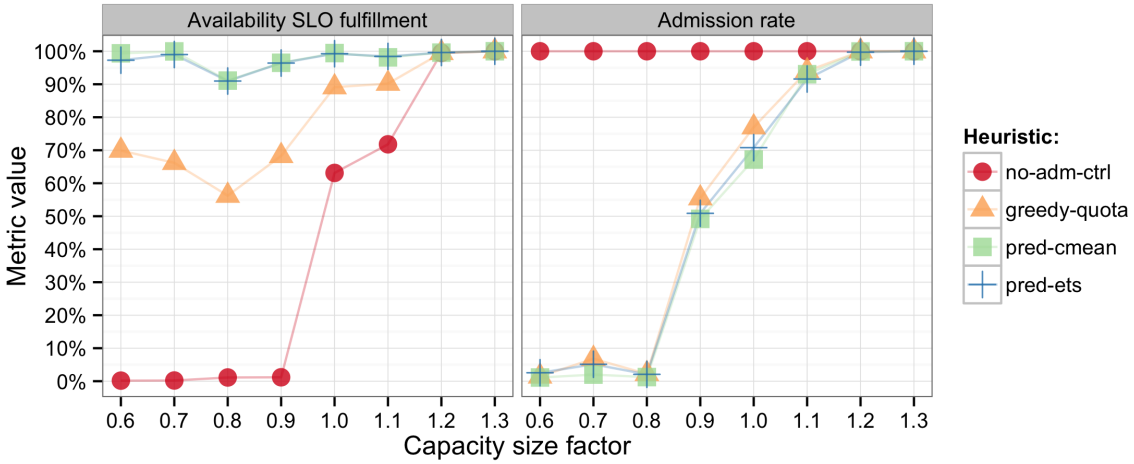

Figura 6.11: Porcentagem de requisições com SLOs de disponibilidade cumpridos (à esquerda) e taxa de admissão (à direita) agregada para todas as classes ao variar a capacidade.

Embora a heurística *greedy-quota* tenha obtido maior taxa de admissão que as heurísticas preditivas, a estratégia gulosa apresentou mais violações de SLO e maior sensibilidade (variação) para diferentes cenários de capacidade da nuvem. Como dito, as heurísticas preditivas conseguem estimar melhor a capacidade disponível para cada classe em épocas futuras e lidar melhor com alta variância ao fazer predições conservadoras. Embora os valores de *SLO fulfillment* para as duas heurísticas preditivas tenham sido muito próximos, a heurística *pred-ets* obteve mais admissões, pois a *pred-cmean* tende a ser mais conservadora e rejeitar mais requisições.

Como dito na análise anterior, a heurística *no-adm-ctrl* sempre admite 100% das requisições. Para as outras heurísticas, o *admission rate* tende a ser menor ao diminuir a capacidade da nuvem. Uma exceção acontece ao reduzir o fator de capacidade de $0{,}8$ para $0{,}7$, caso para o qual a taxa de admissão aumenta. Isto acontece porque no primeiro cenário todas as requisições *prod* são admitidas, enquanto no segundo cenário aproximadamente 25% das requisições *prod* são rejeitadas para cumprir SLOs. Como nos rastros usados as requisições

da classe *prod* requisitam VMs de maior tamanho e por mais tempo, a rejeição de poucas requisições *prod* libera um grande espaço na capacidade suficiente para alocar um número bem maior de VMs *batch* e *free*, que em média demandam bem menos capacidade e tempo, aumentando assim a taxa de admissão geral da nuvem.

A Figura 6.12 mostra as mesmas métricas de *SLO fulfillment* (acima) e *admission rate* (abaixo) em diferentes cenários de capacidade da nuvem, porém agora divididas por classe. Assim como na análise agregada, as heurísticas preditivas obtiveram um alto *SLO fulfillment*, com valores acima de 93% para todos os cenários e classes. Uma exceção acontece com a classe *batch* para o cenário de *capacity size factor* é $0{,}8$, no qual as heurísticas *pred-cmean* e *pred-ets* obtiveram *SLO fulfillments* de $21{,}6\%$ e $69{,}6\%$, respectivamente. Isto acontece porque neste cenário a capacidade para a classe *prod* passa a ser insuficiente, havendo pre-empção de todas as VMs *batch* admitidas para acomodar as requisições *prod* quando há um pico inesperado de demanda. Apesar do baixo cumprimento de SLO neste cenário específico para requisições admitidas na classe *batch*, observa-se que o *admission rate* para esta classe foi menor que $0{,}8\%$ neste cenário, resultando em violações de SLO para uma quantidade muito baixa de requisições e causando pouco impacto no *SLO fulfillment* médio agregado para todas as classes, como visto anteriormente na Figura 6.11.

A Figura 6.13 mostra a utilização média da nuvem para diferentes fatores de capacidade. Como esperado, a utilização da nuvem tende a diminuir ao se aumentar a capacidade da nuvem. As heurísticas preditivas possuem a menor utilização na maioria dos cenários, devido à menor admissão de requisições necessária para cumprir SLOs. Note que quando o *capacity size factor* é igual a $1{,}3$, os valores de utilização, taxa de admissão e cumprimento de SLOs são iguais para todas as heurísticas, representando um cenário indesejado de super-provisão de recursos com baixa utilização e alto custo.

A Figura 6.14 mostra a eficiência da receita para diferentes capacidades da nuvem. Para *capacity size factors* menores que $1{,}1$, a receita para a heurística *no-adm-ctrl* (mostrado apenas parcialmente no gráfico) é sempre negativo. Quando há super-provisão de recursos (fatores de capacidade $1{,}2$ e $1{,}3$), todas as heurísticas possuem receitas altos e similares. No geral, as heurísticas preditivas obtiveram as maiores receitas, sendo bem similares com diferença máxima de $1{,}3\%$ entre elas. A heurística *greedy-quota* também apresentou altos valores de eficiência da receita, tendo uma diferença máxima de $5{,}5\%$ para as heurísticas

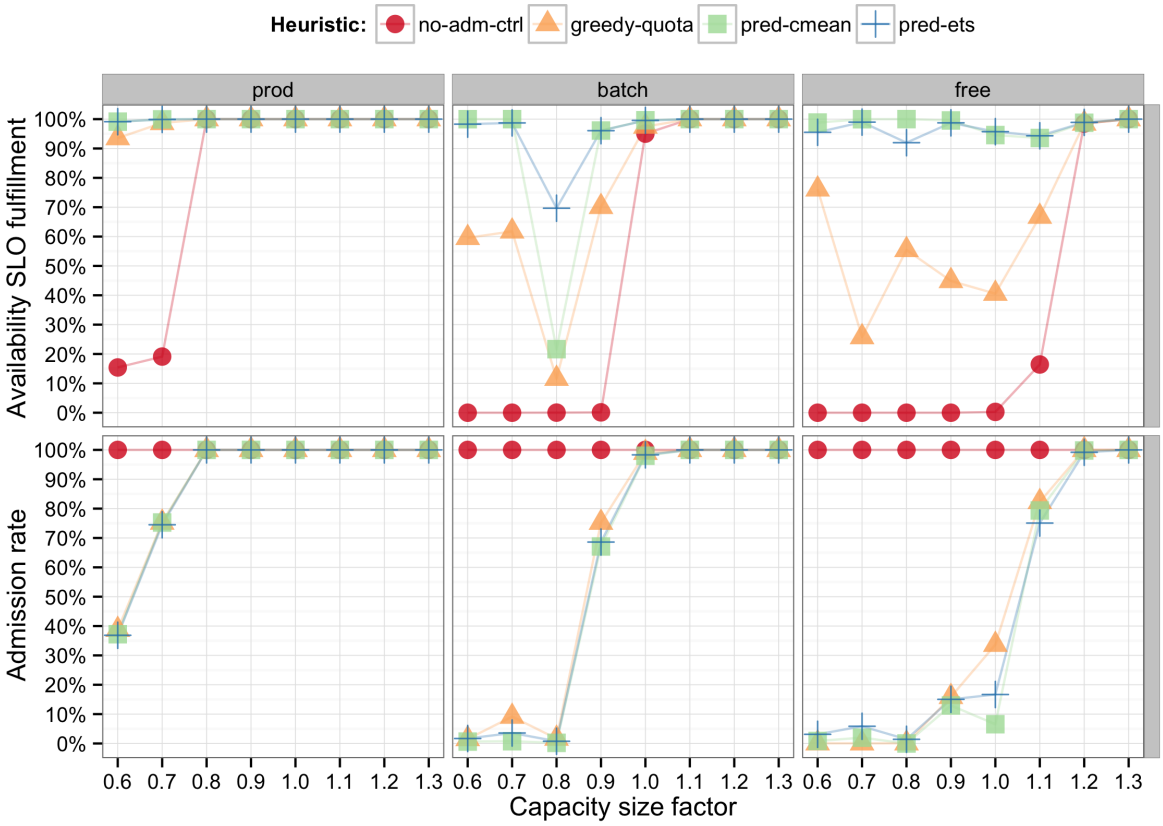

Figura 6.12: Porcentagem de requisições com SLOs de disponibilidade cumpridos (acima) e taxa de admissão (abaixo) para diferentes classes ao variar a capacidade da nuvem.

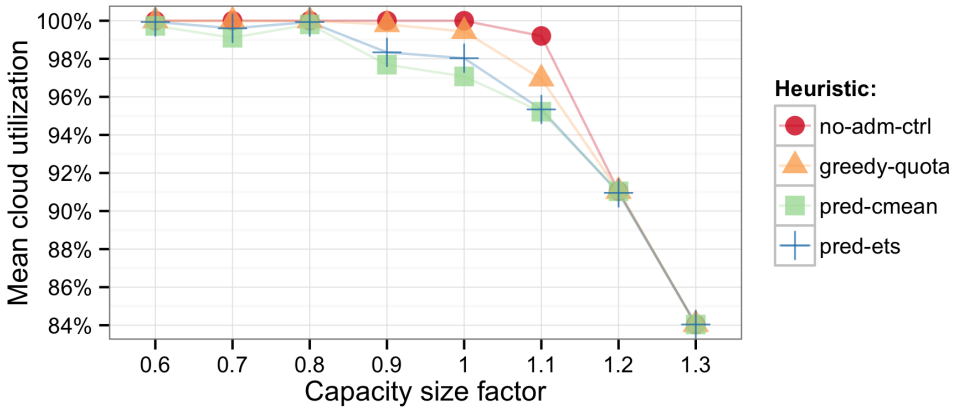

Figura 6.13: Utilização média da nuvem para diferentes capacidades da nuvem.

preditivas.

Vale salientar que a maior parte da receita obtida pelo provedor da nuvem vem da classe

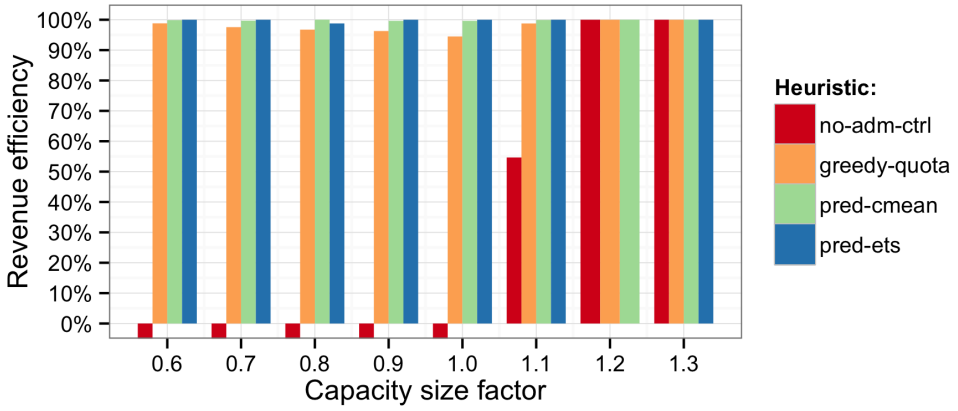

Figura 6.14: Eficiência da receita para diferentes capacidades da nuvem.

prod, pois nos rastros as VMs dessa classe no geral requisitam VMs maiores, executam por mais tempo e consequentemente geram uma maior receita. Como na maioria dos cenários a capacidade total da nuvem é suficiente para alocar todas as requisições da classe *prod*, todas as heurísticas obtiveram um valor alto de receita para esta classe. Portanto, as diferenças nas receitas obtidas pelas diferentes heurísticas provavelmente seriam maiores para cenários em que também houvesse uma alta contenção de recursos para a classe *prod*.

6.6.3 Análise de sensibilidade para a carga da nuvem

A sensibilidade das heurísticas para diferentes cargas de entrada foi analisada variando os valores de *load factor* da nuvem, como mostra a Tabela 6.4. A capacidade da nuvem e os SLOs de disponibilidade são os mesmos do cenário base.

Parâmetro de entrada	**Valores**
Capacity size factor	1
Load factor	$\{0{,}7, 0{,}8, 0{,}9, 1, 1{,}1, 1{,}2, 1{,}3, 1{,}4, 1{,}5, 1{,}6\}$
Availability SLO strength	$\langle prod = 100\%, batch = 90\%, free = 50\% \rangle$

Tabela 6.4: Parâmetros de entrada para os cenários da análise de sensibilidade para a carga de entrada da nuvem.

A Figura 6.15 mostra as métricas de *SLO fulfillment* (à esquerda) e *admission rate* (à direita), agregadas para todas as classes em diferentes cenários de carga da nuvem. Observa-se que quando a carga aumenta o cumprimento dos SLOs tendem a diminuir para as heurísticas

no-adm-ctrl e *greedy-quota*, que no pior caso cumpriram SLOs para $0{,}2\%$ e 36% das requisições, respectivamente. Por outro lado, as heurísticas preditivas não foram significativamente afetadas pela mudança na carga, cumprindo SLOs para 92% das requisições no pior caso.

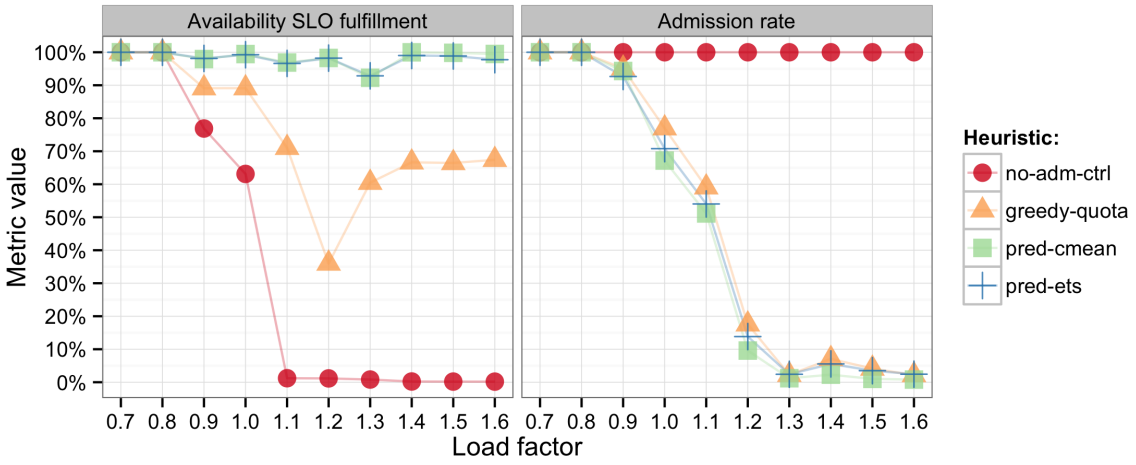

Figura 6.15: Porcentagem de requisições com SLOs de disponibilidade cumpridos (à esquerda) e taxa de admissão (à direita) agregada para todas as classes ao variar a carga.

Para todas as heurísticas, a taxa de admissão tende a diminuir ao se aumentar a carga da nuvem, exceto para *no-adm-ctrl* que não realiza controle de admissão. Para cenários com fatores de carga maiores que $1{,}2$, as taxas de admissão são próximas de zero pois a carga total é muito maior que a capacidade da nuvem nesses cenários. A heurística *greedy-quota* teve uma quantidade de admissões ligeiramente maior do que as heurísticas preditivas, mas o *SLO fulfillment* para a *greedy-quota* foi significantemente menor do que para *pred-cmean* e *pred-ets* na maioria dos cenários, pois as heurísticas preditivas conseguem prever melhor as flutuações futuras da capacidade disponível para as classes, além de serem mais conservadoras.

A Figura 6.16 mostra as métricas de *SLO fulfillment* (acima) e *admission rate* (abaixo), divididas por classe em diferentes cenários de carga da nuvem. Assim como na análise agregada, as heurísticas preditivas obtiveram um alto *SLO fulfillment*, com valores acima de 93% para todos os cenários e classes, exceto para a classe *batch* quando o *load factor* é $1{,}3$ e as heurísticas *pred-cmean* e *pred-ets* obtiveram *SLO fulfillments* de $33{,}4\%$ e $69{,}5\%$, respectivamente. Similar à análise de sensibilidade para a capacidade da nuvem, isto acontece porque

neste cenário a capacidade para a classe *prod* passa a ser insuficiente. Apesar do baixo cumprimento de SLOs para requisições admitidas da classe *batch* neste cenário específico, observa-se que o *admission rate* para a classe *batch* foi menor que $0{,}7\%$ neste cenário, o que causou um impacto muito baixo no *SLO fulfillment* médio agregado para todas as classes, como visto anteriormente na Figura 6.15.

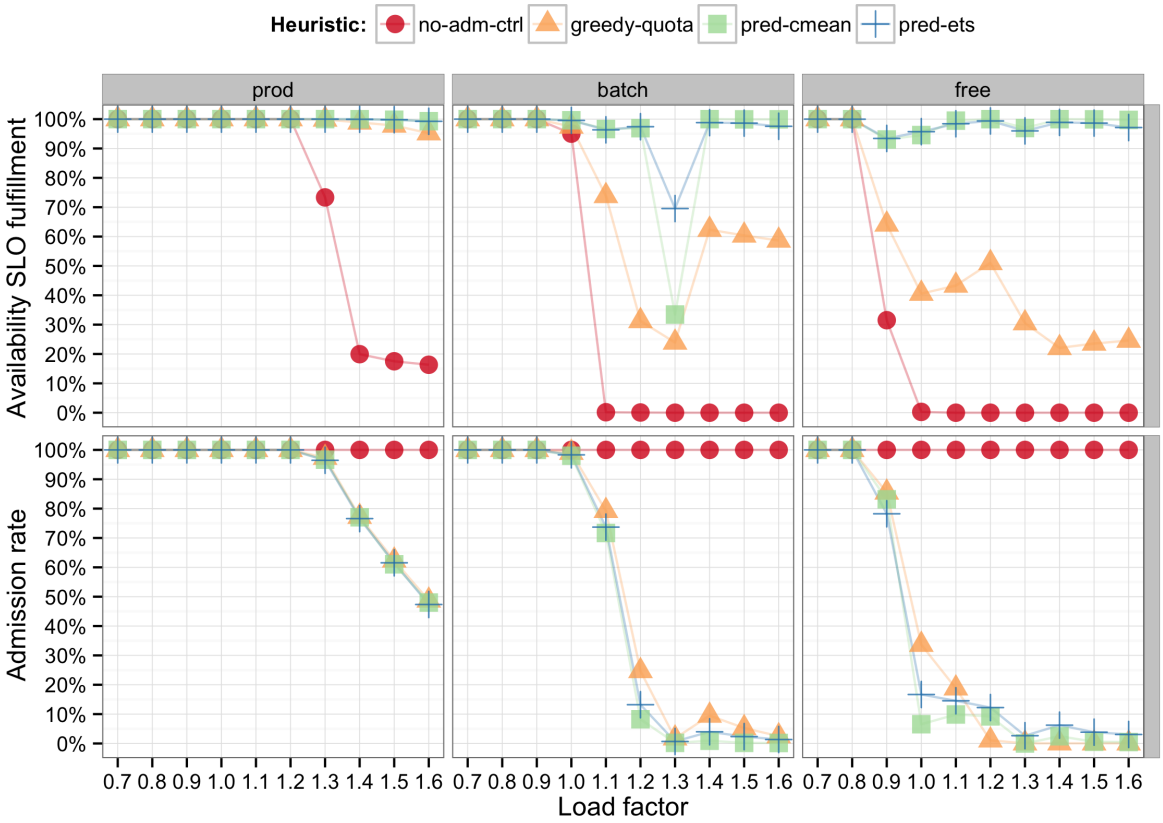

Figura 6.16: Porcentagem de requisições com SLOs de disponibilidade cumpridos (acima) e taxa de admissão (abaixo) para diferentes classes ao variar a carga da nuvem.

A Figura 6.17 mostra a utilização média da nuvem ao variar o fator de carga de trabalho. Para todas as heurísticas, ao se aumentar a carga da nuvem a utilização aumenta até um certo ponto, estabilizando em um valor próximo a 100% em cenários de sobrecarga. Observa-se uma super-provisão de recursos quando o fator de carga é baixo ($\leq 0{,}8$), resultando em uma baixa utilização para todas as heurísticas. Para cenários onde não há super-provisão de recursos, as heurísticas preditivas obtêm uma menor utilização da nuvem, pois elas tendem a rejeitar mais requisições para obter um maior cumprimento de SLOs. Uma forma

de aumentar a utilização da nuvem sem comprometer a taxa de admissão de requisições é diversificar mais os valores de SLO de disponibilidade de VM oferecidos para cada classe, como discutido na próxima seção.

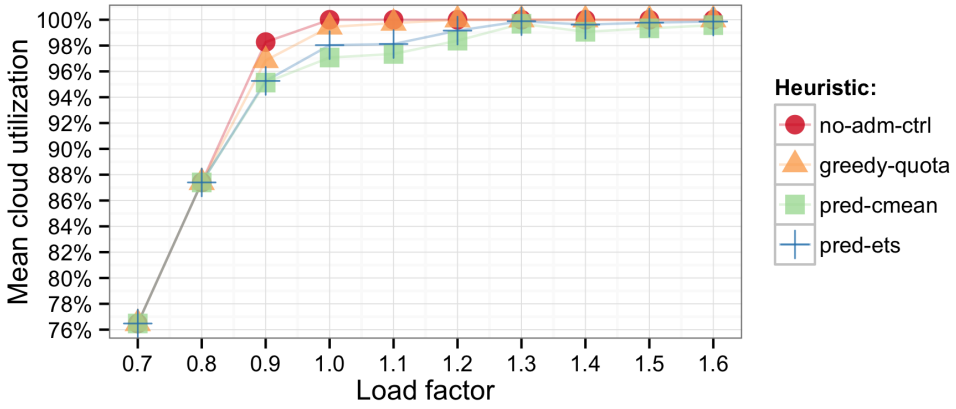

Figura 6.17: Utilização média da nuvem ao variar a intensidade da carga de trabalho.

A Figura 6.18 mostra a eficiência da receita para diferentes cargas de trabalho da nuvem. Para *load factors* maiores ou igual a 1, a receita da heurística *no-adm-ctrl* (mostrado apenas parcialmente no gráfico) é sempre negativa. Por outro lado, quando há baixa contenção de recursos (fatores de carga $0{,}7$ e $0{,}8$), todas as heurísticas possuem receitas similarmente altos. No geral, as heurísticas preditivas obtiveram os maiores receitas, com valores bem próximos tendo uma diferença máxima de $1{,}1\%$ entre elas. A heurística *greedy-quota* também apresentou valores altos de eficiência da receita, tendo uma diferença máxima de $5{,}5\%$ para as heurísticas preditivas. Já as receita obtidas pela heurística que não usa controle de admissão em cenários sem super-provisão de recursos foram bastante inferiores aos das outras heurísticas que utilizam o modelo de quotas proposto nesta tese, o que enfatiza a importância do controle de admissão para aumentar a receita dos provedores.

6.6.4 Análise de sensibilidade para SLOs

A sensibilidade das heurísticas para diferentes SLOs também foi analisada variando os valores de SLOs de disponibilidade de VMs para cada classe, obtendo diferentes cenários de *SLO strength* com combinações de valores que variam de uma qualidade considerada muito baixa (*very-low*) até muito alta (*very-high*), como mostra a Tabela 6.5. O fator de capacidade e de carga são os mesmos do cenário base.

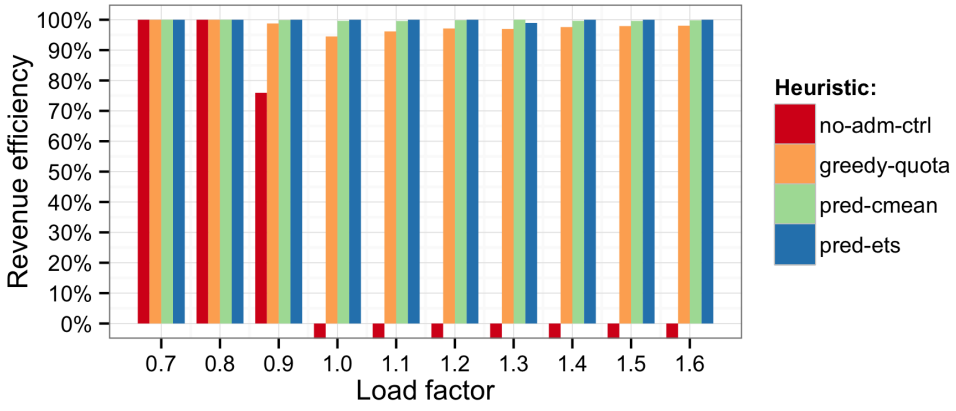

Figura 6.18: Eficiência da receita para diferentes cargas da nuvem.

Parâmetro de entrada	**Valores**
Capacity size factor	1
Load factor	1
Availability SLO strength $\langle prod, batch, free \rangle$	very-low $\langle 100\%, 50\%, 10\% \rangle$ low $\langle 100\%, 70\%, 30\% \rangle$ medium $\langle 100\%, 90\%, 50\% \rangle$ high $\langle 100\%, 99\%, 70\% \rangle$ very-high $\langle 100\%, 99{,}9\%, 90\% \rangle$

Tabela 6.5: Parâmetros de entrada para os cenários da análise de sensibilidade para SLOs.

A Figura 6.19 mostra as métricas de *SLO fulfillment* (à esquerda) e *admission rate* (à direita), agregadas para todas as classes em diferentes cenários de *SLO strength*. As heurísticas que usam controle de admissão com quotas (todas exceto *no-adm-ctrl*) tendem a definir quotas mais conservadoras para SLOs com qualidade mais alta, resultando em um maior *SLO fulfillment*. No pior caso, cada heurística cumpriu SLOs para $63{,}0\%$ (*no-adm-ctrl*), $84{,}9\%$ (*greedy-quota*), $94{,}9\%$ (*pred-cmean*) e $93{,}2\%$ (*pred-ets*). Para as heurísticas preditivas, a taxa de admissão diminui quando os SLOs são mais fortes, pois mais rejeições são necessárias para cumpri-los.

A Figura 6.20 mostra as mesmas métricas de *SLO fulfillment* (acima) e *admission rate* (abaixo) em diferentes cenários de SLO, porém agora divididas por classe. Assim como na análise agregada, as heurísticas preditivas obtiveram altos valores de *SLO fulfillment*, exceto para a classe *free* no cenário de *SLO strength* muito baixo, no qual as heurísticas *pred-cmean*

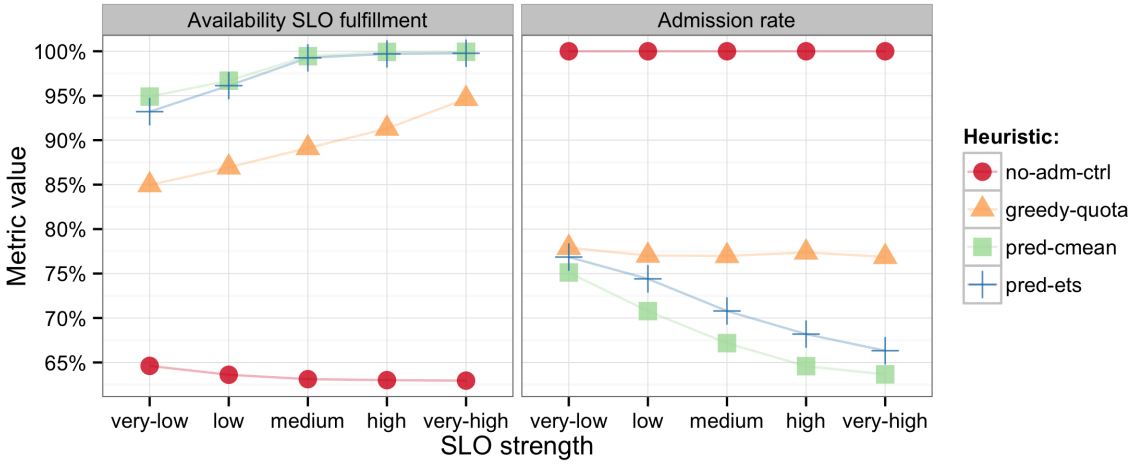

Figura 6.19: Porcentagem de requisições com SLOs de disponibilidade cumpridos (à esquerda) e taxa de admissão (à direita) ao variar a qualidade dos SLOs das classes.

e *pred-ets* obtiveram *SLO fulfillments* de $74{,}7\%$ e $66{,}0\%$, respectivamente. Isto acontece porque os métodos de predição tendem as ser menos conservadores para valores menores de SLO, admitindo mais requisições e ficando mais suscetíveis a violações de SLO.

A Figura 6.21 mostra a utilização média da nuvem ao variar o *SLO strength*. A utilização diminui quando se aumenta a qualidade dos SLOs para as heurísticas que aplicam o controle de admissão com quotas, que é consequencia de uma menor taxa de admissão necessária para cumprir SLOs de maior qualidade.

A Figura 6.22 mostra a eficiência da receita para diferentes combinações de SLOs definidos para as classes. Em todos os cenários de *SLO strength*, a receita para a heurística *no-adm-ctrl* (mostrado apenas parcialmente no gráfico) é negativo. As heurísticas preditivas obtiveram os maiores receitas, sendo bem similares entre elas com diferença máxima de $0{,}7\%$ entre elas. A heurística *greedy-quota* também apresentou altos valores de eficiência da receita, tendo uma diferença máxima de $9{,}1\%$ para as heurísticas preditivas no cenário de SLOs com qualidade muito baixa. As receitas negativas obtidas pela heurística que não usa controle de admissão em todos os cenários reforça a importância de controle de admissão para a receita dos provedores.

O cumprimento de SLOs para as heurísticas preditivas não foi afetado significativamente ao variar os SLOs de disponibilidade de VMs definidos para cada classe. Observou-se tam-

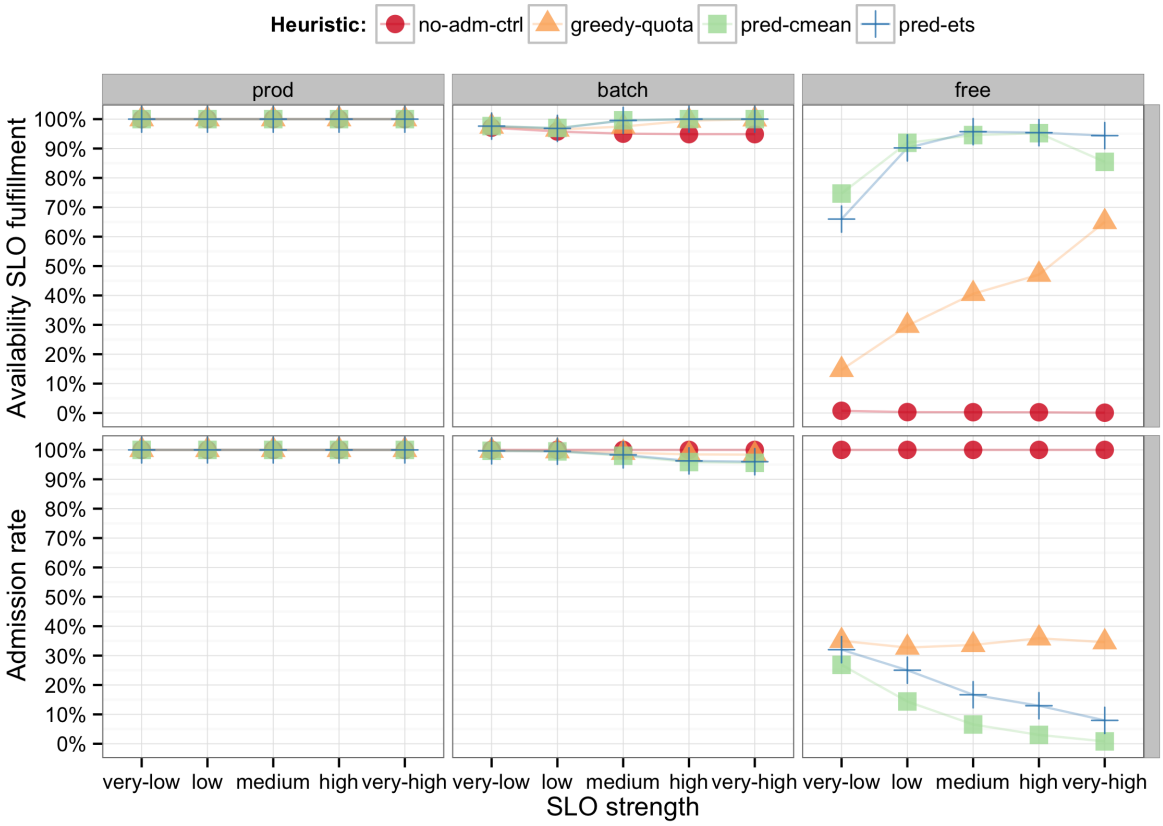

Figura 6.20: Porcentagem de requisições com SLOs de disponibilidade cumpridos (à esquerda) e taxa de admissão (à direita) para diferentes classes ao variar a qualidade dos SLOs das classes.

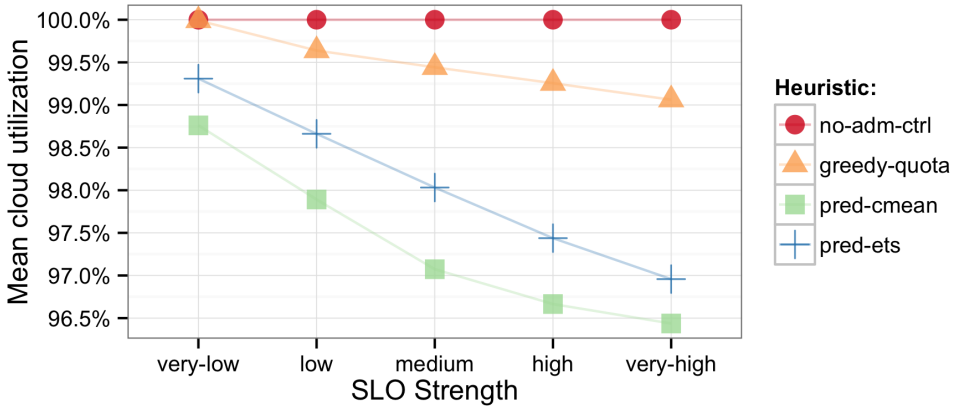

Figura 6.21: Utilização média da nuvem ao variar a qualidade dos SLOs das classes.

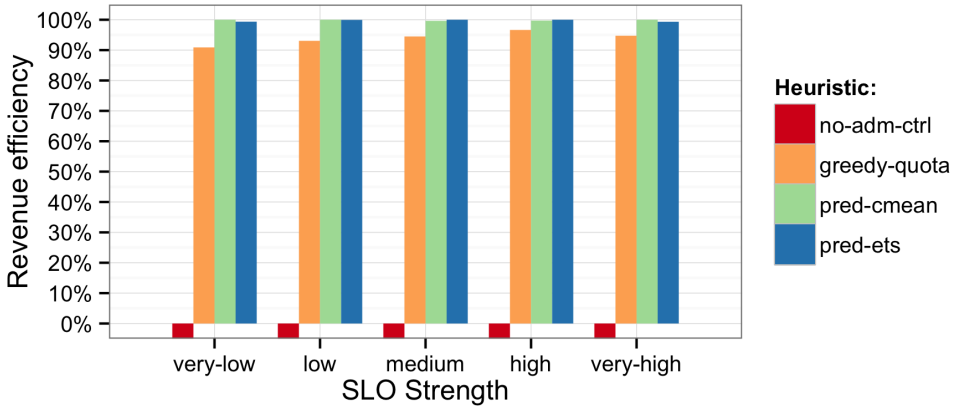

Figura 6.22: Eficiência da receita ao variar a qualidade dos SLOs das classes.

bém que definir SLOs de alta qualidade para todas as classes resulta em uma baixa utilização da nuvem, o que sugere ser mais vantajoso para o provedor oferecer uma maior faixa de SLOs ao invés de concentrar as metas de QoS em altos valores para todas as classes, para que a capacidade da nuvem seja utilizada de forma mais eficiente. As heurísticas preditivas também permitiram o aumento da receita dos provedores para todos os cenários de *SLO strength* avaliados.

6.7 Conclusões do estudo

Provedores de nuvem de IaaS podem oferecer diversas classes de serviço para aumentar a utilização da nuvem e consequentemente suas receitas, atraindo usuários com diferentes orçamentos e requisitos de QoS. Desta forma, um importante problema para provedores da nuvem de IaaS é como cumprir os SLOs definidos para as diferentes classes oferecidas. Neste capítulo, um modelo de controle de admissão baseado em predição foi proposto para endereçar este problema. O modelo define quotas dinamicamente para cada classe, limitando a admissão de requisições de VM através da predição da capacidade disponível para cada classe no futuro.

Os resultados mostraram que controle de admissão é necessário para cumprir SLOs de disponibilidade de VM, a menos que haja uma super-provisão de recursos, o que não é desejável. As heurísticas baseadas em predição não foram afetadas significativamente quando se explorou diferentes cenários de demanda e capacidade da nuvem, além da variação dos

SLOs de cada classe oferecida, apresentando consistentemente altas quantidades de SLOs satisfeitos e maiores receitas quando comparada a outras heurísticas. Constatou-se também que é importante para o provedor oferecer SLOs diversificados para as diferentes classes, com valores baixos para classes de menor prioridade e altos para classes mais importantes, para se obter uma maior utilização do sistema e maior lucratividade, podendo também atrair mais usuários.

Capítulo 7

Planejamento de capacidade em nuvens de IaaS com múltiplas classes

No capítulo anterior, foi apresentado um modelo de controle de admissão que permite a um provedor de IaaS cumprir SLOs de disponibilidade de VM definidos para diferentes classes de serviço. Porém, em cenários de alta contenção de recursos, observou-se que o mecanismo de controle de admissão precisa rejeitar grande parte das requisições para cumprir os SLOs de disponibilidade de VM para as requisições admitidas. Neste capítulo[1], resolve-se este problema propondo-se um modelo de planejamento de capacidade para nuvens de IaaS com múltiplas classes de serviço, que determina a capacidade da nuvem necessária para cumprir SLOs tanto de disponibilidade de VM como também para a taxa de admissão de requisições de VM para as classes oferecidas. Além disso, propõe-se um modelo analítico capaz de estimar métricas de QoS da nuvem para diferentes cenários de demanda, capacidade e classes de serviço, completando assim o modelo de gerência de recursos de nuvem de IaaS proposto nesta tese.

[1]Este capítulo é baseado no artigo submetido para o periódico *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, que ainda estava em processo de revisão durante a escrita deste documento. O artigo está disponível como *Technical Report* [25]

7.1 Introdução

Como demonstrado no Capítulo 6, mecanismos de controle de admissão baseados em predição podem reduzir significativamente as violações de SLO de disponibilidade de VMs, em um ambiente de nuvem de IaaS com múltiplas classes. Porém, em cenários de alta contenção, observou-se que é necessário rejeitar grande parte das requisições para se cumprir os SLOs de disponibilidade de VM. Apesar da rejeição de requisições ser considerada menos prejudicial que violações futuras de seus SLOs de disponibilidade, ela pode ser indesejável para usuários que precisam de elasticidade para escalar suas aplicações. Para lidar com este problema, provedores podem definir metas para a taxa de admissão de requisições (i.e., *Admissibility SLOs*) para as diferentes classes e usar métodos de planejamento de capacidade para encontrar a capacidade mínima da nuvem necessária para cumprir estas metas.

Este capítulo mostra como fazer o planejamento de capacidade de nuvens de IaaS, de tal forma que os SLOs de admissibilidade e de disponibilidade de VMs sejam cumpridos para as diferentes classes oferecidas. Um modelo analítico baseado em teoria das filas é proposto para estimar métricas de QoS para diferentes cenários de demanda, oferta de capacidade da nuvem e classes definidas com diferentes SLOs. Este modelo analítico é usado no planejamento de capacidade da nuvem para, em conjunto com o método de controle de admissão proposto anteriormente, permitir que o provedor cumpra as metas de QoS definidas para as diferentes classes sem precisar super-provisionar seus recursos.

As principais contribuições deste estudo são:

- A formalização de parte do problema de gerência de recursos da nuvem, incluindo decisões de planejamento de capacidade, formulado como um modelo de otimização;
- Um modelo analítico para estimar métricas de QoS e planejar a capacidade de nuvens de IaaS com múltiplas classes, de tal forma que SLOs de disponibilidade de VM e admissibilidade sejam cumpridos;
- Uma validação do modelo comparado-o com resultados obtidos através de simulações, usando rastros de sistemas em produção;
- uma demonstração da eficácia do modelo em planejar a capacidade da nuvem, buscando a capacidade mínima necessária para cumprir os SLOs de cada classe.

Os resultados mostraram que o modelo analítico consegue estimar bem a taxa de admissão de requisições para as diferentes classes, apresentando um erro absoluto de 0,4% em média para a classe de maior prioridade e de 7,8% para a classe de menor prioridade, quando comparado aos valores simulados. O modelo também mostrou-se bastante útil para realizar o planejamento de capacidade da nuvem em diferentes cenários, tomando decisões de capacidade muito similares à melhor capacidade encontrada através de simulações.

O restante deste capítulo está estruturado da seguinte forma. A Seção 7.2 inclui decisões de planejamento de capacidade na formalização do problema da gerência de recursos da nuvem, elaborado como um modelo de otimização. A Seção 7.3 apresenta o modelo analítico do planejamento de capacidade proposto nesta tese, com a descrição das fórmulas de teoria das filas usadas na modelagem e como o modelo foi elaborado com base nelas. A Seção 7.4 mostra a validação do modelo analítico, comparando suas estimativas com resultados obtidos em simulações baseadas em rastros de uma nuvem de larga escala. Por fim, a Seção 7.5 apresenta as conclusões do estudo.

7.2 Formalização do problema

Nesta seção, a formalização do problema da gerência de recursos da nuvem apresentado na Seção 6.2 é adaptado para considerar decisões de planejamento de capacidade. O problema também é formulado como um modelo de otimização, tendo como objetivo minimizar a capacidade nominal da nuvem, sujeito ao cumprimento dos SLOs de admissibilidade e disponibilidade de VMs para todas as classes.

As variáveis de decisão para o planejamento de capacidade são adicionadas nesta formulação, resultando nas seguintes variáveis de decisão para cada fase da gerência de recursos da nuvem:

- **Planejamento de capacidade**: conjunto de H máquinas que irão compor a nuvem, onde cada máquina h ($1 \leq h \leq H$) possui capacidade C_h e disponibilidade média A_h.

- **Controle de admissão**: variáveis binárias de admissão x_{jr} para cada requisição j e cada classe r.

- **Escalonamento**: variáveis binárias de alocação y_{ijrh} para cada requisição j da classe r na máquina h durante a época E_i.

O conjunto de valores de entrada usados em uma solução para um certo período de observação, que representa um **cenário de decisão**, é definido nesta formulação como

$$\Psi' = \langle N, R, W_r, D_{jr}, S_{jr}, \gamma_r^{min}, \theta_r^{min} \rangle.$$

Com isso, o problema de otimização para a gerência da nuvem, considerando decisões de planejamento de capacidade, é formulado como:

Dado Ψ',

minimizar

$$\mathcal{C} = \sum_{h=1}^{H} C_h, \tag{7.1}$$

sujeito a

$$\sum_{r=1}^{R} \sum_{j=1}^{|W_r|} y_{ijrh} \cdot D_{jr} \leq C_h \cdot A_{ih}, \quad \forall i, h \tag{7.2}$$

$$y_{ijrh} \leq x_{jr}, \quad \forall i, j, r, h \tag{7.3}$$

$$\gamma_{jr} \geq \gamma_r^{min}, \ \ \forall j, r \tag{7.4}$$

$$\theta_r \geq \theta_r^{min}, \ \ \forall r \tag{7.5}$$

A função objetivo (Equação 7.1) a ser minimizada representa a capacidade nominal da nuvem, que é a soma das capacidades de todas as H máquinas definidas na fase de planejamento de capacidade. A restrição de capacidade (Equação 7.2), similar à formulação anterior, estabelece que a capacidade alocada para todas as VMs em uma máquina não podem ser maiores que a capacidade da máquina, e que máquinas indisponíveis para o provedor não devem possuir VMs alocadas. A restrição de admissão (Equação 7.3), que também é igual à formulação anterior, define que uma VM só pode ser alocada se e somente se sua requisição for admitida na fase de controle de admissão. Finalmente, as restrições de SLO estabelecem que as disponibilidades das VMs (Equação 7.4) e as taxas de admissão (Equação 7.5) não

podem ser menores que os valores mínimos aceitáveis (i.e., os SLOs) definidos para cada classe, sendo adicionada para esta formulação a restrição do SLO de admissibilidade.

Este problema de otimização também é NP-completo e necessita de dados futuros detalhados e exatos para a demanda e oferta de recursos ao longo do tempo, que não é algo realista de ser obtido e é complexo de ser solucionado. Neste capítulo, propõe-se um modelo analítico que usa como entrada estatísticas genéricas da demanda e da capacidade disponível da nuvem para estimar a QoS a ser obtida para cada classe. O modelo também é usado para planejar a capacidade mínima da nuvem necessária para cumprir SLOs de admissibilidade e disponibilidade de VMs para cada classe.

7.3 Modelo do planejamento de capacidade da nuvem

Nesta seção o modelo analítico do planejamento de capacidade proposto neste trabalho é apresentado. Este modelo usa como base o sistema de nuvem de IaaS descrito no Capítulo 4 e o método de controle de admissão proposto no Capítulo 6.

A elaboração do modelo foi feita com base em teoria das filas, usando aproximações de difusão propostas para filas do tipo $G/GI/c/K$ [78]. Primeiramente, as aproximações de difusão usadas são apresentadas e, em seguida, o modelo de planejamento de capacidade baseado nessas aproximações é formulado.

7.3.1 Aproximações de difusão para filas G/GI/c/K

De acordo com a teoria das filas, o tipo de fila $G/GI/c/K$ possui: processo de chegada geral (G), sendo aplicável a qualquer distribuição de probabilidade; tempos de serviço independentes e identicamente distribuídos com uma distribuição geral (GI); múltiplos servidores (c); e um número máximo de clientes permitidos no sistema (K). No contexto do sistema de nuvem computacional deste trabalho, usou-se um modelo de filas de capacidade limitada, onde clientes (requisições) são rejeitados quando a capacidade do sistema atinge o seu limite. Este modelo de filas representa bem o mecanismo de controle de admissão usado neste trabalho, no qual requisições de VM são rejeitadas quando a quota referente à sua classe é excedida.

O mapeamento entre o sistema de nuvem de IaaS e o modelo de filas $G/GI/c/K$ é feito da seguinte forma:

1. A quantidade de *CPU-cores* das máquinas físicas disponíveis na nuvem correspondem aos c servidores do modelo de filas;

2. A quantidade de *CPU-cores* alocados para VMs em execução ou de requisições na fila de espera correspondem ao número de clientes no sistema de filas;

3. A quantidade máxima de *CPU-cores* requisitada por VMs admitidas pelo provedor (i.e., a quota para cada classe) é representada pelo limite K da capacidade do sistema.

Embora até o momento não haja na literatura uma solução exata para sistemas de fila $G/GI/c/K$, existem aproximações que são bastante úteis. No modelo do planejamento de capacidade deste trabalho, foi usada uma aproximação para filas $G/GI/c/K$ proposta por Whitt [78], que é uma aproximação de difusão baseada em limites de tráfego intenso (conhecida em inglês por *heavy-traffic* ou *diffusion approximations*). Este tipo de aproximação é adequada para sistemas com um grande número de servidores e com alta intensidade de tráfego, servindo bem para o contexto de nuvem computacional abordado neste trabalho, que tipicamente envolve muitos servidores e busca uma alta utilização do sistema.

A seguir são apresentadas as principais fórmulas da aproximação usadas no modelo do planejamento de capacidade. Mais detalhes sobre essas fórmulas podem ser encontrados no artigo de Whitt [78].

A aproximação de difusão para filas $G/GI/c/K$ é baseada no limite β para tráfego intenso, definido como

$$\beta = \sqrt{c} \cdot (1 - \rho) \quad \text{para } 0 < \beta < \infty, \tag{7.6}$$

onde ρ é a intensidade do tráfego e c o número de servidores no sistema.

Para sistemas de fila com capacidade finita, um limite adicional κ é usado ao considerar que $K \to \infty$ quando $c \to \infty$, de tal forma que

$$\kappa = \frac{K - c}{\sqrt{c}} \quad \text{para } 0 < \kappa \leq \infty, \tag{7.7}$$

onde K é o número máximo de clientes permitidos no sistema.

Esta aproximação usa em suas fórmulas uma medida de *asymptotic peakedness*, que quantifica a intensidade da variação na carga do sistema. A métrica de *peakedness* é definida pela variância dividida pela média amostral do número de clientes no sistema em equilíbrio (*steady-state*). O valor da medida de *asymptotic peakedness* se aproxima do valor de *peakedness* à medida que a taxa de chegada no sistema aumenta no regime de tráfego intenso.

A métrica de *asymptotic peakedness* se baseia na distribuição do tempo de serviço H_2^* e é definida como

$$z = \frac{p \cdot (c_a^2 + c_s^2)}{2}, \tag{7.8}$$

onde c_a^2 é o quadrado do coeficiente de variação (SCV, do inglês *squared coefficient of variation*), calculado como a variância dividida pelo quadrado da média) para o tempo entre chegadas; c_s^2 é o quadrado do coeficiente de variação para o tempo de serviço; e a distribuição H_2^* considerada para o tempo de serviço é uma mistura de uma distribuição exponencial com probabilidade p e um ponto de massa em 0 com probabilidade $1 - p$.

Um outro parâmetro de variação v usado na aproximação, que é relacionado à métrica de *asymptotic peakedness*, é definido por

$$v = \frac{z}{p} = \frac{c_a^2 + c_s^2}{2}. \tag{7.9}$$

A função de *asymptotic-delay-probability* representa a probabilidade de atraso para clientes no modelo de filas. Ela tem como parâmetro os limites β e κ, definidos no regime de tráfego intenso nas Equações 7.6 e 7.7, e é definida por

$$\alpha(\beta, \kappa) = [1 + \beta \cdot \Phi(\beta)/(\phi(\beta) \cdot (1 - e^{-\kappa\beta}))]^{-1}, \tag{7.10}$$

onde Φ é a função distribuição acumulada (FDA) e ϕ é a função densidade de probabilidade (FDP) para uma variável aleatória com distribuição normal padrão.

A aproximação para a probabilidade de atraso (*delay probability*) para filas $G/GI/c/K$ é definida como

$$\tau = \alpha(\beta/\sqrt{z}, p \cdot \kappa/\sqrt{z}), \tag{7.11}$$

onde $\alpha(.)$ é a função de *asymptotic-delay-probability* definida na Equação 7.10.

Por fim, a probabilidade de rejeição de clientes (*blocking probability*) por atingir o limite K da capacidade do sistema para filas $G/GI/c/K$ é aproximada pela função

$$\pi(\kappa, \tau, \beta, v, \rho, c) = \frac{f(\kappa, \tau, \beta, v) \cdot v}{\rho \cdot \sqrt{c}}, \tag{7.12}$$

onde $f(.)$ é a FDP da quantidade de clientes na fila para um sistema em equilíbrio, definido para um limite κ como:

$$f(\kappa, \tau, \beta, v) = \frac{\tau \cdot \beta \cdot e^{-\kappa\beta/v}}{v \cdot (1 - e^{-\kappa\beta/v})}. \tag{7.13}$$

7.3.2 Formulação do modelo

Uma limitação da aproximação para filas $G/GI/c/K$ descrita na Seção 7.3.1 em relação ao problema tratado nesta tese é que ela foi derivada para uma única classe de clientes [78], enquanto neste trabalho considera-se um provedor de nuvem oferecendo múltiplas classes de serviço, com diferentes demandas e SLOs. Desta forma, não é trivial aplicar diretamente a aproximação do modelo de filas neste contexto. Como a capacidade da nuvem disponível para classes de menor prioridade depende da demanda de classes de maior prioridade, o modelo não pode ser resolvido independentemente para cada uma das classes. Portanto, propõe-se uma abordagem hierárquica que soluciona o modelo iterativamente, iniciando com a classe de maior prioridade e finalizando com a de menor prioridade.

Neste trabalho foi elaborado um modelo analítico do planejamento de capacidade para responder questões do tipo *what-if*, que indicam o que aconteceria com métricas de desempenho da nuvem se houvessem diferentes cenários de entrada. Mais especificamente, o objetivo do modelo é estimar a taxa de admissão de requisições para cada classe, tendo como entrada a capacidade média da nuvem, a demanda esperada e os SLOs de disponibilidade de VMs para cada classe. Este modelo analítico também é usado para planejar a capacidade da nuvem, estimando a capacidade mínima necessária para atender metas para os SLOs de disponibilidade de VMs e de taxa de admissão para todas as classes.

Os parâmetros de entrada para o modelo analítico são os seguintes:

- $\mathcal{C}$: capacidade nominal da nuvem em *CPU-cores*.
- $\bar{A}$: disponibilidade média das máquinas.
- λ_r : taxa de chegada para cada classe r em *CPU-cores* por unidade de tempo.
- μ_r : taxa de serviço para cada classe r em *CPU-cores* por unidade de tempo.
- $c^2_{a,r}$: quadrado do coeficiente de variação (SCV) do intervalo entre chegadas para cada classe r.

- $c_{s,r}^2$: quadrado do coeficiente de variação (SCV) do tempo de serviço para cada classe r.

- γ_r^{min} : mínima disponibilidade de VM permitida (SLO) para cada classe r.

Para um conjunto de valores de entrada, o modelo gera como saída uma estimativa para:

- θ_r : taxa de admissão de requisições de VM para cada classe r, ao se usar o controle de admissão baseado em quotas.

Uma outra limitação da aproximação para filas $G/GI/c/K$ é que ela considera um número constante c de servidores. Porém, no contexto deste trabalho a quantidade de servidores pode mudar ao longo do tempo, fazendo com que a capacidade disponível na nuvem também varie. Além disso, as máquinas físicas e os tamanhos das VMs requisitadas na nuvem são tipicamente heterogêneas, enquanto o modelo de filas assume que a associação entre cliente e servidor é direta e homogênea. Para lidar com estas limitações, este modelo considera que o número de servidores c no sistema de filas é representado pela média da capacidade disponível na nuvem em *CPU-cores*, definido por

$$c = \bar{A} \cdot \mathcal{C}, \tag{7.14}$$

onde $\bar{A}$ é a disponibilidade média das máquinas e $\mathcal{C}$ é a capacidade nominal da nuvem em *CPU-cores*.

De acordo com a política de escalonamento preemptivo por prioridade descrito na Seção 6.4, cada classe r pode ter diferentes capacidades disponíveis para alocar requisições de VM. A capacidade média disponível para a classe r é calculada como a média da capacidade restante das classes de maior prioridade que r, e é definida por

$$c_r = c - \sum_{k=1}^{r-1} n_k \tag{7.15}$$

onde n_k é a quantidade média de clientes da classe k no sistema, que é igual à média da capacidade requisitada por VMs da classe k em *CPU-cores* ($n_k \geq 0, \forall k$). Observe que a capacidade disponível para uma classe nunca é maior que a capacidade disponível para classes de maior prioridade ($c_r \leq c_k, \forall r > k$).

A intensidade do tráfego ρ_r para a classe r é estimada usando a Lei de Little [58] e é definida por

$$\rho_r = \frac{\lambda_r}{\mu_r \cdot c_r}, \tag{7.16}$$

O cálculo do número de clientes da classe r no sistema também se baseia na Lei de Little e é definido como

$$n_r = \frac{\lambda_r}{\mu_r} = c_r \cdot \rho_r. \tag{7.17}$$

Considera-se que o número máximo de clientes para cada classe r permitidos no sistema com capacidade limitada é igual à quota definida para a classe. A partir do modelo de controle de admissão baseado em quotas proposto nesta tese (Equação 6.7), define-se o número máximo de clientes no sistema (quota) para a classe r como

$$K_r = \frac{c_r}{\gamma_r^{min}}, \tag{7.18}$$

onde γ_r^{min} é o SLO de disponibilidade de VMs para a classe r.

Assumiu-se que a distribuição de tempo de serviço é uma combinação de distribuições exponenciais – ou seja, considerou-se $p = 1$ na aproximação descrita na Seção 7.3.1, já que este parâmetro é difícil de ser estimado na prática. Esta simplificação torna a solução mais simples e não afeta significativamente os resultados. Desta forma, a partir das Equações 7.8 e 7.9, pode-se definir os parâmetros de variação do modelo como

$$v_r = z_r = \frac{c_{a,r}^2 + c_{s,r}^2}{2}. \tag{7.19}$$

A taxa de admissão para a classe r é estimada como a probabilidade de uma requisição ser admitida (i.e., o inverso da *blocking probability*) de acordo com a aproximação para filas $G/GI/c/K$, definida por

$$\theta_r = 1 - \pi(\kappa_r, \tau_r, \beta_r, v_r, \rho_r, c_r), \tag{7.20}$$

onde $\pi(.)$ é a função de probabilidade de rejeição aproximada para filas $G/GI/c/K$, definida na Equação 7.12.

A seguir são apresentados os algoritmos que implementam o modelo de planejamento de capacidade proposto neste trabalho.

O Algoritmo 2 estima a taxa de admissão θ_r para cada classe r quando o mecanismo de controle de admissão baseado em quotas está em operação, usando o modelo analítico

proposto. Esta função recebe como entrada a capacidade da nuvem, assim como a taxa média de chegada, tempo médio de serviço, parâmetro de variação da demanda e SLO de disponibilidade de VMs para cada classe.

Algoritmo 2 Estima a taxa de admissão de requisições para múltiplas classes.

1: **function** ESTIMATEQOS(c, λ_r, μ_r, v_r, γ_r^{min})
2: $p \leftarrow 1$
3: $c_1 \leftarrow c$ ▷ Eq. 7.14
4: $R \leftarrow$ *length*(λ_r)
5: **for** $r \leftarrow 1, R$ **do**
6: $\rho_r \leftarrow \lambda_r/(\mu_r * c_r)$ ▷ Eq. 7.16
7: $K_r \leftarrow c_r/\gamma_r^{min}$ ▷ Eq. 7.18
8: $\beta_r \leftarrow \sqrt{c_r} * (1 - \rho_r)$ ▷ Eq. 7.6
9: $\kappa_r \leftarrow (K_r - c_r)/\sqrt{c_r}$ ▷ Eq. 7.7
10: $z_r \leftarrow p * v_r$ ▷ Eq. 7.8
11: $\tau_r \leftarrow \alpha(\beta_r/\sqrt{z_r}, p * \kappa_r/\sqrt{z_r})$ ▷ Eq. 7.11
12: $n_r \leftarrow c_r * \rho_r$ ▷ Eq. 7.17
13: $\theta_r \leftarrow 1 - \pi(\kappa_r, \tau_r, \beta_r, v_r, \rho_r, c_r)$ ▷ Eq. 7.20
14: $c_{r+1} \leftarrow c_r - n_r$ ▷ Eq. 7.15
15: **end for**
16: **return** θ_r
17: **end function**

O Algoritmo 3 é um método de busca simples, baseado em *hill-climbing*, para encontrar a capacidade mínima da nuvem necessária para cumprir os SLOs de disponibilidade de VMs e de taxa de admissão para cada classe. O procedimento é descrito da seguinte forma: iniciando com capacidade zero, o método iterativamente incrementa a capacidade da nuvem em ε unidades, e estima para cada cenário de capacidade a taxa de admissão a ser obtida por cada classe; a função então retorna a capacidade mínima que satisfaz os SLOs de taxa de admissão para todas as classes, com valor ε para o parâmetro de acurácia. Como a taxa de admissão de requisições cresce monotonicamente ao se aumentar a capacidade da nuvem e a implementação tem uma complexidade linear $O(R \times c)$, este método simples de *hill-climbing* foi eficiente e encontrou rapidamente soluções para o modelo. Outros métodos de busca mais

elaborados, como *simulated annealing* ou busca binária podem ser usados se necessário.

Algoritmo 3 Busca a menor capacidade necessária para cumprir SLOs de taxa de admissão.

1: **function** FINDBESTCAPACITY(ε, λ_r, μ_r, v_r, γ_r^{min}, θ_r^{min})
2: $c \leftarrow 0$
3: **repeat**
4: $c \leftarrow c + \varepsilon$
5: $\theta_r \leftarrow$ ESTIMATEQOS(c, λ_r, μ_r, v_r, γ_r^{min})
6: **until** $\theta_r \geq \theta_r^{min}$
7: **return** c
8: **end function**

7.4 Validação do modelo

Esta seção descreve como o modelo do planejamento de capacidade foi instanciado e apresenta os resultados da sua validação, comparando valores para métricas de QoS obtidos através de simulação com os valores estimados pelo modelo analítico proposto neste trabalho.

7.4.1 Instanciando o modelo

O simulador usado na validação do modelo do planejamento de capacidade foi o mesmo usado na avaliação do modelo de controle de admissão na Seção 6.6. Os dados de demanda e capacidade contidos nos rastros da Google descritos na Seção 6.3 e a metodologia descrita na Seção 6.5 também foram usados nestas simulações. Foi utilizada a heurística *pred-ets* de controle de admissão, que apresentou os melhores resultados na avaliação apresentada na Seção 6.6. Também foi usado o mesmo cenário base de simulação mostrado na Tabela 6.2, variando a partir dele parâmetros relativos à capacidade da nuvem e da intensidade da carga para explorar diferentes cenários com questões *what-if*.

O modelo do planejamento de capacidade descrito na Seção 7.3 foi instanciado a partir da implementação dos Algoritmos 2 e 3. Os valores dos parâmetros de entrada foram obtidos da seguinte forma.

Para as análises *what-if*, a capacidade média c da nuvem usada como entrada foi obtida a partir dos rastros da Google. A partir desse valor, aplicou-se um fator de capacidade para explorar diferentes cenários de capacidade da nuvem. Para planejar a capacidade da nuvem, o parâmetro c é a saída do modelo ao invés de sua entrada, denotando a melhor capacidade estimada pelo modelo que cumpre os SLOs para as classes, para um valor de acurácia ε.

A taxa média de chegada λ_r de requisições para cada classe r também é obtida dos rastros, calculando a média de CPU-cores requisitados para cada classe em cada intervalo de 5 minutos. Diferentes cenários de demanda são explorados ao se aplicar *load factors* aos valores base de taxa de chegada obtidos dos rastros.

O tempo médio de serviço μ_r de requisições para cada classe não é facilmente obtido dos rastros. Não há como saber o tempo de serviço para várias VMs que foram submetidas antes do início da coleta dos rastros, ou que não finalizaram até o fim da coleta. Estas situações acontecem principalmente para classe de produção (*prod*), com 26% das requisições caindo em um desses casos. Descartar requisições de VM que caem nesses casos se torna inviável pela quantidade significativa de ocorrências. Para lidar com esta situação, estimou-se o tempo médio de serviço para requisições da classe r com base na Lei de Little, com a seguinte fórmula:

$$\hat{\mu}_r = \frac{n_r^{\infty}}{\lambda_r}, \tag{7.21}$$

onde n_r^{∞} é a quantidade média de clientes da classe r observados no sistema para simulações com um número infinito de servidores; desta forma, não há tempo de espera na fila e o tempo médio de resposta observado é igual ao tempo médio de serviço, que é usado como um valor de entrada aproximado.

Para calcular o parâmetro de variação v_r usado como entrada do modelo para cada classe, é necessário obter valores do quadrado do coeficiente de variação (SCV) para o tempo entre chegadas ($c_{a,r}^2$) e tempo de serviço ($c_{s,r}^2$) para cada classe r. Esses valores também foram obtidos dos rastros, que foram calculados como a variância amostral dividida pelo quadrado da média amostral para cada uma das métricas [78].

Os SLOs de disponibilidade de VMs (γ_r^{min}) para cada classe r usados como entrada foram os mesmos definidos no cenário base de avaliação do controle de admissão (ver Tabela 6.2), com SLOs de disponibilidade de VM (γ_r^{min}) definidos como $\langle 100\%, 90\%, 50\% \rangle$ para as classes *prod*, *batch* e *free*, respectivamente. Os SLOs para a taxa de admissão (θ_r^{min})

para as classes *prod* e *batch* foram definidos como 99,9% e 99%, respectivamente. A classe *free* foi considerada oportunista com relação a esta métrica, não sendo definidos SLOs de taxa de admissão para esta classe. Desta forma, a demanda desta classe de menor prioridade não afeta decisões de planejamento de capacidade.

Como dito anteriormente, algumas VMs iniciaram sua execução antes do início da coleta dos rastros e não finalizaram até o fim da coleta; por suas longas durações, elas foram denominadas *VMs permanentes*. A capacidade total requisitada por VMs permanentes é significativa para a classe *prod* (24% da capacidade total requisitada), mas não é tão expressiva para as classes *batch* e *free* (menos de 1%). Assumindo que VMs permanentes são conhecidas, considerou-se que a capacidade disponível para VMs não permanentes é a sobra da capacidade deixada por VMs permanentes. A métrica de taxa de admissão é calculada apenas para VMs não permanentes. No planejamento de capacidade da nuvem, primeiro as VMs permanentes são ignoradas para se buscar a melhor capacidade apenas para VMs não permanentes. Em seguida, adiciona-se ao valor encontrado o total de capacidade requisitado por VMs permanentes, obtendo assim a estimativa da melhor capacidade para todas as requisições de VMs.

A métrica de *erro absoluto do modelo* (ξ_r), que compara a taxa de admissão observada em simulação com o valor estimado pelo modelo analítico, é calculada para cada classe r como:

$$\xi_r = |\theta_r - \hat{\theta}_r|, \tag{7.22}$$

onde θ_r é o valor da taxa de admissão para a classe r observado em simulação e $\hat{\theta}_r$ é o valor estimado pelo modelo.

7.4.2 Resultados

Nesta seção, os resultados da validação do modelo do planejamento de capacidade são apresentados. Primeiramente é apresentada uma análise *what-if*, que demonstra a eficácia do modelo ao estimar a taxa de admissão de requisições de VM para diferentes capacidades e carga da nuvem, comparando seus resultados com valores simulados. Em seguida, o modelo é usado para planejar a capacidade da nuvem, estimando a menor capacidade da nuvem necessária para cumprir SLOs para disponibilidade de VMs e taxa de admissão para todas as

classes.

Análise de questões *what-if*

O modelo de *what-if* proposto e analisado nesta seção busca estimar a taxa de admissão de requisições de VM para cada classe, gerada pelo método de controle de admissão baseado em quota para que sejam cumpridos SLOs de disponibilidade de VM, para diferentes cenários de demanda média para cada classe, capacidade média disponível na nuvem e conjunto de SLOs de disponibilidade de VM definidos para cada classe.

Nesta seção, duas questões *what-if* foram analisadas:

- E se a capacidade total da nuvem fosse alterada, qual seria a taxa de admissão de requisições de VM para cada classe?
- E se a carga de entrada da nuvem fosse alterada, qual seria a taxa de admissão de requisições de VM para cada classe?

O modelo analítico proposto na Seção 7.3.2 é usado para responder essas questões, comparando seus resultados com os valores obtidos através de simulações. O modelo só trata de taxa de admissão porque o método preditivo de controle de admissão já garante um alto cumprimento de SLOs de disponibilidade de VM, como demonstrado no Capítulo 6.

A Figura 7.1 compara as taxas de admissão obtidas em simulações (*simulation*, representado por triângulos verdes) com os valores estimados pelo modelo analítico (*model*, representado por círculos vermelhos), para diferentes classes e capacidades da nuvem. A média do erro absoluto do modelo para as classes *prod*, *batch* e *free* foi de $0{,}4\%$, $2{,}3\%$ e $7{,}8\%$, respectivamente, enquanto o valor máximo foi de $2{,}1\%$, $13{,}7\%$ e $23{,}4\%$, respectivamente.

Observe que o erro do modelo tende a ser maior para classes de menor prioridade. Isto acontece porque a capacidade disponível para classes de menor prioridade depende da demanda das classes de maior prioridade. Portanto, mais níveis de incerteza e variação são adicionados para prioridades menores, tornando mais difícil a captura de seu comportamento através do modelo analítico. Este não deve ser um fator crítico para a maioria dos casos, pois o cumprimento de SLOs para classes de menor prioridade tendem a ser mais flexíveis e menos importantes, além de gerar um impacto menor na receita do provedor. De toda forma,

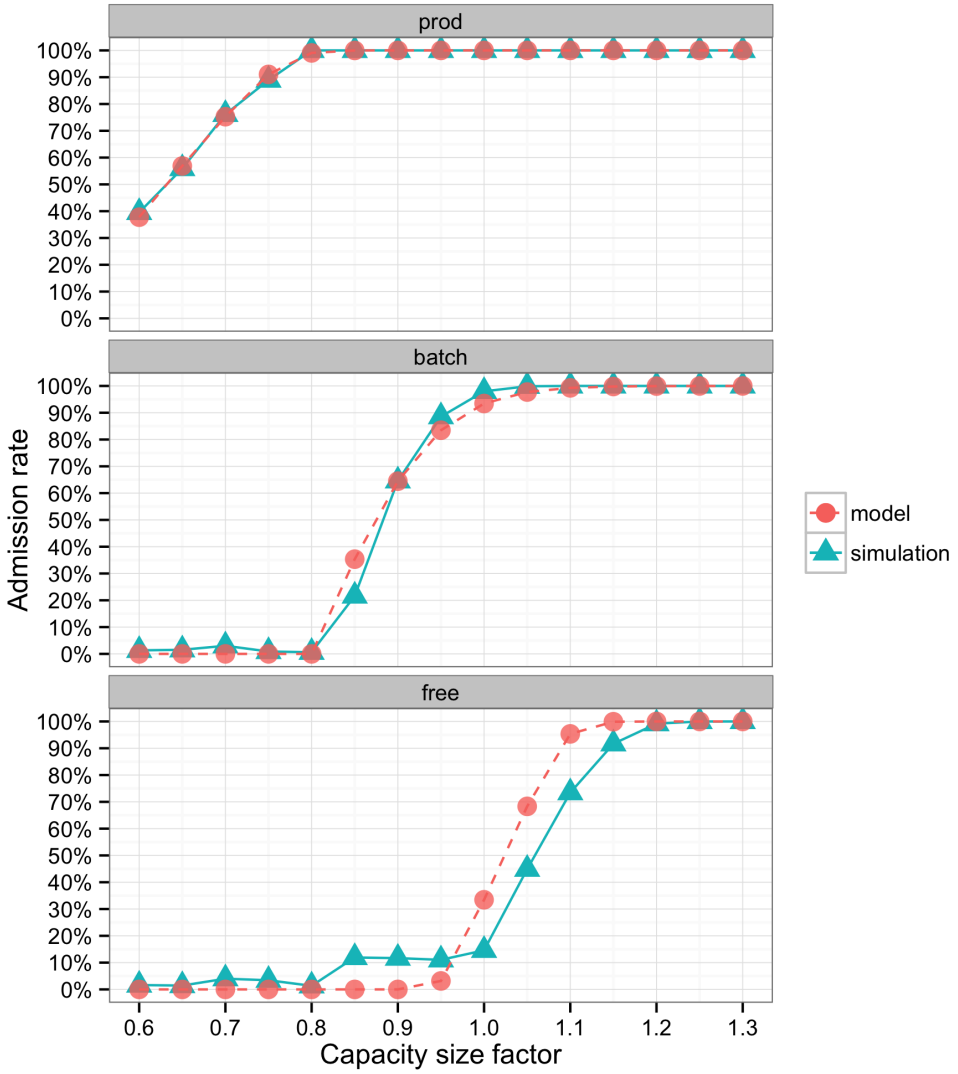

Figura 7.1: Taxas de admissão observadas em simulações comparadas com valores estimados pelo modelo de planejamento de capacidade, para diferentes classes e capacidades da nuvem.

para os casos em que o cumprimento de SLOs para classes de menor prioridade também sejam críticos, sugere-se o uso de uma margem adicional de segurança ao planejar a capacidade da nuvem para estas classes, para que o risco de violar SLOs para elas seja reduzido.

A Figura 7.2 compara a taxa de admissão obtida em simulações com os valores estimados pelo modelo, para diferentes classes e cargas de entrada da nuvem. Neste cenário, a média do erro absoluto do modelo para as classes *prod*, *batch* e *free* foi de $0{,}4\%$, $2{,}6\%$ e $7{,}1\%$, respectivamente, enquanto o valor máximo foi de $3{,}1\%$, $13{,}5\%$ e $20{,}5\%$, respectivamente. De forma similar à análise para diferentes capacidades, o erro do modelo também tende a ser maior para classes de menor prioridade. Observa-se também para classes de menor prio-

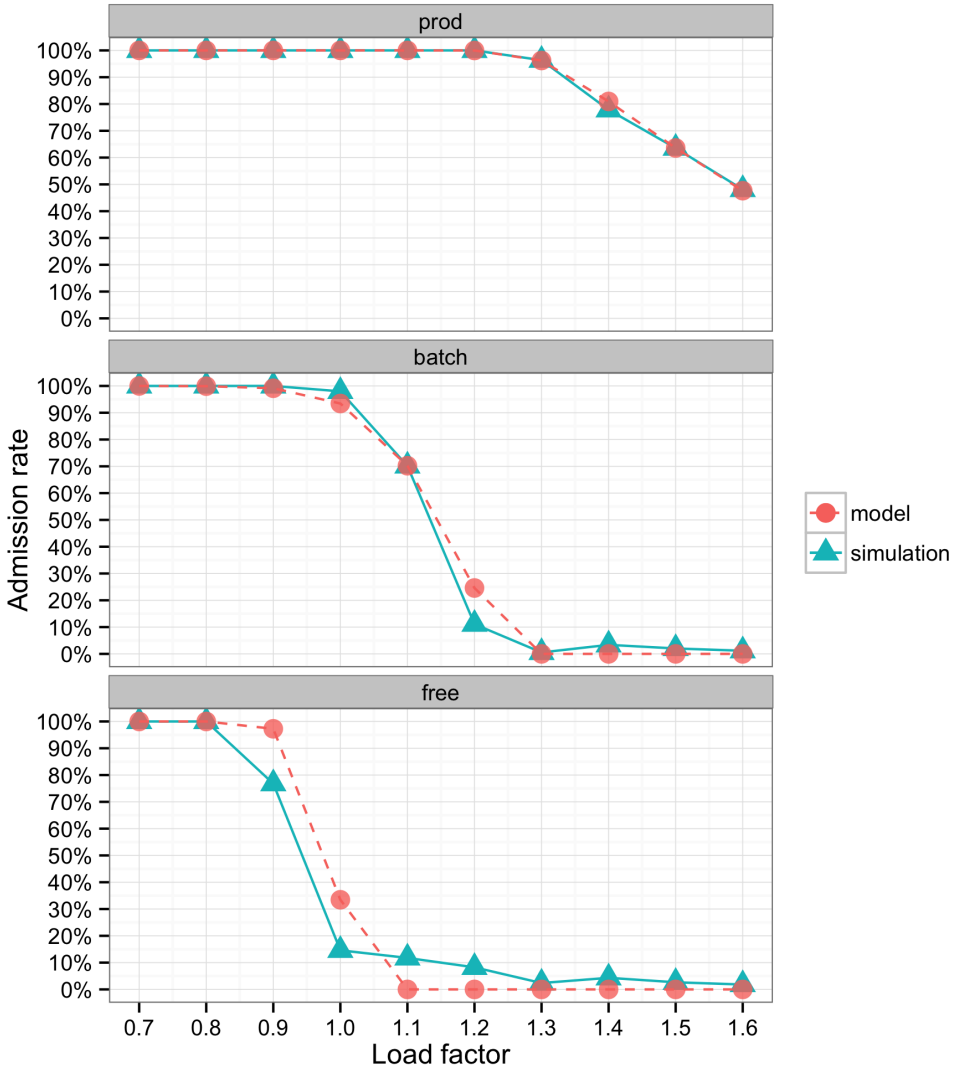

Figura 7.2: Taxas de admissão observadas em simulações comparada com valores estimados pelo modelo de planejamento de capacidade, para diferentes classes e intensidades de carga.

ridade que a taxa de admissão decai mais fortemente ao se aumentar a carga. Isto acontece porque o aumento do fator de carga tanto aumenta a carga para classes de menor prioridade como também reduz a capacidade disponível para elas, já que a demanda das classes de maior prioridade também aumenta, causando assim um maior impacto para as classes menos prioritárias.

Análise do planejamento de capacidade

O modelo analítico proposto também é usado para realizar um planejamento de capacidade da nuvem eficiente. O objetivo é encontrar a melhor capacidade para cada cenário de de-

manda – ou seja, o modelo busca estimar a capacidade mínima necessária para cumprir os SLOs de taxa de admissão oferecidos para cada classe.

Para validar o modelo, foram feitas simulações do ambiente de nuvem para diferentes intensidades de carga de trabalho, variando valores do parâmetro de fator de carga. Para identificar a melhor capacidade da nuvem para cada cenário de carga, foram simulados para cada fator de carga diferentes valores de capacidade da nuvem, variando o *capacity size factor* com valores no intervalo $[0{,}5, 2{,}5]$ com incrementos de $\varepsilon = 0{,}1$. Considerou-se como a melhor capacidade da nuvem para um certo fator de carga a capacidade mínima necessária para cumprir os SLOs de admissibilidade para todas as classes, entre as capacidades simuladas.

O modelo analítico do planejamento de capacidade também foi aplicado para estimar a melhor capacidade da nuvem para os diferentes cenários de carga, usando o Algoritmo 3. Em seguida, as estimativas geradas pelo modelo analítico do melhor fator de capacidade encontrado para cada cenário de carga foram comparadas aos resultados obtidos nas simulações. Como o modelo analítico executa muito mais rápido que as simulações, usou-se para o modelo uma melhor acurácia, realizando incrementos de $\varepsilon = 0{,}01$ a cada iteração do algoritmo, que explora diferentes fatores de capacidade da nuvem para buscar a capacidade mínima necessária para satisfazer os SLOs.

A Figura 7.3 compara os melhores fatores de capacidade obtidos em simulações com os valores estimados pelo modelo analítico do planejamento de capacidade, para diferentes cenários de carga. Observa-se que o valor observado para a melhor capacidade da nuvem cresce linearmente com o aumento da carga. O modelo do planejamento de capacidade proposto neste trabalho captura bem este comportamento para os cenários testados, estimando valores bem próximos à melhor capacidade observada nas simulações, necessárias para cumprir SLOs de disponibilidade de VMs e de taxa de admissão para todas as classes.

Embora o modelo analítico do planejamento de capacidade proposto neste trabalho se baseie em premissas que podem não acontecer na prática, constatou-se que o modelo é robusto e útil para responder questões *what-if*, estimando métricas de QoS para diferentes cenários da nuvem, e para planejar a capacidade necessária para cumprir SLOs para a taxa de admissão e de disponibilidade de VMs para múltiplas classes. As estimativas geradas pelo modelo analítico capturam bem o comportamento observado nas simulações ao variar cenários de

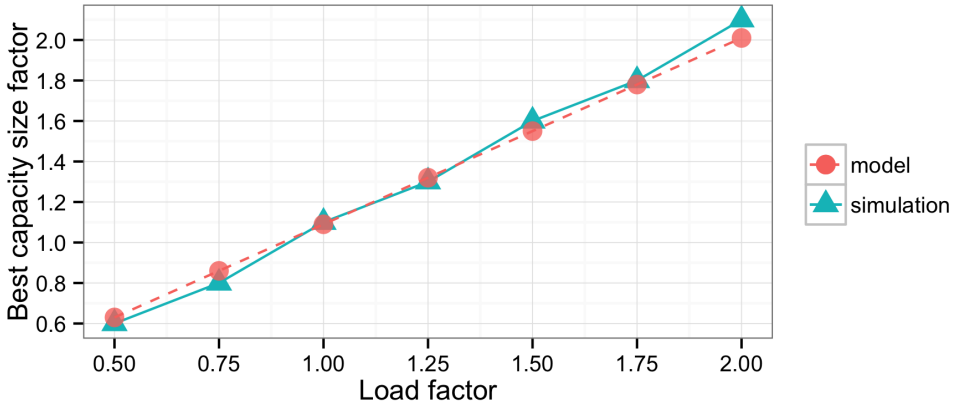

Figura 7.3: Capacidade mínima necessária para cumprir SLOs obtidas em simulações comparadas com os valores estimados pelo modelo do planejamento de capacidade da nuvem.

demanda e capacidade da nuvem. Observações similares sobre a robustez de modelos de filas foram feitos por Jeff Buzen quando ele desenvolveu a teoria da análise operacional e derivou os mesmos resultados da análise de filas através de análise estocástica, mas com premissas bem menos rígidas [19].

7.5 Conclusões do estudo

Mecanismos de controle de admissão se mostraram bastante eficazes para provedores de nuvem cumprirem SLOs de disponibilidade de VM ao oferecerem múltiplas classes de serviço. Porém, quando a capacidade não é super-provida, é necessário rejeitar muitas requisições para cumprir esses SLOs.

Neste capítulo, um modelo de planejamento de capacidade foi proposto para que os provedores também possam oferecer e cumprir metas para taxa de admissão de VMs, além das metas de disponibilidade de VM já endereçadas pelo controle de admissão. Foi elaborado um modelo analítico baseado em teoria das filas para estimar a taxa de admissão de requisições. Este modelo é usado para o planejamento de capacidade da nuvem, buscando encontrar a capacidade mínima necessária para cumprir SLOs de admissibilidade e disponibilidade de VM.

Os resultados mostraram que o modelo analítico é bastante útil e eficaz para encontrar a melhor capacidade da nuvem necessária para cumprir metas de SLO, capturando bem a

relação entre a capacidade de nuvem, sua demanda, os SLOs para as diferentes classes e o desempenho observado através das métricas de QoS.

Capítulo 8

Considerações finais e trabalhos futuros

Neste capítulo, são feitas as considerações finais do trabalho, descrevendo as suas conclusões, identificando as suas limitações e listando possíveis trabalhos futuros.

8.1 Conclusões

Nesta tese foi estudado o problema da gerência de recursos de nuvens computacionais com múltiplas classes de serviço. O principal objetivo do trabalho foi investigar a hipótese de que provedores de nuvem de IaaS podem aumentar sua utilização e receita ao oferecerem múltiplas classes de serviço, realizando uma gerência de recursos adequada para que as garantias de QoS oferecidas para as diferentes classes sejam cumpridas.

Verificou-se que esta hipótese é verdadeira para vários cenários de nuvem avaliados neste trabalho, para os quais foram demonstradas grandes vantagens de se oferecer múltiplas classes de serviço na nuvem. Porém, observou-se que para obter esses benefícios é necessário realizar uma gerência de recursos eficiente, de tal forma que as garantias de QoS para as diferentes classes sejam definidas adequadamente e cumpridas pelo provedor. Desta forma, esta tese também mostrou *como* provedores de IaaS podem realizar uma gerência de recursos adequada para múltiplas classes de serviço.

Em um primeiro estudo (no Capítulo 5), mostrou-se como uma nova classe de serviço pode ser introduzida em uma nuvem de IaaS existente, com base na capacidade excedente da nuvem. Um método baseado em predição foi proposto para planejar a capacidade a ser ofertada através desta nova classe e definir garantias de disponibilidade para a mesma. Os

resultados demonstraram a eficácia do método proposto e destacaram como o reaproveitamento adequado de sobras de recurso da nuvem através de uma nova classe, com garantias de QoS, pode aumentar a utilização e a receita do provedor da nuvem.

No estudo seguinte (no Capítulo 6), considerou-se a gerência de recursos para todas as diferentes classes oferecidas na nuvem, avançando o estudo anterior que realizou o planejamento de uma única classe baseada em sobras. Foi elaborado um modelo de controle de admissão baseado em predição, que permite ao provedor definir diferentes classes de serviço e cumprir SLOs de disponibilidade de VM para elas. Os resultados mostraram que o mecanismo de controle de admissão proposto consegue cumprir bem os SLOs de disponibilidade de VM definidos para as classes. Também foi mostrado como a definição de diferentes níveis de SLO impacta na utilização da nuvem, sugerindo que ao definir suas classes o provedor deve oferecer níveis de SLO diversificados (i.e., em uma larga faixa de valores), para que a capacidade da nuvem sejam bem preenchida e que os usuários tenham mais opções de serviço. Porém, observou-se que em cenários de alta contenção, o controle de admissão precisa rejeitar muitas requisições para cumprir os SLOs de disponibilidade de VM das que foram admitidas.

Por último, mas não menos importante, estudou-se (no Capítulo 6) como complementar o mecanismo de controle de admissão de tal forma que o provedor ofereça e cumpra garantias para a taxa de admissão de requisições de VM para cada classe, além dos SLOs de disponibilidade de VM já garantidos. Para isto, foi proposto um método de planejamento de capacidade, que busca encontrar a capacidade mínima necessária para cumprir as metas de taxa de admissão e disponibilidade de VM definidas para cada classe. Este método é baseado em um modelo analítico, elaborado usando aproximações de teoria das filas, que estima métricas de QoS da nuvem para diferentes cenários de demanda, capacidade e classes de serviço. Os resultados deste estudo mostraram que o modelo analítico é bastante eficaz para encontrar a melhor capacidade da nuvem, necessária para cumprir os SLOs das diferentes classes definidas. Para diferentes cenários, o modelo capturou bem o comportamento das métricas de QoS observadas em simulações, sendo uma ferramenta bastante útil para responder questões do tipo *what-if* para gerenciar nuvens de IaaS com múltiplas classes de serviço.

8.2 Limitações do trabalho

O escopo definido neste trabalho de tese resultou em algumas limitações do trabalho, com premissas que podem ser endereçadas em trabalhos futuros. As principais limitações do trabalho são as seguintes.

- Os mecanismos de gerência de recursos propostos foram aplicados considerando dados de CPU como o único tipo de recurso na tomada de decisões do provedor da nuvem. Apesar de CPU ser o recurso gargalo do sistema nos dados de nuvem analisados nesta tese, em outros ambientes pode-se ter outros tipos de recurso (e.g., memória e disco) que poderiam ser analisados e utilizados na gerência de recursos da nuvem, ou até mesmo usar a combinação de dados de múltiplos recursos para se ter uma gerência mais genérica.

- As requisições de VMs usadas para avaliar os mecanismos propostos são consideradas independentes. Ou seja, a gerência é feita para cada requisição individualmente, sem que a decisão para uma requisição impacte as outras. Na prática, pode-se ter requisições de VM que são dependentes e fazem parte de um conjunto de requisições (e.g., *jobs* MapReduce ou diferentes VMs que oferecem em conjunto um serviço *web*). Portanto, no futuro pode-se incorporar a dependência de VMs nos modelos propostos e avaliar as consequências desta relação entre diferentes requisições.

- As simulações realizadas nas avaliações dos métodos propostos nesta tese não realizam a alocação de VMs em máquinas físicas específicas, considerando apenas a capacidade agregada da nuvem ao tomar decisões de controle de admissão e planejamento de capacidade. Na prática, a alocação de VMs em máquinas físicas poderá gerar uma fragmentação de recursos, fazendo com que a capacidade útil disponível na nuvem seja menor do que a capacidade agregada considerada na avaliação. Por outro lado, provedores de nuvem também podem realizar *overbooking*, alocando VMs de tal forma que a soma dos recursos requisitados seja maior do que a capacidade disponível nas máquinas físicas. Portanto, a fragmentação tende a diminuir a capacidade total útil da nuvem, enquanto o *overbooking* tende a aumentar essa capacidade total útil. Estas questões não foram abordadas na tese; um modelo mais detalhado de simulação

poderia considerar tanto a fragmentação quanto o *overbooking* na nuvem para que a capacidade disponível para a alocação de VMs fosse simulada de forma mais precisa.

8.3 Trabalhos futuros

Como possíveis trabalhos futuros, as seguintes atividades de pesquisa são sugeridas:

1. Estudar modelos de negócio para as diferentes classes de serviço de acordo com seus SLOs, aprofundando a análise da receita do provedor com base em teorias econômicas. Vale mencionar que outros pesquisadores publicaram recentemente um artigo nesta direção [80], usando como base a nova classe *Economy* proposta nesta tese e publicada em um dos nossos artigos.

2. Elaborar e avaliar métodos de alocação de VMs em máquinas físicas (*VM placement*) que complementem de forma eficiente os modelos de planejamento de capacidade e controle de admissão propostos nesta tese, para que seja oferecida uma solução completa de gerência de recursos de nuvens computacionais com múltiplas classes de serviço.

3. Considerar nos modelos de gerência de recursos a utilização de VMs ao longo do tempo, ao invés de levar em consideração apenas a capacidade requisitada (ou tamanho da VM). Com isto, pode-se elaborar técnicas para o provedor realizar estratégias de *overbooking* de recursos, para atingir uma maior utilização da nuvem sem afetar o desempenho das aplicações, podendo estender os modelos propostos nesta tese para incorporar esta nova estratégia.

4. Expandir os modelos para considerar múltiplos tipos de recurso, ao invés de focar apenas em CPU ao tomar decisões de gerência de recursos. Apesar de CPU ser o recurso gargalo dos sistemas analisados nesta tese, considerar outros tipos de recurso nos modelos de gerência permite ao provedor planejar melhor a configuração da máquinas físicas da nuvem, de tal forma que não se tenha um desperdício tão grande de outros recursos como memória e disco, reduzindo os custos da infraestrutura.

5. Implementar os modelos propostos em sistemas de gerência de recursos de nuvem existentes (e.g., *OpenStack* [11] e Kubernetes [17]), para que eles sejam usados na prática e que mais experimentos sejam realizados para validar os modelos propostos.

6. Aplicar os modelos propostos nesta tese e analisar sua utilidade em outros contextos, como em nuvens computacionais de SaaS (*Software as a Service*), PaaS (*Platform as a Service*), assim como na gerência de *clusters* e grades computacionais, identificando os ajustes adequados e as diferentes métricas de QoS a serem oferecidas para os diferentes tipos de sistema.

Bibliografia

[1] Amazon EC2 – instance purchasing options. "https://aws.amazon.com/ec2/purchasing-options", April 2015.

[2] Amazon EC2 – service level agreement. "http://aws.amazon.com/ec2/sla", June 2015.

[3] Amazon EC2 FAQs. "http://aws.amazon.com/ec2/faqs", June 2015.

[4] Amazon Elastic Compute Cloud (EC2) – home page. "http://aws.amazon.com/ec2", June 2015.

[5] Google App Engine – home page. "https://developers.google.com/appengine", June 2015.

[6] Google Compute Engine - Preemptible Instances. "https://cloud.google.com/compute/docs/instances/preemptible", June 2015.

[7] Google Compute Engine (GCE) – home page. "https://cloud.google.com/products/compute-engine", June 2015.

[8] Google Compute Engine (GCE) – service level agreement. "https://developers.google.com/compute/sla", June 2015.

[9] Microsoft Azure – home page. "http://azure.microsoft.com", June 2015.

[10] Microsoft Azure – service level agreements. "http://azure.microsoft.com/en-us/support/legal/sla", June 2015.

[11] OpenStack – open source software for creating private and public clouds. "https://www.openstack.org/", December 2015.

[12] Rackspace Managed – service level agreement. "`http://www.rackspace.com/managed_hosting/support/servicelevels/managedsla`", June 2015.

[13] Salesforce – home page. "`http://www.salesforce.com`", June 2015.

[14] Omar Arif Abdul-Rahman and Kento Aida. Towards understanding the usage behavior of Google cloud users: the mice and elephants phenomenon. In *IEEE International Conference Cloud Computing Technology and Science (CloudCom)*, 2014.

[15] Luiz André Barroso and Urs Hölzle. Dealing with failures and repairs. In *The datacenter as a computer: An introduction to the design of warehouse-scale machines*, volume 4, pages 1–108. Morgan & Claypool Publishers, 2009.

[16] George Edward Pelham Box and Gwilym Jenkins. *Time Series Analysis, Forecasting and Control*. Holden-Day, Incorporated, 1990.

[17] Eric A. Brewer. Kubernetes and the path to cloud native. In *ACM Symposium on Cloud Computing (SoCC)*, 2015.

[18] Kenneth P. Burnham and David R. Anderson. *Model selection and multimodel inference: a practical information-theoretic approach*. Springer, 2 edition, 2002.

[19] J.P. Buzen. Fundamental operational laws of computer system performance. *Acta Informatica*, 7(2):167–182, 1976.

[20] David Candeia, Raquel Lopes, and Ricardo Araújo Santos. Planejamento de capacidade a longo prazo dirigido por métricas de negócio para aplicações SaaS. In *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, 2013.

[21] Marcus Carvalho and Francisco Brasileiro. Modelagem da carga de trabalho de grades computacionais baseada no comportamento dos usuários. In *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, 2012.

[22] Marcus Carvalho and Francisco Brasileiro. A user-based model of grid computing workloads. In *ACM/IEEE 13th International Conference on Grid Computing (GRID)*, 2012.

[23] Marcus Carvalho, Walfredo Cirne, Franciso Brasileiro, and John Wilkes. Long-term SLOs for reclaimed cloud computing resources. In *ACM Symposium on Cloud Computing (SoCC)*, 2014.

[24] Marcus Carvalho, Daniel Menasce, and Francisco Brasileiro. Prediction-based admission control for IaaS clouds with multiple service classes. In *IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 82–90, Vancouver, BC, Canada, November 2015.

[25] Marcus Carvalho, Daniel Menascé, and Francisco Brasileiro. Capacity planning and admission control of IaaS clouds with multiple service classes. Technical report, UFCG/DSC/LSD, January 2016. Disponível em: `http://www.lsd.ufcg.edu.br/~marcus/papers/UFCG-LSD-TR-2016-01-CARVALHO.pdf`.

[26] Marcus Carvalho, Renato Miceli, Paulo Ditarso Maciel Jr, Francisco Brasileiro, and Raquel Lopes. Predicting the quality of service of a peer-to-peer desktop grid. In *IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid)*, 2010.

[27] Emiliano Casalicchio, Daniel A. Menascé, and Arwa Aldhalaan. Autonomic resource provisioning in cloud systems with availability goals. In *ACM Cloud and Autonomic Computing Conference (CAC)*, 2013.

[28] Chris Chatfield. Forecasting. In *The analysis of time series: an introduction*. CRC Press, Boca Raton, FL, USA, 6th edition, 2004.

[29] L. Cherkasova and P. Phaal. Session-based admission control: a mechanism for peak load management of commercial web sites. *IEEE Transactions on Computers (TC)*, 2002.

[30] Walfredo Cirne and Eitan Frachtenberg. Web-scale job scheduling. In *International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP)*, volume 7698 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2012.

[31] Rostand Costa, Francisco Brasileiro, Guido Lemos, and Dênio Sousa. Analyzing the

impact of elasticity on the profit of cloud computing providers. *Future Generation Computer Systems (FGCS)*, 29(7):1777–1785, September 2013.

[32] Matheus Cunha, Nabor Mendonça, and Américo Sampaio. Cloud Crawler: Um ambiente programável para avaliar o desempenho de aplicações em nuvens de infraestrutura. In *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, 2013.

[33] Matheus Cunha, Nabor Mendonça, and Américo Sampaio. A declarative environment for automatic performance evaluation in IaaS clouds. In *IEEE International Conference on Cloud Computing (CLOUD)*, pages 285–292, 2013.

[34] Sheng Di, Derrick Kondo, and Walfredo Cirne. Characterization and comparison of cloud versus Grid workloads. In *IEEE International Conference on Cluster Computing (CLUSTER)*, pages 230–238, September 2012.

[35] Sheng Di, Derrick Kondo, and Walfredo Cirne. Host load prediction in a Google compute cloud with a Bayesian model. In *International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*, pages 21:1–21:11, Salt Lake City, UT, USA, November 2012.

[36] Javier Espadas, Arturo Molina, Guillermo Jiménez, Martín Molina, Raúl Ramírez, and David Concha. A tenant-based resource allocation model for scaling software-as-a-service applications over cloud computing infrastructures. *Future Generation Computer Systems (FGCS)*, 29(1):273–286, January 2013.

[37] D.G. Feitelson and A.M. Weil. Utilization and predictability in scheduling the ibm sp2 with backfilling. In *Merged International Parallel Processing Symposium and Symposium on Parallel and Distributed Processing*, 1998.

[38] Armando Fox, Rean Griffith, A Joseph, R Katz, A Konwinski, G Lee, D Patterson, A Rabkin, and I Stoica. Above the clouds: A berkeley view of cloud computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, 28:13, 2009.

[39] R. Ghosh, F. Longo, Ruofan Xia, V.K. Naik, and K.S. Trivedi. Stochastic model driven capacity planning for an infrastructure-as-a-service cloud. *IEEE Transactions on Services Computing (TSC)*, 7(4), 2014.

[40] Rahul Ghosh, Francesco Longo, Vijay K. Naik, and Kishor S. Trivedi. Modeling and performance analysis of large scale IaaS clouds. *Future Generation Computer Systems (FGCS)*, 29(5), 2013.

[41] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper. Capacity management and demand prediction for next generation data centers. In *IEEE International Conference on Web Services (ICWS)*, pages 43–50, July 2007.

[42] Zhenhuan Gong, Xiaohui Gu, and John Wilkes. PRESS: PRedictive Elastic ReSource Scaling for cloud systems. In *International Conference on Network and Service Management (CNSM)*, pages 9–16, 2010.

[43] R. J. Hyndman and G. Athanasopoulos. Exponential smoothing. In *Forecast: principles and practice*. OTexts, 2013.

[44] Rob J. Hyndman and Yeasmin Khandakar. Automatic time series forecasting: the forecast package for R. *Journal of Statistical Software*, 27(3):1–22, July 2008.

[45] Rob J. Hyndman, Anne B. Koehler, J. Keith Ord, and Ralph D. Snyder. Prediction intervals for exponential smoothing using two new classes of state space models. *Journal of Forecasting*, 24(1):17–37, 2005.

[46] Gartner Inc. Forecast: It services, 2011-2017, 4q13 update. "`https://www.gartner.com/doc/2637515/forecast-it-services--q`", December 2013.

[47] Gartner Inc. Gartner says cloud computing will become the bulk of new it spend by 2016. "`http://www.gartner.com/newsroom/id/2613015`", October 2013.

[48] Harold Kerzner. *Project Management: A Systems Approach to Planning, Scheduling, and Controlling*. Wiley, 11 edition, 2013.

[49] Mukil Kesavan, Irfan Ahmad, Orran Krieger, Ravi Soundararajan, Ada Gavrilovska, and Karsten Schwan. Practical compute capacity management for virtualized datacenters. *IEEE Transactions on Cloud Computing (TCC)*, 1(1):1, 2013.

[50] H. Khazaei, J. Misic, and V.B. Misic. Performance analysis of cloud computing centers using M/G/m/m+r queuing systems. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 23(5), 2012.

[51] H. Khazaei, J. Misic, and V.B. Misic. A fine-grained performance model of cloud computing centers. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 24(11), 2013.

[52] H. Khazaei, J. Misic, and V.B. Misic. Performance of cloud centers with high degree of virtualization under batch task arrivals. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 24(12), 2013.

[53] Paulo Ditarso Maciel, Francisco Brasileiro, Raquel Lopes, Marcus Carvalho, and Miranda Mowbray. Evaluating the impact of planning long-term contracts on the management of a hybrid it infrastructure. In *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 89–96. IEEE, 2011.

[54] Paulo Ditarso Maciel, Francisco Brasileiro, Ricardo Araújo Santos, David Candeia, Raquel Lopes, Marcus Carvalho, Renato Miceli, Nazareno Andrade, and Miranda Mowbray. Business-driven short-term management of a hybrid it infrastructure. *Journal of Parallel and Distributed Computing (JPDC)*, 72(2):106–119, 2012.

[55] Dan C Marinescu. Cloud resource management and scheduling. In *Cloud Computing: Theory and Practice*. Morgan Kaufmann Publishers Inc., 2013.

[56] Paul Marshall, Kate Keahey, and Timothy Freeman. Improving utilization of infrastructure clouds. In *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pages 205–214, 2011.

[57] Daniel Menascé and Shouvik Bardhan. Epochs: Trace-driven analytical modeling of job execution times. Technical report, George Mason University, 2014.

[58] Daniel A. Menasce, Lawrence Dowdy, and Virgilio A.F. Almeida. *Performance by Design: Computer Capacity Planning By Example*. Prentice Hall PTR, 2004.

[59] Xiaoqiao Meng, Canturk Isci, Jeffrey Kephart, Li Zhang, Eric Bouillet, and Dimitrios Pendarakis. Efficient resource provisioning in compute clouds via VM multiplexing. In *IEEE/ACM International Conference on Autonomic Computing and Communications (ICAC)*, pages 11–20, Washington, DC, USA, 2010.

[60] Asit K. Mishra, Joseph L. Hellerstein, Walfredo Cirne, and Chita R. Das. Towards characterizing cloud backend workloads: insights from Google compute clusters. *SIGMETRICS Performance Evaluation Review*, 37(4):34–41, March 2010.

[61] Fabio Jorge Almeida Morais, Francisco Vilar Brasileiro, Raquel Vigolvino Lopes, Ricardo Araújo Santos, Augusto Macedo, Wade Satterfield, and Leandro Rosa. Um arcabouço para provisionamento automático de recursos em provedores de IaaS independente do tipo de aplicação. In *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, 2013.

[62] Fabio Jorge Almeida Morais, Francisco Vilar Brasileiro, Raquel Vigolvino Lopes, Ricardo Araújo Santos, Wade Satterfield, and Leandro Rosa. Autoflex: Service agnostic auto-scaling framework for IaaS deployment models. In *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pages 42–49, 2013.

[63] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2015.

[64] Charles Reiss, Alexey Tumanov, Gregory R. Ganger, Randy H. Katz, and Michael A. Kozuch. Heterogeneity and dynamicity of clouds at scale: Google trace analysis. In *ACM Symposium on Cloud Computing (SoCC)*, pages 7:1–7:13, San Jose, CA, USA, 2012.

[65] Charles Reiss, John Wilkes, and Joseph L. Hellerstein. Google cluster-usage traces: format + schema. Technical report, Google Inc., November 2011. Posted at URL `http://code.google.com/p/googleclusterdata/wiki/TraceVersion2`.

[66] Rami Rosen. Linux containers and the future cloud. *Linux Journal*, 2014(240), April 2014.

[67] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

[68] Lılia Rodrigues Sampaio and Raquel Vigolvino Lopes. Towards practical auto scaling of user facing applications. In *IEEE Latin America Conference on Cloud Computing and Communications (LATINCLOUD)*, pages 60–65, 2012.

[69] Zhiming Shen, Sethuraman Subbiah, Xiaohui Gu, and John Wilkes. CloudScale: elastic resource scaling for multi-tenant cloud systems. In Jeffrey S. Chase and Amr El Abbadi, editors, *ACM Symposium on Cloud Computing (SoCC)*, page 5, 2011.

[70] Jim Smith and Ravi Nair. *Virtual machines: versatile platforms for systems and processes*. Elsevier, 2005.

[71] M. Unuvar, Y.N. Doganata, A.N. Tantawi, and M. Steinder. Cloud overbooking through stochastic admission controller. In *International Conference Network and Service Management (CNSM)*, 2014.

[72] Merve Unuvar, Yurdaer N. Doganata, and Asser N. Tantawi. Configuring cloud admission policies under dynamic demand. In *IEEE International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2013.

[73] Luis M Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A break in the clouds: towards a cloud definition. *ACM SIGCOMM Computer Communication Review*, 39(1):50–55, 2008.

[74] Abhishek Verma, Luis Pedrosa, Madhukar R. Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. Large-scale cluster management at Google with Borg. In *European Conference Computer Systems (EuroSys)*, 2015.

[75] Carl A. Waldspurger. Memory resource management in VMware ESX Server. In *OS Design and Implementation (OSDI'02)*, pages 181–194, December 2002.

[76] Ronald E. Walpole, Raymond H. Myers, Sharon L. Myers, and Keying Ye. *Probability & statistics for engineers and scientists*. Pearson Education, 9th edition, 2012.

[77] Aaron Weiss. Computing in the clouds. *Networker*, 11(4), 2007.

[78] Ward Whitt. A diffusion approximation for the G/GI/n/m queue. *Operations Research*, 52:922–941, 2004.

[79] John Wilkes, Jeffrey Mogul, and Jaap Suermondt. Utilification. In *ACM SIGOPS European workshop*, page 13, 2004.

[80] Jie Xu and Chenbo Zhu. Optimal pricing and capacity planning of a new economy cloud computing service class. In *Cloud and Autonomic Computing (ICCAC), 2015 International Conference on*, pages 149–157, Sept 2015.

[81] Qi Zhang, Joseph Hellerstein, and Raouf Boutaba. Characterizing task usage shapes in Google compute clusters. In *Workshop on Large Scale Distributed Systems and Middleware (LADIS)*, Seattle, WA, USA, September 2011.