Ítalo Fernando Scotá Cunha

Gerenciamento de Capacidade de Infra-Estruturas Multicamadas com Restrições de Energia e Ataques de Segurança

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientadora:

Jussara Marques de Almeida

Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Belo Horizonte

23 de novembro de 2007

Agradecimentos

Agradeço em primeiro lugar aos meus pais. À minha mãe, com quem minha dívida vem acumulando a 24 anos, pela atenção, carinho e as várias lições aprendidas. Agradeço ao meu pai, que no decorrer da minha vida me moldou no que sou hoje, servindo de modelo. Agradeço especialmente a compreensão e o suporte que eles me deram ao longo dos meus estudos.

Devo muito ao galerão do VOD! Alex Brogues, criando o melhor ambiente de trabalho que há! Aprendi muito com o Marcus Rocha e Marcelo Maia, quando trabalhamos juntos em outro projeto. Itamar Viana e João Palotti me ajudaram na reta final deste trabalho. Obrigado a todos!

Agradeço aos meus tios Pedro Scottá, que sempre me dedicou atenção especial, e A. Rocha, de quem o nome já fala tudo! Agradeço também aos meus tios Eduardo, Ricardo e Berenice pelo apoio desde que cheguei em Belo Horizonte. Agradeço a todo meu clã de primos que sabem fazer os melhores momentos. Por último, agradeço a minha mãe adotiva, Marlena da Silva, pelo continuado esforço na manutenção do meu bem estar.

Agradeço a todos os professores que dedicaram o seu tempo a me ensinar. Agradeço especialmente aos Professores Geraldo Leão e Paulo Faria, cujas contribuições à minha formação me abriram muitas portas. À Jussara Almeida meus sinceros elogios e agradecimentos pela sua dedicação e orientação, que me propiciaram ganhar XP ao dobro da velocidade normal nos últimos 5 anos, contribuição inestimável à minha formação.

Agradeço aos meus amigos, que me permitiram conhecer distintas qualidades do ser humano. Tanatron e Don, de personalidade calma e com quem discutir qualquer assunto é certeza de aprender alguma coisa. Krost, um dos raros samurais modernos, e Aishameriane, minha estrelinha. Agradeço também a minha irmã, que simplesmente por crescer me ensinou muita coisa. Obrigado Mariana pelo amor.

Por fim, agradeço a todos que me ajudaram nesta caminhada.

gg!

Resumo

Ao longo dos últimos anos, um número crescente de empresas vem utilizando terceirização de capacidade computacional como uma abordagem financeiramente atrativa para hospedar seus serviços. Esta dissertação considera o cenário onde diferentes serviços Web transacionais de terceiros são hospedados numa infra-estrutura compartilhada. O foco é no gerenciamento de capacidade dessas infra-estruturas, o que consiste em alocar os recursos disponíveis entre os serviços hospedados de forma a maximizar os objetivos de negócio, isto é, o lucro total, do provedor.

O surgimento de novas demandas por parte dos clientes impõe desafios operacionais inéditos ao problema de gerenciamento de capacidade dessas infra-estruturas compartilhadas. Além disso, serviços Web atuais demandam plataformas de hospedagem multicamadas, por exemplo, camadas de apresentação, aplicação e banco de dados. O gerenciamento de capacidade de uma infra-estrutura multicamadas fica ainda mais desafiador devido à maior complexidade do sistema. Ao longo dos últimos anos, os esforços da área de gerenciamento têm tradicionalmente focado apenas em objetivos de desempenho. Porém, a qualidade do serviço provido às aplicações hospedadas, e, por fim, o lucro obtido pelo provedor, dependem também de outros aspectos, como segurança e restrições de energia.

Esta dissertação estende um arcabouço prévio baseado na modelagem da infra-estrutura em uma única camada, para capturar, de forma mais precisa, os componentes principais de infra-estruturas multicamadas. A solução apresentada captura ainda, num arcabouço unificado, os compromissos chave de desempenho e custo que surgem quando operando sob ataques de segurança e restrições de energia. As principais contribuições são: (1) um modelo de desempenho multicamadas mais preciso, que captura heterogeneidade de aplicações e o paralelismo inerente a plataformas multicamadas, (2) um modelo de negócio flexível, com contratos adaptativos para períodos de ataques de segurança ou restrições de energia e (3) modelos de otimização muito mais complexos, responsáveis por calcular as decisões de alocação de capacidade, combinando os modelos de desempenho e negócio.

A nova abordagem é avaliada através de simulação com cargas sintéticas e cargas realistas, em vários cenários. Os resultados mostram que nova solução multicamadas é significativamente mais eficaz, em termos do lucro obtido pelo provedor, do que a abordagem anterior sobre a qual foi construída, que usa um modelo de desempenho de camada única. Ela também supera significativamente alocação estática de capacidade para cargas pesadas e heterogêneas. O arcabouço consegue minimizar os impactos de ataques de segurança e restrições de energia através da alocação consciente de capacidade, por exemplo minimizando os recursos desperdiçados com requisições ilegítimas. Por último, contratos adaptativos podem ser utilizados para encontrar um compromisso entre os interesses do provedor e de seus clientes.

Abstract

Along the last years, an increasing number of enterprises have turned to computational capacity outsourcing as a financially attractive approach to host their online services. This dissertation considers the scenario where many different third-party transactional Web services are hosted in a shared infrastructure. The focus is on the capacity management of these infrastructures, which consist in finding the most cost-effective way to allocate the available resources, aiming at maximizing the provider's business objective, i.e., total profit.

The emergence of new demands by the clients impose additional operational problems to the capacity management of theses shared infrastructures. Furthermore, current Web services demand multi-tier hosting platforms, e.g., presentation, application and database tiers. The task of capacity management of such a multi-tier infrastructure is further challenging as a result of the higher system complexity. Capacity management of hosting infrastructures have traditionally focused on performance objectives. However, quality of service provided to the hosted applications and, ultimately, the provider's profit, depend also on other aspects, such as security attacks and energy constraints.

This dissertation extends a previous framework based on a single-tier infrastructure model to capture, more accurately, the main trade-offs of multi-tier infrastructures. The solution presented also captures, in an unified framework, the key performance and cost trade-offs that arise when operating under security attacks and energy constraints. The main contributions are: (1) a more accurate multi-tier performance model, that captures application-specific bottlenecks and the inherent parallelism of multi-tier platforms, (2) a flexible business model, with adaptive contracts for periods when security attacks or energy constraints are in effect, and (3) a new and much more complex optimization model, combining both performance and business models, responsible for calculating the capacity allocation decisions.

The new approach is evaluated through simulation experiments with synthetic and realistic workloads in various scenarios. The results show that the new multi-tier solution is far more cost-effective, in terms of the provider's attained profit, than the previous approach, which it is built from, that employs a single-tier performance model. It also significantly outperforms static capacity allocation for heavy and heterogeneous workloads. The framework can minimize the impacts of security attacks and energy constraints through conscious capacity allocation decisions. Finally, adaptive contracts can be used to find a compromise between the provider's and its clients' interests.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

Com o crescimento exponencial da popularidade da Internet nos últimos anos, vimos surgir uma grande quantidade de diferentes serviços Web como correio eletrônico, serviços bancários, portais especializados, lojas virtuais, blogs e outros. Além disso, serviços Web se tornaram uma solução popular para empresas oferecerem seus serviços tradicionais para usuários *online* e implantar operações de negócio, antes locais, ao redor do globo.

Um número crescente de empresas vem utilizando terceirização de capacidade computacional como uma abordagem financeiramente atrativa para hospedar seus serviços cada vez mais populares [89]. A terceirização de capacidade está revolucionando como serviços computacionais são consolidados [110]. As principais vantagens envolvidas são a minimização de riscos no dimensionamento inicial e custos relacionados à manutenção da infra-estrutura, bem como menor custo de operação devido à melhor utilização dos recursos resultante do compartilhamento da infra-estrutura terceirizada pelas várias aplicações hospedadas.

Nesse cenário, objetivando a satisfação de seus usuários, provedores de serviço, doravante chamados de *clientes*, demandam dos provedores de infra-estrutura que uma fração significativa das requisições de seus usuários sejam servidas com métricas de qualidade que atendam requisitos específicos. Assim, clientes assinam contratos de Acordo de Nível de Serviço (*Service Level Agreement*) (SLA), que especificam os custos da hospedagem e os requisitos de qualidade da sua aplicação [21], com um provedor que hospeda uma diversidade de serviços Web. Por outro lado, o objetivo do provedor de infra-estrutura, ou simplesmente o *provedor*, é encontrar a estratégia com a melhor relação custo-benefício para gerenciar seus recursos disponíveis, compartilhados por todas as aplicações hospedadas, satisfazendo os requisitos de qualidade de serviço estabelecido com seus clientes.

Esta dissertação considera o cenário onde vários diferentes serviços Web transa-

cionais de terceiros são hospedados em uma infra-estrutura compartilhada. O foco é no gerenciamento de capacidade dessas infra-estruturas, o que consiste em explorar os recursos disponíveis da melhor forma possível com o objetivo de maximizar os objetivos de negócio do provedor.

1.1 Desafios

O surgimento de novas demandas por parte dos clientes impõe desafios operacionais inéditos à tarefa de gerenciamento de capacidade dessas infra-estruturas compartilhadas. Os clientes expressam seus requisitos em função de duas medidas de desempenho do sistema. Primeiro, clientes demandam que os provedores forneçam garantias sobre a taxa de processamento, em requisições por segundo, atingida pelas suas aplicações. Segundo, essa taxa de processamento deve atender requisitos sobre o tempo de resposta, isto é, o tempo que a infra-estrutura leva para processar cada uma das requisições.

Para ser competitivo, porém, não basta apenas atender requisitos sobre a média do tempo de resposta, como feito tradicionalmente. Clientes cobram garantias sobre o atendimento de requisitos sobre a cauda da distribuição do tempo de resposta [60], significando que há um limite superior para a probabilidade do tempo de resposta de uma requisição violar seu requisito de qualidade. Por último, mas não menos importante, há um recente aumento de interesse em contratos de serviço nos quais o cliente paga somente pelos recursos que foram efetivamente utilizados [110], forçando os provedores de serviço a reverem seus contratos tradicionais e propor novos esquemas de contabilização mais flexíveis.

Esses requisitos de desempenho têm impacto determinante na renda obtida pelo provedor pela hospedagem das aplicações [21]. Conseqüentemente, o gerenciamento de capacidade deve ser dirigido por um modelo de negócio no qual disputas por capacidade entre aplicações hospedadas são resolvidas sob a luz do objetivo financeiro do provedor, ao invés de simplesmente atender requisitos de desempenho como taxa de processamento e tempo de resposta. Esses desafios implicam na necessidade de contratos e modelos de negócio mais complexos que, por fim, levam a soluções de gerenciamento de capacidade mais sofisticadas.

Além dos fatores discutidos acima, a heterogeneidade das aplicações e flutuações diárias tipicamente altas da carga de trabalho [12, 22, 102], que mudam drasticamente

a necessidade por recursos das aplicações durante sua operação, não podem ser efetivamente acomodadas com estratégias de gerenciamento de capacidade estáticas tradicionais. Isso evidencia a necessidade de estratégias autônomas de gerenciamento que adaptam o sistema em tempo real em resposta a essas mudanças.

Nesse cenário, a virtualização de recursos [14,16,108] pode contribuir para a construção de um ambiente com melhor relação custo-benefício, através da criação de máquinas virtuais sobre a infra-estrutura física, cada uma composta de um conjunto de instâncias virtuais dos recursos físicos e destinadas a servir uma aplicação específica. A virtualização facilita a implantação e manutenção de aplicações, provê isolamento de capacidade entre elas e, por último, inclui mecanismos para controle da capacidade alocada a cada máquina virtual.

Porém, serviços Web atuais demandam complexas e heterogêneas plataformas de serviço multicamadas. Serviços típicos incluem uma camada de apresentação, compostas de servidores HTTP [33,68], uma camada de aplicação [27,69] e possivelmente uma camada de persistência, composta de um banco de dados [1,44,77]. O gerenciamento de capacidade de uma infra-estrutura compartilhada multicamadas fica particularmente desafiador devido a várias razões. O gerenciamento integrado desses complexos sistemas demanda modelos analíticos que combinam diferentes técnicas para representar os vários fatores que impactam o desempenho do sistema, sem que sua complexidade comprometa sua implantação em soluções reais.

Ao longo dos últimos anos, pesquisa em gerenciamento de capacidade de infra-estruturas tem enfatizado o desenvolvimento de técnicas para atingir objetivos de desempenho. Porém, a qualidade de serviço provida aos clientes e, conseqüentemente, o lucro obtido pelo provedor de infra-estrutura dependem de outros aspectos.

Um desses aspectos é o impacto de ataques de segurança, que visam comprometer a qualidade do serviço recebido pelos usuários, no objetivo de negócio do provedor. Durante um ataque, requisições ilegítimas admitidas no sistema consomem recursos disponíveis, impactando o desempenho de requisições legítimas concorrentes da mesma aplicação. Por outro lado, o provedor também é penalizado pois este não capitaliza sobre os recursos alocados para acomodar o tráfego ilegítimo.

Outros aspectos que adicionam desafios e impactam a tarefa de gerenciamento de capacidade são custos e restrições de energia. Esses aspectos são cada vez mais importantes. O instituto Gartner de consultoria reporta que atualmente até 10% do orçamento de empresas são gastos com despesas de energia, e espera que esta fração

aumente ainda mais [41]. Nesse cenário, novos compromissos surgem para lidar com custos (de energia) e benefícios (em requisições processadas) da alocação de capacidade, em particular quando ocorrem restrições de energia, planejadas ou não, que reduzem a capacidade disponível para atender requisições.

1.2 Estado da Arte

Vários trabalhos na literatura abordam o problema de gerenciamento de capacidade de infra-estruturas compartilhadas [10, 21, 22, 24, 31, 59–61, 67, 81, 86, 87, 102, 111], utilizando diferentes abordagens e técnicas. Esses trabalhos abordam os desafios relacionados aos requisitos de qualidade de serviço – taxa de processamento e tempo de resposta – e o objetivo de negócio do provedor. Em [3, 4] foi proposto um arcabouço de gerenciamento autônomo de capacidade que aborda esses pontos. Esse arcabouço combina um modelo de negócio derivado de um contrato de serviço de dois níveis e um modelo de desempenho que provê garantias sobre a cauda da distribuição do tempo de resposta em um modelo de otimização, que decide como alocar dinamicamente a capacidade disponível entre as aplicações hospedadas visando maximizar o objetivo de negócio do provedor.

Os trabalhos acima consideram apenas infra-estruturas e serviços Web com uma camada, representando a fração dos recursos físicos destinados a cada aplicação como um recurso único. Essa é uma representação simplificada do sistema, especialmente para aplicações heterogêneas executando em plataformas multicamadas. Assim, ela pode levar a imprecisões significativas, em última instância impactando a eficácia da solução. Apenas uma quantidade limitada de trabalhos recentes [99, 100] aborda o problema de gerenciamento de capacidade de infra-estruturas multicamadas. Porém, esses trabalhos fazem simplificações na modelagem da infra-estrutura, provêm garantias de qualidade apenas sobre a média dos tempos de resposta e não consideram objetivos de negócio do provedor.

Em relação aos ataques de segurança, a grande maioria dos esforços prévios objetiva melhorar a detecção de ataques e a robustez do sistema alvo e são desconexos da tarefa de gerenciamento de capacidade. Apesar da miríade de técnicas de detecção [13, 15, 39, 62, 72, 85, 91, 104] e defesa [7, 37, 38, 51, 53, 73, 92, 105, 106] contra ataques disponíveis, relatórios recentes indicam que ataques de segurança, especialmente os que atingem aplicações específicas [30, 51, 71], ainda causam perdas financeiras de

grande magnitude [38,42,47,58,95].

Por último, há na literatura um número reduzido de propostas de mecanismos de gerenciamento de capacidade que introduzem custos relacionados a energia [10, 22,23]. Todavia, essas propostas consideram cenários simplificados, combinando garantias sobre as médias das métricas de desempenho e plataformas com apenas uma camada, o que pode não capturar com precisão cargas de trabalho variáveis ou aplicações heterogêneas.

1.3 Objetivos

Motivados pelos desafios apresentados e o atual estado da arte das soluções de gerenciamento de capacidade, os principais objetivos desta dissertação são:

- Criar um arcabouço autônomo de gerenciamento de capacidade para infra-estruturas multicamadas. Esse arcabouço precisa combinar um modelo de desempenho preciso e um modelo de negócio flexível em um modelo de otimização, responsável por encontrar a configuração ótima da infra-estrutura.
- Capturar e compreender os principais compromissos nos objetivos de negócio do provedor e de seus clientes introduzidos por ataques de segurança nessas infra-estruturas multicamadas, para minimizar o impacto destes ataques por meio da alocação consciente de capacidade.
- Adicionar custos de energia no modelo de negócio utilizado pelo arcabouço. A minimização de consumo de energia, por meio da melhor utilização dos recursos da infra-estrutura, pode levar a ganhos consideráveis de rentabilidade.
- Considerar restrições de energia no modelo de otimização. Durante períodos nos quais o suprimento de energia está reduzido, o arcabouço deve desligar a parte menos crítica da infra-estrutura e priorizar aplicações que resultam em maior rentabilidade, de acordo com os objetivos de negócio do provedor.

1.4 Contribuições

A principal contribuição deste trabalho é um arcabouço de gerenciamento de capacidade mais rico e preciso que seus predecessores, construído a partir do arcabouço

apresentado em [3, 4]. Ele herda características desejáveis como decisões de alocação que levam em consideração o objetivo de negócio do provedor, ao contrário de apenas requisitos de desempenho, e garantias sobre a cauda da distribuição do tempo de resposta. Além disso, ele é aplicável a infra-estruturas multicamadas e vai além dos objetivos tradicionais de desempenho, capturando compromissos entre custo e rentabilidade introduzidos por ataques de segurança, custos e restrições de energia. O novo arcabouço utiliza novos modelos de negócio, desempenho e otimização, onde também temos contribuições significativas:

- O novo modelo de desempenho, baseado em teoria de filas, captura os principais fatores que impactam o desempenho de infra-estruturas de hospedagem multicamadas. Apesar da estimativa da cauda da distribuição do tempo de resposta ser significativamente mais desafiadora nesse cenário, nosso modelo é capaz de fazê-la com maior precisão.

- Os modelos de negócio e otimização foram estendidos para capturar os aspectos primários relacionados a ataques de segurança, custos e restrições de energia. Foram implementadas estratégias alternativas baseadas em contratos de serviço adaptativos que podem ser usados pelos clientes para aliviar o impacto negativo desses aspectos na qualidade de serviço provida aos usuários de sua aplicação.

- O complexo modelo de otimização não-linear, que combina os modelos de desempenho e negócio para encontrar a melhor configuração da infra-estrutura, não tem solução trivial. São discutidos cada um dos desafios existentes para a solução do modelo bem como diferentes estratégias para garantir sua convergência e qualidade das soluções encontradas.

É apresentada uma extensa avaliação do nosso arcabouço de gerenciamento de capacidade por meio de simulações. É coberta uma grande quantidade de cenários de interesse dentro do espaço de projeto, utilizando cargas sintéticas para evidenciar e discutir os principais compromissos e cargas realistas para validá-los. As principais conclusões incluem:

- O arcabouço multicamadas escala para cenários práticos, com o tempo de solução do modelo de otimização crescendo com uma fração pequena do número de aplicações hospedadas.

- Através da comparação do novo arcabouço com a solução prévia de camada única no qual é baseado [3,4] e alocação estática de capacidade, são mostradas suas vantagens e discutidos os principais compromissos existentes no gerenciamento de capacidade de infra-estruturas multicamadas.

- São mostrados como os impactos de ataques de segurança e restrições de energia podem ser minimizados por meio da alocação consciente de capacidade, por exemplo minimizando os recursos desperdiçados com requisições ilegítimas, e como contratos adaptativos podem ser utilizados para encontrar um equilíbrio entre os interesses do provedor e seus clientes.

- De forma complementar, é avaliada a sensibilidade do novo arcabouço a uma de suas premissas centrais, que diz respeito à variabilidade do tempo necessário para processar cada requisição em cada camada.

1.5 Organização

Esta dissertação é organizada como segue. O Capítulo 2 apresenta trabalhos relacionados na área de gerenciamento de capacidade, ataques de segurança, custos e restrições de energia. O Capítulo 3 define e formaliza o problema tratado nesta dissertação e a plataforma alvo. O Capítulo 4 apresenta o arcabouço autônomo de gerenciamento de capacidade prévio [3,4] estendido nesta dissertação. As extensões para considerar infra-estruturas multicamadas, ataques de segurança, custos e restrições de energia são apresentadas no Capítulo 5. A avaliação do novo arcabouço num grande número de cenários sintéticos e realistas é apresentada nos Capítulos 6 e 7. Por fim, o Capítulo 8 sumariza as principais conclusões, contribuições e idéias para trabalhos futuros.

2 *Trabalhos Relacionados*

Nesta seção apresentaremos uma revisão bibliográfica dos trabalhos relacionados a esta dissertação. Esta revisão tem como objetivo situar este trabalho no contexto maior no qual se insere e dar embasamento teórico para que o leitor compreenda os capítulos seguintes. O problema de gerenciamento de capacidade de infra-estruturas, abordando várias de suas facetas, é visitado na Seção 2.1. As Seções 2.2 e 2.3 discutem ataques de segurança e energia, respectivamente, sem perder o foco no objetivo de gerenciamento de capacidade.

2.1 Computação Utilitária

Os autores de [110] definem computação utilitária como o provisionamento escalável, confiável e robusto de serviços eletrônicos em resposta a eventos internos ou externos. A idéia de computação utilitária vem revolucionando a forma que serviços computacionais são comprados, organizados e distribuídos. Algumas motivações dadas em favor da computação utilitária estão no fato de que é mais fácil organizar a infra-estrutura para servir aplicações do que o inverso, é importante aproveitar melhor a mão-de-obra de escassos gerentes para infra-estruturas computacionais e é cada vez mais essencial adaptar rapidamente a mudanças de demanda.

Ainda em [110], os autores argumentam que uma plataforma completa para computação utilitária deve ir além do provisionamento autônomo de capacidade e considerar outros fatores como um modelo de negócio. Modelos de negócio são especialmente importantes pois eles definem a interação entre o provedor da infra-estrutura e seus clientes, as empresas que pagam para terem seus serviços hospedados. Especificamente, os modelos de negócio definem o retorno financeiro obtido pelo provedor da infra-estrutura e a qualidade de serviço provida a seus clientes. Por fim, os autores discutem uma série de problemas que precisam ser solucionados para a consolidação da computação utilitária, como controle dinâmico, confiabilidade, segurança, interfa-

ces comuns e dimensionamento baseado em requisitos de qualidade.

Um estudo de 11 casos de terceirização de infra-estrutura computacional é feito em [89], discutindo os benefícios e riscos associados. Apesar da terceirização trazer vantagens como diminuição dos custos e capacidade sob demanda, empresas que optaram pela terceirização procuraram minimizar riscos estratégicos mantendo conhecimento em áreas chaves como economia e administração, terceirizando apenas parte do conhecimento técnico. Uma das conclusões do trabalho é que terceirização de infra-estruturas vai se tornar mais comum, apesar de serem necessárias métricas e definições de qualidade bem gerenciadas.

Para firmar as métricas e definições de qualidade são necessários contratos de negócio flexíveis entre prestadores de serviço e provedores de infra-estrutura. Rappa [83] discute o futuro dos modelos de negócio para computação utilitária, comparando esta a outros serviços comuns como água e energia elétrica. O autor comenta que usuários dependem cada vez mais de serviços eletrônicos, tornando a confiança no funcionamento da computação utilitária um fator importante para seu sucesso. Além disso, quanto mais difundidos esses serviços eletrônicos, maior será a importância da escalabilidade e utilização eficiente dos recursos computacionais disponíveis.

De forma similar, os autores de [21] discutem três interesses cruciais do provedor de infra-estrutura: maximizar a satisfação de seus clientes com relatórios competitivos do nível de serviço recebido, diminuir o custo de prover qualidade e, por último, minimizar o impacto de violações de qualidade de serviço no objetivo de negócio do provedor (lucro), o que é viabilizado por um modelo de negócio.

Devido aos fatores enunciados nesses trabalhos, a Seção 2.1.1 discute alguns trabalhos relacionados à qualidade de serviço prestado ao usuário e a consequente rentabilidade obtida pelo provedor. Estes trabalhos servem de base para a formulação de modelos de negócio aplicáveis à computação utilitária. Em seguida, a Seção 2.1.2 discute mecanismos de virtualização. Por último, a Seção 2.1.3, apresenta vários esforços realizados na direção dos outros requisitos da computação utilitária como escalabilidade, eficiência e automação.

2.1.1 Qualidade de Serviço e Rentabilidade

A qualidade do serviço provido está relacionada com a satisfação dos usuários das aplicações hospedadas. Há várias formas de medir qualidade de serviço. A mais

comum, utilizada no modelo de negócio descrito na Seção 4.2, combina requisitos sobre a taxa de processamento da aplicação, medida em requisições por segundo, e o tempo de resposta das requisições. O tempo de resposta mede a duração do intervalo que a infra-estrutura leva para processar uma requisição.

A rentabilidade do provedor depende fortemente da qualidade de serviço prestada. Isso por que a quantidade de usuários de uma aplicação e o preço que eles pagam, que definem a rentabilidade da aplicação, dependem da qualidade do serviço recebido. Por fim, a empresa que opera a aplicação paga ao provedor uma quantidade relativa ao lucro obtido. A relação entre a qualidade de serviço e a rentabilidade do provedor é analisada em [50]. Os autores analisam a variação da rentabilidade de acordo com diferentes níveis de qualidade de serviço prestado, modelando o relacionamento entre o tempo de resposta e o valor que os usuários aceitam pagar pelo serviço. Os autores validam o modelo com opiniões de usuários reais, concluindo que o valor do serviço decresce rapidamente com o aumento do tempo de resposta.

Um modelo para estimar o valor de um usuário para uma empresa é proposto em [32]. O modelo é genérico o bastante para ser aplicável a vários tipos de aplicações e os autores apresentam um método de parametrização do modelo através dos registros de acesso (*logs*). O modelo é validado para uma aplicação de loja virtual; neste contexto, por exemplo, um cliente que compra freqüentemente tem maior valor que um cliente que raramente compra algum produto.

A compreensão a respeito do valor de um usuário é refinada em [66]. Partindo da premissa que o objetivo do provedor de infra-estrutura é o lucro obtido com a hospedagem das aplicações, eles concluem que o gerenciamento da capacidade dessas infra-estruturas deve focar na rentabilidade do provedor e não em simples requisitos de desempenho das aplicações hospedadas. Assim, eles definem o conceito de taxa de processamento rentável, medida em dólares por segundo. Para melhorar a taxa de processamento rentável, os autores propõem um esquema de priorização de requisições de usuários de acordo com o potencial de renda de cada uma delas. Para tanto, eles usam grafos de modelo de comportamento do usuário (*customer behavior model graph*) (CBMG) [65]. Por exemplo, as requisições de um usuário de uma loja virtual com um carrinho cheio receberiam maior prioridade que requisições de um usuário que está apenas visualizando produtos. Os autores de [97] apresentam uma abordagem similar, com métodos alternativos para decidir a qual CBMG um usuário pertence e como estimar o valor de uma requisição.

2.1.2 Mecanismos de Virtualização

A tarefa de hospedar várias aplicações em hardware compartilhado é facilitada por mecanismos de virtualização. Estes mecanismos criam máquinas virtuais sobre o *hardware*, e cada uma dessas máquinas virtuais pode então executar *software* distinto de forma totalmente isolada. Sistemas operacionais [28, 34, 70, 78] tradicionais não são adequados para infra-estruturas de hospedagem, pois tratam isolamento e alocação de recursos em uma só entidade: o processo. Uma discussão detalhada deste problema é apresentada em [14], no qual os autores apresentam uma solução para desassociar o isolamento e a alocação de recursos em duas entidades diferentes.

Mecanismos de virtualização podem ser divididos em dois grupos. O primeiro deles permite acesso apenas a instâncias virtuais, emuladas, dos recursos físicos. O segundo, conhecido como paravirtualização, permite acesso *controlado* aos recursos físicos sempre que possível. Mecanismos de paravirtualização são alvo de maior interesse pois atingem desempenho de ordem de grandeza superior a mecanismos do primeiro grupo [107].

Alguns mecanismos de virtualização, como o apresentado em [108], não objetivam virtualizar perfeitamente os recursos físicos, com todas as suas funcionalidades. Através da virtualização de apenas um subconjunto das funcionalidades, maior escalabilidade e eficiência são atingidos, ao custo de maior dificuldade e limitações sobre as aplicações que podem ser executadas. Por exemplo, sistemas operacionais genéricos não são suportados, o que faz que aplicações tenham de ser especificamente desenvolvidas para o ambiente. Em contrapartida, outros mecanismos [16, 103] objetivam virtualizar todas as funcionalidades dos recursos físicos, e assim conseguem executar diferentes instâncias de sistemas operacionais em suas máquinas virtuais.

Mecanismos de virtualização têm sido alvo de grande interesse na consolidação de infra-estruturas para hospedagem de aplicações. Elas facilitam a implantação (remoção) de aplicações através da simples criação (destruição) de máquinas virtuais e provêm isolamento de capacidade entre aplicações hospedadas, evitando que erros e sobrecarga em uma aplicação afete outras máquinas virtuais. Além disso, a hospedagem de várias aplicações no mesmo *hardware* melhora o compartilhamento e a utilização dos recursos.

Implementação e análise de arcabouços de gerenciamento de capacidade que utilizam o Xen [16] são apresentados em [20] e [94]. Os autores de [94] caracterizam o

tempo necessário para execução de operações básicas sobre máquinas virtuais como iniciar, destruir, parar, continuar e migrar. Em [20] os autores apresentam um método para classificar cargas de trabalho que tiram maior benefício de migração dinâmica, ou seja, remapeamento dinâmico de máquinas virtuais e recursos físicos.

2.1.3 Gerenciamento de Infra-Estruturas

Como dito anteriormente, vários requisitos são impostos sobre as infra-estruturas de hospedagem, como confiabilidade, disponibilidade, qualidade de serviço, escalabilidade, dentre outros. Devido a esses requisitos e à crescente complexidade dessas infra-estruturas, sua gerência manual se tornou excessivamente árdua para humanos. Sob esta luz, abordagens autônomas (ou auto-adaptativas) surgiram como uma solução para remediar a complexidade desses sistemas. Assim, uma quantidade significativa de trabalhos prévios propõem arcabouços de controle autônomos, cada um usando diferentes técnicas para uma variedade de diferentes propósitos.

Métodos de Controle de Admissão

Métodos de controle de admissão decidem quais requisições devem ser processadas pela infra-estrutura. Requisições que não são admitidas são rejeitadas e não utilizam nenhum recurso da infra-estrutura (a não ser os necessários para sua rejeição). O controle das requisições admitidas evita que a infra-estrutura fique sobrecarregada, melhorando a qualidade do serviço recebido pelos usuários. Todos os trabalhos desta seção têm uma importante característica comum: eles consideram que a capacidade disponível para processamento das requisições é fixa, focando apenas no controle de admissão para controle da utilização ou rentabilidade do provedor.

Os autores de [24] apresentam um método de controle de admissão de sessões num servidor Web, para evitar que este fique sobrecarregado em períodos de carga alta. O mecanismo proposto visa manter a utilização do servidor abaixo de um valor limite. A utilização é definida como a razão entre a taxa de processamento que precisa ser realizado pelo sistema e sua capacidade total de processamento [54]. Os autores apresentam duas técnicas distintas de selecionar as sessões admitidas. Outro mecanismo de controle de admissão que aumenta suavemente a fração das sessões rejeitadas é apresentado em [111], evitando oscilações na utilização do sistema que podem acontecer em controles de admissão binários, que decidem periodicamente

aceitar ou rejeitar todas as sessões, como uma das alternativas em [24].

Em [31] os autores avaliam o impacto de políticas de controle de admissão combinadas com políticas de escalonamento. Para o conjunto de requisições admitidas, os autores consideram escalonamento normal (primeiro-a-chegar primeiro-a-sair) e escalonamento prioritário (menor trabalho primeiro). Para o método de escalonamento prioritário, é proposto um método para evitar inanição de requisições grandes. Experimentos com um protótipo real mostram que o sistema atinge desempenho estável durante períodos de sobrecarga, e que o escalonamento prioritário melhora o tempo de resposta de requisições pequenas significativamente, ao custo de um pequeno atraso das requisições grandes.

Em [81], os autores também combinam controle de admissão com várias políticas de escalonamento, considerando também o valor financeiro (preço) de cada processo, ao invés de simplesmente seu tamanho como em [31]. Os autores analisam o desempenho da solução quando há incerteza sobre alguns parâmetros como a capacidade do sistema, taxa de chegada de processos e tolerância do usuário a atrasos. Uma das principais conclusões é que controle de admissão é crucial em infra-estruturas sobrecarregadas, evitando sobrecarga, garantindo qualidade de serviço e melhorando a rentabilidade do provedor através do favorecimento das requisições com maior valor.

Métodos de Alocação de Capacidade

Os métodos apresentados nesta seção podem ser visto com o complemento dos apresentados na seção anterior. Aqui, os métodos focam apenas na alocação de capacidade para atender todas as requisições, sem controle de admissão, da carga de trabalho da infra-estrutura. O objetivo é alocar a menor capacidade possível que ainda satisfaça requisitos de qualidade dos clientes ou maximize a rentabilidade do provedor. Esses métodos não são aplicáveis em cenários onde a infra-estrutura não tem capacidade suficiente para atender a carga de trabalho.

Os autores de [86] consideram um cenário onde um provedor de serviço provê recursos sob demanda. Caso os recursos não estejam disponíveis no momento da demanda, existe um prazo para que eles sejam disponibilizados. Neste cenário, é proposto um método que pesquisa que visa diminuir a capacidade alocada através do compartilhamento de recursos entre aplicações. As decisões sobre quais aplicações devem compartilhar o mesmo *hardware* depende da contenção por recursos existente entre elas. Os mesmos autores apresentam em [87] um método de alocação

de capacidade que objetiva manter a utilização dos recursos dentro de um intervalo, diminuindo (aumentando) a alocação de recursos quando a utilização estiver baixa (elevada). É apresentada uma análise da convergência do método para várias combinações de parâmetros como quantidade mínima e máxima de processadores que podem ser alocados para uma aplicação.

Em [60] é proposto um controlador de recursos. Os autores o acoplam ao HP-UX *Process Resource Manager* [25], que aloca capacidades de acordo com as decisões do controlador. O controlador decide a capacidade para cada aplicação proporcional à diferença entre o tempo de resposta médio, monitorado, e o valor de tempo de resposta que precisa ser satisfeito. Os autores de [67] apresentam um método de alocação de capacidade de recursos, especialmente processador (CPU), através de duas abordagens: configuração dinâmica de prioridade e alocação dinâmica capacidade. Os autores usam heurísticas de pesquisa combinatória para encontrar a alocação de capacidade ou a prioridade das aplicações, que devem atender requisitos de qualidade de serviço sobre a média do tempo de resposta das requisições.

Os autores de [59] apresentam um método de alocação de capacidade que modela o desempenho da infra-estrutura com um modelo de filas. Os autores usam filas M/M/1 com escalonamento FCFS (primeiro-a-chegar primeiro-a-sair) e filas M/G/1 com compartilhamento de processador (*processor sharing*) (PS). O modelo da infra-estrutura é acoplado a um modelo de otimização com restrições convexas que encontra a melhor alocação de capacidade. É mostrado que esse modelo de otimização tem uma interpretação geométrica simples no caso de restrições sobre a média do tempo de resposta. A versão do modelo de otimização que provê garantias sobre a cauda da distribuição do tempo de resposta, porém, não é tão simples.

Modelos analíticos de filas são tratados em profundidade em [54]. A escolha do trabalho anterior em utilizar filas M/M/1 FCFS e filas M/G/1 PS é comum em trabalhos da área, pois uma fórmula fechada para a distribuição do tempo de resposta é conhecida para estas duas filas. A fila com compartilhamento de processador é mais simples que a fila com escalonamento FCFS e por isto pode ser utilizada com qualquer distribuição do tempo de serviço [54].

Métodos de Controle Utilitários

Esta seção apresenta métodos de controle utilitários. Esses métodos combinam as duas idéias apresentadas nas seções anteriores: alocação de capacidade e controle de

admissão. Desta forma, esses métodos respondem a flutuações de demanda e podem ser aplicados mesmo quando a infra-estrutura está sobrecarregada.

Um dos primeiros trabalhos a considerar computação utilitária foi apresentado em [61]. Os autores propõem modelos de desempenho baseados em filas M/M/1 FCFS e M/G/1 PS juntamente com um modelo de negócio simples. Eles consideram garantias sobre a cauda da distribuição do tempo de resposta e que uma requisição pode voltar à infra-estrutura depois de receber serviço. O problema de otimização responsável por maximizar o lucro do provedor é separável e os autores apresentam um método iterativo para solucioná-lo. A solução do modelo de otimização é a alocação de capacidade para cada aplicação e a quantidade de requisições que podem ser aceitas para processamento sem violar o requisito de qualidade.

O trabalho em [102] apresenta uma caracterização da carga de um servidor Web, concluindo que chegadas de sessões seguem um processo de Poisson. Partindo disso, os autores propõem um modelo de desempenho baseado em filas M/G/1 PS, utilizando três aproximações para o cálculo da distribuição do tempo de resposta. O modelo de desempenho e um modelo de negócio simples são usados por um modelo de otimização que considera restrições sobre a cauda da distribuição do tempo de resposta.

Gerenciamento de Infra-Estruturas Multicamadas

Uma limitação de todas as soluções de gerenciamento de capacidade descritas até este ponto é que elas só consideram infra-estruturas com uma camada. Porém, aplicações modernas são geralmente compostas de múltiplas camadas. Cada camada é responsável por uma tarefa específica no processo de servir uma requisição. Exemplos de camadas típicas de serviços Web modernos incluem: camada de apresentação (ex.: servidores HTTP [33,68]) que formatam os dados para envio ao usuário; camada de aplicação (ex.: servidores J2EE ou .NET [27,69]) que implementam funcionalidades como pesquisa por conteúdo ou execução de lance em sítios de leilões; e camada de banco de dados (ex.: Oracle, PostgreSQL ou MySQL [1, 44, 77]) que armazenam os dados persistentes da aplicação.

Há uma pequena quantidade de trabalhos tratando de gerenciamento de capacidade em infra-estruturas multicamadas. Modelos analíticos de filas em rede, que poderiam ser utilizados para modelagem de desempenho dessas infra-estruturas, já foram estudados no passado [17,88]. Porém, apesar desses modelos serem precisos e

flexíveis, acoplá-los a um modelo de otimização não é trivial devido à complexidade das equações associadas.

Um modelo de desempenho de infra-estruturas Web multicamadas é apresentado e validado em [99]. Os autores utilizam uma rede de filas com política de escalonamento de compartilhamento de processador (PS) para modelar o desempenho da infra-estrutura. Esse modelo de desempenho é utilizado numa análise de valores médios (*mean value analysis*) [55], para calcular a média do tempo de resposta das requisições. Os autores validam o modelo com experimentos reais e mostram que ele prevê com precisão o tempo médio de resposta das aplicações hospedadas. Apesar disso, um modelo de desempenho por si só não serve para fazer gerenciamento autônomo de capacidade, precisando para isso estar acoplado a um arcabouço.

Um arcabouço completo é proposto em [100]. Os autores utilizam filas G/G/1 PS para modelar o desempenho da infra-estrutura, e apresentam um método para alocar a capacidade entre as aplicações de forma a garantir requisitos sobre a média do tempo de resposta. Este método faz uma simplificação da infra-estrutura e considera cada camada individualmente, forçando o tempo médio de resposta em cada camada ser menor que um valor fixado, garantindo assim que a soma dos tempos médios de resposta de todas as camadas seja menor que o requisito de qualidade.

O arcabouço proposto nesta dissertação é uma contribuição ao campo de gerenciamento de capacidade pois combina vários quesitos desejáveis. Primeiro, ele utiliza um modelo de negócio flexível e adaptativo. Segundo, o modelo de desempenho captura os principais fatores de infra-estruturas multicamadas, sendo capaz de prover garantias sobre a cauda da distribuição do tempo de resposta. Tudo isso é acoplado a um modelo de otimização simples o bastante para ser solucionado em tempo real. O arcabouço proposto ainda considera ataques de segurança, custos e restrições de energia; trabalhos destas áreas são apresentados nas próximas seções.

2.2 Ataques de Segurança

Vários tipos de ataques maliciosos podem ser realizados na Internet. A motivação por trás deles são variadas. Vários são feitos por fama ou auto-afirmação, outros por vingança ou rixas pessoais. Há ataques com interesse financeiro, comprometendo o sítio ou serviço de um concorrente, piorando sua imagem ou gerando insatisfação entre seus clientes [29, 42, 46]. Esses ataques são possíveis devido a vários fatores. A

arquitetura da Internet atual preza pela eficiência, disponibilizando apenas o mínimo de recursos necessários para mover pacotes de dados de um ponto a outro da rede, segundo a política do melhor esforço. Essa arquitetura é flexível, permitindo vários artefatos que possibilitam ou potencializam ataques.

Significativo esforço para a classificação de ataques de negação de serviço e métodos de defesa foi feito em [71]. Apesar desse trabalho classificar ataques segundo um grande número de características, este texto foca apenas as características mais relevantes para computação utilitária, que é o foco do nosso trabalho. A Seção 2.2.1 discute as duas metodologias de ataques: semânticos e de força bruta. Na Seção 2.2.2 são discutidos os tipos de alvo de um ataque: aplicações específicas ou a infra-estrutura como um todo. A Seção 2.2.3 apresenta mecanismos de defesa contra ataques, que podem ser implantados no servidor ou na rede. Por último, a Seção 2.2.4 discute métodos de detecção e rastreamento de ataques.

2.2.1 Metodologias de Ataque

Em *ataques semânticos*, o usuário malicioso possui conhecimento específico do sistema alvo, como alguma falha de segurança ou problema de desempenho, que possa ser utilizado para efetuar um ataque. Geralmente, ataques semânticos podem ser efetivos mesmo que o atacante possua muito menos recursos que o alvo, e podem ser solucionados por meio do reparo do problema de segurança ou desempenho utilizado para efetuá-lo. O impacto de um ataque semântico é difícil de prever pois depende da gravidade da falha utilizada para realizá-lo.

Um exemplo de ataque semântico é inundação de pacotes SYN [90], onde uma característica do protocolo TCP é utilizada para exaurir recursos limitados de um servidor. Apesar de várias contramedidas terem sido feitas contra este tipo de ataque, como [90], a contramedida mais difundida atualmente é chamada SYN *cookies* [18], que elimina a característica original do protocolo TCP usada pelo ataque.

Ataques de *força bruta* saturam todos os recursos disponíveis em um servidor por meio do envio de uma quantidade de requisições, ou dados, maiores do que a suportada. Uma quantidade de recursos proporcional à disponível no alvo é necessária aos atacantes para efetivar o ataque. Para atacar alvos bem provisionados, atacantes usam ataques distribuídos partindo de milhares de máquinas. Essas máquinas que cooperam para realizar ataques formam as chamadas *botnets* [95].

Ataques de força bruta que enviam requisições para um serviço visam consumir recursos que deveriam ser destinados a usuários legítimos. Quanto maior for o número de requisições ilegítimas enviadas pelo usuário malicioso, menor será a capacidade dedicada aos usuários legítimos e pior será a qualidade do serviço recebido por eles. No caso de ataques que enviam um grande volume de dados, essa inundação de dados deve ser intensa o suficiente para sobrecarregar os recursos de rede no alvo do ataque, fazendo com que dados legítimos sejam descartados devido à incapacidade dos roteadores de rede em transmitir todos os dados enviados.

2.2.2 Tipos de Alvos de Ataques

No âmbito da computação utilitária, ataques de negação de serviço podem afetar a infra-estrutura como um todo, comprometendo todas as aplicações hospedadas, ou apenas uma aplicação específica, sem comprometer as demais.

Ataques que afetam a infra-estrutura incluem os de força bruta que enviam um grande volume de dados ao alvo, o que impede o recebimento de requisições legítimas, afetando o funcionamento de todas as aplicações. Quando toda a infra-estrutura está comprometida por algum ataque, é necessário a utilização de algum método de defesa específico, como os discutidos na Seção 2.2.3; pois o controle de admissão e alocação de capacidade funcionam no nível de aplicações, não tendo meios de mitigar o impacto de um ataque no nível da infra-estrutura.

No caso de ataques que afetam uma aplicação específica, alocação dinâmica de capacidade e controle de admissão podem levar em consideração o ataque para tomar decisões de alocação e admissão que minimizem o impacto do ataque no objetivo de negócio do provedor (ex.: rentabilidade) e seus clientes (ex.: qualidade de serviço). Esta dissertação foca neste tipo de ataque.

2.2.3 Defesas Contra Ataques

Esta seção apresenta diferentes tipos de defesas contra ataques de negação de serviço sob o foco da computação utilitária. Defesas contra ataques de negação de serviço podem ser preventivas [18, 53, 73, 90, 92] ou reativas [38, 105, 106]. Alguns métodos apresentam ambas características [7, 13].

Métodos preventivos tomam medidas proativas com o objetivo de evitar ou difi-

cultar a ocorrência de um ataque. Métodos preventivos podem ser feitos, por exemplo, através do controle de acesso aos recursos. Métodos reativos são aqueles que tomam medidas contra ataques em andamento depois que eles são detectados. As medidas tomadas podem incluir a identificação do agente por meio de técnicas de rastreamento [91] ou filtragem dos fluxos de dados ilegítimos. Num cenário de computação utilitária, a natureza reativa do módulo de gerenciamento de capacidade cria um ambiente propício para a integração de defesas reativas.

Métodos de Defesa Implantados no Servidor

Métodos de defesa implantados no servidor são aqueles que adicionam funcionalidades e utilizam recursos do servidor para protegê-lo de ataques de negação de serviço. Métodos implantados no servidor são de especial interesse para esse trabalho, pois podem ser adicionados ao nosso módulo de gerência de capacidade, criando uma solução muito mais robusta para infra-estruturas de hospedagem.

Em [92], os autores apresentam um sistema operacional capaz de se defender de ataques de negação de serviço por meio de dois métodos ortogonais. O primeiro deles é a contabilização dos recursos disponíveis e utilizados, de forma que qualquer uso indevido de recursos possa ser identificado e impedido. O segundo método de proteção é a criação de domínios de proteção distintos para cada aplicação, isolando uma da outra. Esses conceitos estão presentes hoje nos sistemas de virtualização [16, 108], utilizados no arcabouço proposto nesta dissertação.

O mecanismo proposto em [106] para defesa de ataques de negação de serviço é efetivo contra ataques força-bruta no nível da aplicação. O mecanismo proposto adiciona custo às requisições, cobrado através de problemas criptográficos para os quais clientes precisam encontrar soluções. Durante um ataque, os clientes gastam processamento para aumentar a qualidade de suas soluções de forma a garantir a obtenção do serviço. Como os atacantes enviam milhares de requisições por segundo, eles não terão como encontrar soluções tão boas quanto as dos clientes legítimos. Em contrapartida, há um aumento da latência do início de uma conexão devido à procura e conferência das soluções.

De forma similar, os autores de [105] propõem um mecanismo chamado *Speak-up*, efetivo contra ataques de força-bruta a aplicações específicas. Nesse mecanismo, os clientes também são incentivados a pagar para receber serviço, sendo servidos de acordo com a quantidade de dados que enviam ao servidor. A premissa de funci-

onamento do mecanismo é embasada no fato de que os atacantes estarão gastando toda sua banda para enviar o máximo de requisições, enquanto os clientes legítimos possuem banda ociosa disponível. O problema do *Speak-up* é que ele assume que a infra-estrutura tem banda de rede disponível para receber todos os dados enviados pelos atacantes e pelos clientes legítimos, o que nem sempre é realista.

Outros mecanismos de defesa incluem idéias similares como: (1) a diferenciação de requisições legítimas de requisições ilegítimas através da utilização de CAPTCHAS [73], uma figura com letras distorcidas que apenas humanos conseguem reconhecer, (2) arquiteturas dedicadas ao monitoramento e defesa de ataques [7], ou ainda (3) defesas contra um tipo específico de ataque [90].

Métodos de Defesa Implantados na Rede

Métodos de defesa implantados na rede não são o foco do nosso trabalho pois geralmente necessitam de intervenções na estrutura da Internet, o que geralmente não é possível, além de estar fora do escopo de um arcabouço de gerenciamento de capacidade. Devido a essa limitação, essas soluções não são largamente difundidas, apesar de serem soluções promissoras para a defesa de ataques de força-bruta no nível da infra-estrutura.

Os autores de [53] apresentam uma rede sobreposta (*overlay*) chamada SOS (*Secure Overlay Services*). O foco do SOS é garantir a comunicação entre dois pontos da rede sem que agentes externos possam comprometê-la. A solução usa tunelamento seguro, roteamento através de funções de *hashing* e filtragem de dados. O funcionamento do SOS é baseado na autenticação de um cliente legítimo. Após autenticado, esse cliente obtém acesso à rede SOS, que se encarrega de encaminhar seus dados de forma segura até o destino. Em [73], o SOS foi estendido para funcionar numa arquitetura aberta onde o usuário autentica-se resolvendo um CAPTCHA.

Uma abordagem auto-adaptativa para defesa de ataques é apresentada em [38]. Os autores apresentam um modelo para detecção de anomalias na rede. Quando um ataque é detectado, fluxos legítimos são isolados de fluxos ilegítimos. O sistema utiliza uma estrutura de rede chamada *Cognitive Packet Networks* (CPN) [37], que possibilita ao servidor informar aos roteadores de rede que bloqueiem ou limitem tráfego dos fluxos ilegítimos. Além de bloquear e limitar fluxos ilegítimos, a rede também encontra caminhos robustos para transferência dos dados legítimos. A filtragem de dados ao longo da rede é eficiente não só para evitar que os recursos do servidor se-

jam exauridos, mas também para evitar saturação dos canais de transmissão da rede, o que torna as CPNs muito mais resistentes a ataques contra a rede.

2.2.4 Detectando Ataques

A detecção de ataques compartilha muitas características com sistemas de detecção de intrusão (SDI), discutidos em [13]. Esses sistemas utilizam dois métodos para detectar ataques. O primeiro é baseado em assinaturas, que só funcionam contra ataques já conhecidos, que são catalogados num banco de dados. Quando o sistema de detecção reconhece na rede ou no servidor um padrão (assinatura) já catalogado no banco de dados, um ataque é identificado. As vantagens desses sistemas são que ataques conhecidos sempre são identificados e nenhum falso positivo é levantado. Porém, esse método não identifica ataques novos ou desconhecidos e necessita de que o banco de dados seja mantido sempre atualizado. Um exemplo de SDI que faz detecção baseada em assinaturas é o SNORT [85].

O segundo método de detecção de ataques funciona através da detecção de anomalias na rede ou no servidor. Esses métodos constroem um perfil do comportamento normal da rede ou serviço, e quando o comportamento corrente difere significativamente do comportamento normal, um ataque é identificado. A detecção de anomalias funciona contra todos os tipos de ataques, porém pode ser imprecisa ou levantar falsos-positivos, principalmente devido ao fato de que anomalias nem sempre são maléficas, como é o caso das *flash crowds*.

Um método automático de detecção de anomalias baseado em *wavelets* é apresentado em [15]. O método permite a detecção do padrão normal de funcionamento do sistema no nível de horas, dias, semanas e meses. Variações em qualquer um dos níveis analisados são detectadas através da análise da diferença entre as medidas atuais e o padrão esperado, possibilitando a detecção de anomalias de curta e longa duração. Os autores analisam diferentes tipos de anomalias: *flash crowds*, ataques, problemas de disponibilidade e erros de medição. Os autores mostram que a quantidade de falsos-negativos é satisfatoriamente baixa, mas nenhuma análise sobre falsos-positivos é apresentada.

Em [72], os autores usam uma técnica chamada *backscatter* para estimar a quantidade e características de ataques de inundação na Internet. Os autores argumentam que uma estimativa do número de ataques que está acontecendo na Internet pode ser

obtida através do monitoramento de um grande número de endereços IP. Uma limitação da metodologia é que apenas ataques que falsificam o campo de origem e geram uma resposta do alvo podem ser detectados; outros tipos de ataques, como ataques de reflexão [47] ou que enviam requisições a uma aplicação, não são capturados.

O uso de um computador conectado à Internet para ser vítima de subversão por uma *botnet* de forma proposital, para depois usá-lo na análise do tráfego gerado e recebido pela *botnet*, é chamado de *honeypot* [104]. Essa técnica difere das anteriores no sentido de que não visa apenas detectar a anomalia, mas compreender seu comportamento. A criação de *honeypots* deve ser feita de forma controlada, permitindo que pesquisadores analisem o comportamento das *botnets* mas sem cooperar com elas na execução de ataques. Em [91], os autores apresentam um método para rastrear um pacote IP específico na Internet até a sua fonte. Se tal ferramenta estivesse disponível na Internet, as fontes de ataques poderiam ser rastreadas e as devidas medidas poderiam ser tomadas contra elas. Porém, a difusão da ferramenta é impedida devido a problemas de escalabilidade e dificuldades de implantação.

Por último, a caracterização de ataques é difícil por que os dados são difíceis de serem coletados. Além disso, o acesso a registros de tráfego é difícil de ser conseguido por questões de privacidade. Mesmo quando dados estão disponíveis, pesquisadores não têm como conferir ou estimar a corretude da caracterização, pois os atacantes não liberam informações sobre seus ataques. Algumas características de ataques de segurança são levantadas em [47] e [72].

2.3 Custos e Restrições de Energia

Desde o início da computação, processadores (CPU) vêm ficando cada vez mais velozes. Esse aumento de velocidade vem acompanhado de um aumento no consumo de energia e calor dissipado. Centros computacionais modernos concentram de dezenas a milhares de computadores num pequeno espaço físico, resultando numa alta densidade de consumo de energia, medida em W/m^2.

Em [19] os autores fazem um estudo do consumo de energia de centros computacionais, analisando o impacto de economia de energia e tendências tecnológicas em centros computacionais de alta-densidade. Os autores calculam uma densidade de energia de 355 W/m^2 e argumentam que a densidade do consumo de energia ficou constante durante o ano analisado. Porém, esta estabilidade no consumo deve-se a

economia de energia no sistema de refrigeramento e iluminação; o consumo da infra-estrutura computacional, especificamente, aumentou 55% no mesmo período.

Problemas de disponibilidade de energia já vêm sendo fator crucial para a decisão de onde construir grandes centros computacionais [63], por exemplo, próximos a usinas de energia. A empresa de consultoria Gartner reporta que atualmente gastos com energia correspondem a menos de 10% do orçamento de empresas, porém este número pode aumentar para mais de 50% nos próximos anos [41]. O Green Grid [5] é uma iniciativa na direção de infra-estruturas que fazem uso eficiente de energia, apoiada pelos maiores fabricantes de *hardware* e prestadores de serviço de hospedagem atuais, que visa padronizar métricas e objetivos utilizados no projeto de novas infra-estruturas mais econômicas. Outros esforços para minimizar o consumo de energia são apresentados em [64], por exemplo o projeto *80 Plus*, que certifica fontes de energia para computadores com eficiência maior que 80%.

Energia é um ponto crucial em redes de sensores ou dispositivos móveis. Nesses contextos, uma enorme quantidade de trabalhos já abordaram o problema de disponibilidade e consumo de energia. O leitor é direcionado para [9, 49], e referências inclusas, como uma cobertura sobre o assunto. Os esforços das áreas de redes de sensores e dispositivos móveis não são aplicáveis ao nosso foco de gerenciamento de capacidade, pois a plataforma considerada e os desafios são diferentes.

2.3.1 Economia de Energia em Centros Computacionais

Métodos para economizar energia baseados em *hardware* foram apresentados em [56]. Os autores discutem o estado atual das tecnologias e métodos de economia de energia na CPU como: variação dinâmica de voltagem e freqüência, processamento simultâneo de processos, concentração de carga e substituição de processadores poderosos por outros menores mais econômicos. Os autores também motivam e exploram métodos de economia de energia em memória, como posicionamento dos dados, mapeamento de endereços de memória, compressão e coerência de cache.

Um mecanismo para programas MPI [74] economizarem energia em centros de processamento com controle dinâmico de energia é apresentado em [52]. Os autores argumentam que uma quantidade significativa de energia pode ser economizada devido ao desbalanceamento de carga entre os nós de processamento. O mecanismo proposto diminui a freqüência do processador [11, 26] dos nós com carga menor, de

forma que eles terminem sua parte do trabalho junto dos outros nós, economizando energia sem aumentar o tempo total de execução do processo.

Outros métodos similares que visam economizar energia sem inflacionar o tempo de execução de processos científicos são propostos em [36] e [45]. Em [36] os autores comparam três formas de fazer o controle de freqüência, com crescente economia de energia: através de um programa externo, configuração manual antes da execução do programa e controle explicito da freqüência no código da aplicação. Em [45] os autores propõem um algoritmo para controlar a freqüência do processador utilizando a quantidade de acessos externos ao processador (ex.: acessos a disco e *cache misses*). Os métodos atingem economias significativas de energia a custo de pequenos atrasos no tempo de execução das aplicações. Porém, a quantidade de energia que pode ser economizada depende fortemente da aplicação.

Por último, os autores de [57] propõem um processador com aceleração vetorial como uma alternativa econômica em ambientes de computação científica, apontando para a possibilidade da utilização de processadores especializados para economizar energia e ainda melhorar o desempenho de aplicações específicas.

2.3.2 Gerenciamento de Capacidade Face a Custos de Energia

Na área de gerenciamento de capacidade, poucos trabalhos consideraram custos de energia. O trabalho em [10] associa custos de energia à capacidade alocada para as aplicações. Desta forma, o arcabouço objetiva maximizar a rentabilidade do provedor, calculando o mínimo de capacidade necessária para atender a carga de trabalho admitida para processamento, desligando os recursos que não estão sendo utilizados para economizar energia. Os autores concluem que ignorar custos de energia mantendo a infra-estrutura sempre operacional não é eficiente.

Além de custos de energia, os autores de [22] consideram restrições de energia, períodos nos quais há diminuição, planejada ou não, da energia disponível. O arcabouço captura custos de energia e o valor pago por cada aplicação hospedada, objetivando maximizar a rentabilidade do provedor em períodos de sobrecarga ou redução da disponibilidade de energia. Os autores consideram apenas restrições de desempenho sobre a utilização da infra-estrutura, o que não é suficiente para capturar requisitos sobre o tempo de resposta.

Em [23] é apresentado um arcabouço de gerenciamento de capacidade que mo-

dela custos de energia em duas instâncias. Na primeira instância, o custo de energia cresce linearmente com a quantidade de servidores ligados para atender a carga. Em segunda instância, o arcabouço considera variação dinâmica de freqüência nos processadores [11,26], onde o custo de energia varia de forma não linear com a freqüência de processamento. Os autores ainda consideram o custo resultante da diminuição da vida útil de equipamentos (e.x.: discos rígidos) devido aos desligamentos. Apesar da precisão da modelagem dos custos de energia, os autores consideram apenas restrições sobre a média do tempo de resposta e nenhum modelo de negócio, objetivando apenas minimizar o consumo de energia.

Nosso trabalho contribui sobre os trabalhos anteriores combinando em um arcabouço unificado um modelo de desempenho para infra-estruturas multicamadas, restrições sobre a cauda da distribuição do tempo de resposta, um flexível modelo de negócio, ataques de segurança, custos e restrições de energia.

3 *Definição do Problema*

Este capítulo formaliza o problema abordado nesta dissertação. A Seção 3.1 apresenta o contexto do nosso trabalho, os atores envolvidos e os contratos firmados entre eles. A infra-estrutura de hospedagem considerada é apresentada na Seção 3.2. O problema de gerenciamento de capacidade é formalmente definido na Seção 3.3. Por fim, as principais premissas do nosso trabalho são discutidas na Seção 3.4.

3.1 Contexto

Esta dissertação considera o cenário, cada vez mais comum, onde um provedor possui infra-estrutura dedicada à hospedagem de serviços Web de terceiros. A terceirização da hospedagem de serviços Web tais como correio eletrônico, blogs, portais de notícias, transferência de arquivos, entre outros, é uma prática economicamente atrativa para os provedores desses serviços consolidarem seus servidores [89]. Nesse cenário, consideramos a interação de dois agentes:

Provedor: Definimos como provedor a entidade que gerencia uma infra-estrutura compartilhada e provê serviços de hospedagem para vários serviços Web. Akamai [96] e Rackspace [82] são exemplos de provedores.

Cliente: Definimos como clientes os provedores de serviço Web, que utilizam serviços de hospedagem terceirizados, e capitalizam da utilização de seus serviços por usuários finais. Exemplos de clientes da Akamai incluem MySpace [75], Audi [6], IHC [48] e o *National Center for Missing & Exploited Children* [76].

Nesse cenário, os clientes firmam contratos com o provedor de infra-estrutura, contendo um Acordo de Nível de Serviço (*Service Level Agreement*) (SLA). Nesta dissertação consideramos contratos SLA onde os clientes pagam pela hospedagem de

suas aplicações e, em retorno, exigem do provedor que ele satisfaça certas expectativas de qualidade de serviço. Os principais componentes do SLA são:

Taxa de Processamento: O requisito de taxa de processamento é o primeiro dos requisitos de qualidade de serviço do SLA. Ele define quantas requisições por segundo o provedor deve atender da aplicação hospedada. Quanto maior o requisito de taxa de processamento, maior a capacidade que precisa ser dedicada à aplicação em períodos de carga pesada.

Tempo de Resposta: O segundo requisito de qualidade de serviço é o tempo de resposta, que deve ser satisfeito para cada uma das requisições atendidas em função do requisito anterior. O tempo de resposta de uma requisição é medido como o tempo que a infra-estrutura gasta para atendê-la, desde que é recebida até quando sua resposta é enviada ao cliente, desconsiderando tempo de transmissão na rede. Quanto maior o requisito do tempo de resposta, mais tempo as requisições podem permanecer na infra-estrutura e menor é a capacidade que o provedor precisa dedicar a cada requisição individual.

Custo por Requisição: O custo cobrado dos clientes pela hospedagem das aplicações é proporcional à rijeza de seus requisitos de qualidade de serviço descritos acima. Clientes que executam aplicações críticas podem estar dispostos a pagar mais para obter uma taxa de processamento maior e/ou menores tempos de resposta, enquanto outros clientes podem optar por requisitos de desempenho mais amenos a custos mais baixos.

O objetivo do provedor nesse cenário é encontrar a configuração mais eficiente de seus recursos disponíveis, atendendo os requisitos de qualidade de seus clientes e maximizando o lucro obtido da hospedagem de seus serviços.

Cada um desses serviços Web pode ser composto de diferentes tipos de requisições, caracterizadas por diferentes cargas de trabalho, diferentes demandas por recursos e executadas por componentes de *software* independentes. Por exemplo, uma requisição de *login* é tratada por um módulo de *software* com conexão criptografada e uma consulta simples ao banco de dados, enquanto uma requisição de pesquisa por conteúdo é tratada por outro módulo de *software* e pode resultar em várias consultas ao banco de dados. Alternativamente, requisições idênticas, como duas requisições de pesquisa, podem ter requisitos de qualidade diferentes, por exemplo se uma delas

estiver associada a um plano básico e a outra estiver associada a um plano preferencial. Cada um desses tipos de requisição é definido como uma *classe de aplicação*, e supomos que a infra-estrutura hospeda um total de N classes independentes. A carga de um serviço Web pode ser vista então como um conjunto de classes homogêneas, com requisições de uma mesma classe estatisticamente indistinguíveis.

3.2 Infra-estrutura de Hospedagem

Consideramos que a infra-estrutura do provedor é composta de múltiplas camadas, como é o caso de muitos serviços Web, cada uma delas responsável por uma tarefa específica no processo de servir uma requisição. Consideramos que a infra-estrutura tem um número arbitrário K de camadas. Camadas operam em paralelo e requisições visitam camadas em seqüência. Isto é, uma requisição da classe i chega à camada j, é servida, e depois deixa o sistema com probabilidade $p_{i,j}$ ou segue para a camada $j+1$ com probabilidade $1-p_{i,j}$, para $i \in [1,N]$ e $j \in [1,K)$. Todas requisições que chegam à última camada deixam a infra-estrutura após receberem serviço, ou seja, $p_{i,K} = 1$.

Cada camada é hospedada em *hardware* distinto, que é compartilhado por todas as classes. Camadas podem ser constituídas, por exemplo, de computadores multiprocessadores ou um aglomerado (*cluster*) de computadores pessoais. Uma infra-estrutura desse tipo precisa garantir isolamento entre as classes, dentre outras funcionalidades desejáveis. O isolamento é necessário para evitar que problemas como sobrecarga, erros e falhas de segurança em uma classe afetem outras. Assim, supomos que cada camada utiliza um mecanismo de virtualização, como o Xen [16] ou VMWare [103], que provê isolamento e possibilita ao servidor aumentar ou reduzir dinamicamente a quantidade de recursos físicos dedicados a cada classe em cada camada. De fato, essas tecnologias estão sendo alvo de renovado interesse como meio de consolidação de servidores, melhorando a segurança, confiabilidade, disponibilidade, reduzindo custos e provendo flexibilidade [110].

Além do mecanismo de virtualização, supomos que cada classe de aplicação utiliza um esquema de controle de admissão que limita a taxa com a qual as requisições são aceitas para processamento na infra-estrutura. Utilizando algum dos métodos existentes, por exemplo os apresentados em [80], o controle de admissão é responsável por evitar que a taxa de chegada de requisições sobrecarregue a infra-estrutura,

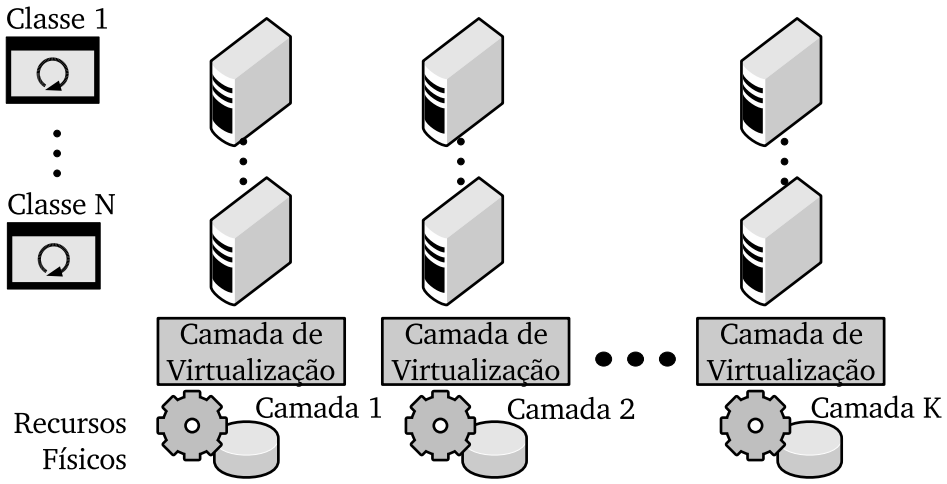


Figura 3.1: Infra-estrutura de Hospedagem Multicamadas

rejeitando uma fração delas, garantindo assim que os requisitos de qualidade de serviço das requisições processadas sejam atendidos.

A Figura 3.1 mostra a plataforma de hospedagem considerada. Sobre a infra-estrutura física de cada camada, o mecanismo de virtualização cria N máquinas virtuais isoladas, uma para cada classe. Dadas K camadas, cada requisição de uma classe é servida por K máquinas virtuais dedicadas àquela classe. Esse modelo de hospedagem isola classes umas das outras, cada uma usando K máquinas virtuais como se fossem servidores dedicados funcionando a uma fração da capacidade física total.

Uma plataforma alternativa, onde todas as camadas são criadas através de um mecanismo de virtualização sobre os mesmos recursos físicos, foi proposta em [67]. Essa abordagem tem a vantagem de simplificar o dimensionamento das camadas. Porém, cenários onde camadas possuem *hardware* específico – por exemplo uma camada de armazenamento e outra de processamento – não são capturados.

3.3 Gerenciamento de Capacidade

O problema de gerenciamento de capacidade é definido como a determinação da fração da capacidade física que deve ser destinada a cada classe i em cada camada j, bem como a taxa de requisições de cada classe i que deve ser admitida para processamento; de forma a maximizar o objetivo de negócio do provedor, tendo como restrição os contratos SLA firmados com cada cliente.

O objetivo de negócio do provedor é expresso como o lucro obtido pela hospedagem das aplicações. Apesar do foco no lucro do provedor, os interesses dos clientes – taxa de processamento, tempo de resposta e custo por requisição – também serão levados em consideração nas extensões do arcabouço para ataques de segurança, custos e restrições de energia apresentadas no Capítulo 5.

Esta dissertação trata do gerenciamento de capacidade da infra-estrutura para as classes hospedadas através de todas as camadas de forma autônoma. Através do provisionamento independente de cada camada, é possível levar em consideração contenção por recursos específicos em cada aplicação, alocando capacidade às aplicações nas camadas onde elas necessitam mais. O gerenciamento de capacidade durante ataques de segurança e restrições de energia objetiva minimizar o impacto negativo desses fatores nos objetivos do provedor e seus clientes.

3.4 Premissas Principais

Esta seção discute as principais premissas deste trabalho. Elas limitam o espaço de projeto considerado e simplificam o sistema alvo.

- Chegadas de requisições para cada classe são dadas por um processo de Poisson, como observado em sistemas reais [61,79,102].
- O tempo de serviço de uma requisição em cada camada segue uma distribuição exponencial. Essa abordagem é usada como uma modelagem razoável para centros de serviços transacionais [61,67,99,102]. Entretanto, é apresentada uma avaliação da sensibilidade do arcabouço a essa premissa na Seção 6.5.
- Todas as requisições de uma classe são estatisticamente indistinguíveis, possuindo a mesma demanda média por serviço em cada camada do sistema.
- O tempo de transmissão dos dados na rede não é considerado. Em outras palavras, o tempo de resposta refere-se ao tempo que a requisição gasta recebendo serviço na infra-estrutura.
- Considera-se que os tempos de residência de uma requisição em cada camada são independentes uns dos outros. Essa premissa de independência troca precisão por simplicidade e aplicabilidade. Argumentamos que a solução proposta

captura os principais aspectos e compromissos do desempenho do sistema, melhorando sobre modelos anteriores, ainda sendo capaz de resolver um complexo modelo de otimização de forma eficiente para cenários práticos. A modelagem de interdependências é deixada como trabalho futuro.

- A capacidade alocada em cada camada é modelada como uma variável contínua.
- Apenas ataques de segurança que visam uma aplicação são considerados. Outros tipos de ataques são discutidos na Seção 5.4. Além disso, requisições de clientes legítimos e as enviadas durante um ataque possuem o mesmo tempo médio de serviço em cada camada.
- Custos de energia são modelados como uma função linear da capacidade alocada. Limitações dessa estratégia bem como modelagens alternativas são discutidas na Seção 5.5.

4 *Arcabouço Autônomo de Gerenciamento de Capacidade*

Esta seção apresenta o arcabouço de gerenciamento autônomo de capacidade proposto anteriormente em [3, 4], que é estendido no Capítulo 5 para prover gerenciamento de capacidade de infra-estruturas multicamadas, considerando ataques de segurança, custos e restrições de energia. A Seção 4.1 apresenta o funcionamento do arcabouço e a interação entre seus módulos. O módulo central do arcabouço, o gerenciador de capacidade, é composto de três componentes principais: um modelo de negócio, detalhado na Seção 4.2, um modelo de desempenho e um modelo de otimização, apresentados na Seção 4.3.

4.1 Arcabouço Autônomo

Para construir um sistema autônomo, ou seja, auto-adaptativo, o arcabouço de gerenciamento de capacidade propõe um modo de operação do sistema baseado em controle cíclico, mostrado na Figura 4.1. O seu núcleo é o *gerenciador de capacidade*, que é chamado periodicamente para alocar a capacidade disponível em cada camada entre as classes de aplicações hospedadas, objetivando maximizar o objetivo de negócio do provedor. O intervalo entre duas intervenções do gerenciador de capacidade é chamado de *intervalo de controle*.

Ao final de cada intervalo de controle, o gerenciador de capacidade utiliza os parâmetros do sistema e os requisitos do SLA de cada classe de aplicação (seta 1 na Figura 4.1), juntamente com estimativas da carga de trabalho esperada para cada classe no próximo intervalo (seta 3), para calcular a nova configuração da infra-estrutura de hospedagem para o próximo intervalo (seta 4).

O módulo *previsor de carga* é responsável por monitorar a carga de trabalho para todas as classes (seta 2). Ele contém armazenamento local (ex.: um banco de dados)

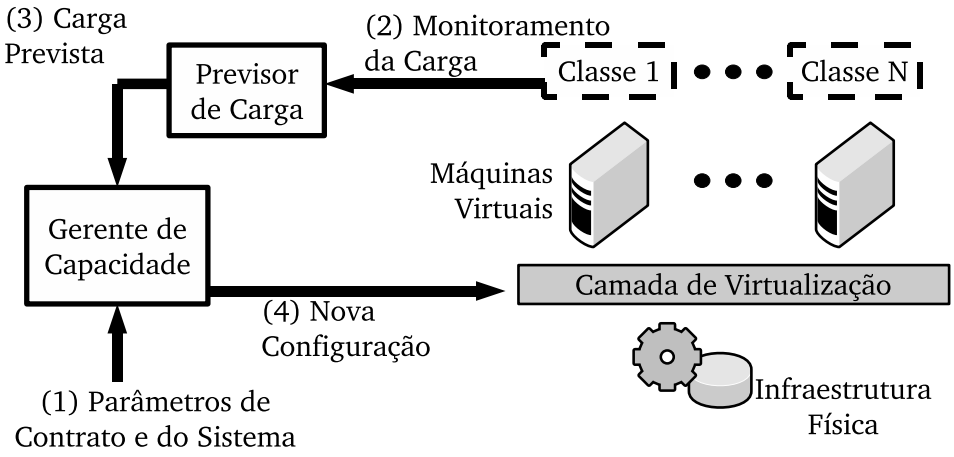


Figura 4.1: Gerenciamento Autônomo de Capacidade (perspectiva de 1 camada)

para registrar a carga passada, utilizando esses dados para caracterizar seu comportamento. Um dos vários métodos existentes de previsão de carga [2], tais como médias deslizantes (*moving averages*), deve ser utilizado para estimar a carga das classes no futuro próximo, baseado no comportamento observado no passado. A saída do previsor de carga é a taxa de chegada de requisições por segundo esperada para cada classe i durante o próximo intervalo, denotada λ_i, e utilizada pelo gerenciador de capacidade (seta 3) no cálculo da configuração da infra-estrutura para o próximo intervalo. O projeto e avaliação desse módulo está fora do escopo desta dissertação.

A nova configuração da infra-estrutura (seta 4) é composta de dois componentes. O primeiro e mais importante é a fração da capacidade disponível que deve ser alocada a cada máquina virtual e, conseqüentemente, sua classe i associada. Essa fração é denotada como f_i. O segundo parâmetro é a taxa de requisições que deve ser aceita para processamento na infra-estrutura, denotada por λ_i^{ac}. Caso a capacidade alocada à classe i seja insuficiente para processar todas as requisições previstas, λ_i^{ac} será menor que λ_i para evitar sobrecarga ou violações dos requisitos do SLA. A nova configuração do sistema é composta de f_i e λ_i^{ac}. O primeiro é enviado ao mecanismo de virtualização, que atualiza o mapeamento dos recursos físicos, e o segundo ao módulo de controle de admissão, que rejeita requisições excessivas.

A duração do intervalo de controle pode ser fixa ou variável, dependendo das características do sistema e da estabilidade das cargas de trabalho das classes hospedadas. Se as cargas apresentam grande variabilidade, o gerenciador de capacidade pode ser executado em intervalos regulares com duração na qual a carga não mude significativamente. Por outro lado, se as cargas variam pouco, o gerenciador de capa-

Figura 4.2: Componentes do Gerenciador de Capacidade

cidade pode ser executado sempre que a carga das aplicações tiver mudado significativamente desde a sua última execução, o que resulta em um intervalo de controle com duração variável. De qualquer forma, sua duração mínima é limitada pelo tempo que o gerenciador de capacidade leva para reconfigurar o sistema.

Os parâmetros de sistema são definidos na Seção 4.1.1 e os requisitos do SLA são definidos juntos do modelo de negócio na Seção 4.2. O gerenciador de capacidade, mostrado na Figura 4.2, é composto dos modelo de negócio e desempenho, que são combinados em um modelo de otimização responsável por encontrar a melhor configuração da infra-estrutura.

4.1.1 Parâmetros do Sistema

Os parâmetros do sistema são a utilização máxima planejada para cada máquina virtual, definida como ν_i, e a média do tempo de serviço das requisições de cada classe, definida como d_i^*.

A utilização de cada máquina virtual, denotada ρ_i, é definida como a razão entre a capacidade de processamento necessária para atender as requisições admitidas e a capacidade de processamento total da máquina virtual [54]. A utilização máxima planejada para cada máquina virtual, ν_i, limita o valor de ρ_i. Esse parâmetro é adicionado para evitar que a qualidade do serviço prestado ao cliente seja degradado devido a sobrecarga nos recursos. Essa limitação é necessária e garante estabilidade às máquinas virtuais, evitando que a média e a variância do tempo de resposta aumentem indefinidamente quando ρ_i se aproxima de 1.

O tempo de serviço médio de uma requisição da classe i, quando executando à capacidade total da infra-estrutura, é expresso por d_i^*. Todas as requisições de uma classe i são estatisticamente indistinguíveis, possuindo a mesma demanda média por serviço, d_i^*. Definindo a demanda por recurso na infra-estrutura compartilhada como

d_i e levando em consideração que a classe i recebe apenas a fração f_i da capacidade da infra-estrutura, temos $d_i = d_i^* / f_i$.

Os valores de d_i^* podem ser medidos num ambiente de teste pré-produção da aplicação na infra-estrutura física antes de ser colocada operacional. Como em [99], isso pode ser feito executando a aplicação para uma carga previamente armazenada ou gerada sinteticamente, enquanto as métricas de interesse são monitoradas. De forma similar, d_i^* pode ser inflacionado para levar em consideração sobrecargas de desempenho fixas impostas pela virtualização [10].

Note que os parâmetros de configuração do sistema e do SLA podem mudar sempre que ocorrerem alterações na infra-estrutura ou nos contratos entre o provedor e seus clientes: adição ou remoção de classes de aplicação na infra-estrutura, mudança de capacidade de uma camada, uma atualização na aplicação hospedada ou mudança nos requisitos de qualidade de serviço. O SLA, formalizado pelo contrato entre o provedor e seus clientes, é discutido na próxima seção.

4.2 Modelo de Negócio

A Internet oferece uma imensa quantidade de serviços que usualmente recebem carga baixa ou moderada, mas ocasionalmente recebem um surto de requisições excepcionalmente grande [15,22,61,67,102]. Exemplos comuns são os portais de notícias que esperam um número previsível de usuários em operação normal mas, sempre que algum evento especial acontece, são sobrecarregados com um surto de requisições para o sítio, mudando as necessidades de capacidade dramaticamente [12]. Esse fenômeno, conhecido como *flash crowd*, gera congestionamento na infra-estrutura de serviço, causando atrasos significativos aos usuários.

O modelo de negócio em [3,4] aborda a alta variabilidade das cargas de trabalho de serviços Web, propondo contratos com dois níveis de requisitos, que correspondem a dois modos de operação diferentes, a saber: normal e sobrecarregado. No modo de operação normal, os clientes contratam o nível de serviço que satisfaz suas necessidades para a maior parte do tempo. No modo sobrecarregado, um nível de serviço mais alto é estabelecido, até onde o provedor tem um incentivo para alocar capacidade extra a uma aplicação com o objetivo de acomodar picos ocasionais de carga.

Para o modo de operação normal, o SLA define uma taxa de processamento X_i^N

para cada classe i, que espera-se que o provedor atenda, dado que a taxa de chegadas de requisições seja alta o suficiente, isto é, $\lambda_i \geq X_i^N$. No caso de violações do SLA, o provedor concorda em reembolsar parte da cobrança pelo serviço aos clientes. Esse reembolso é proporcional à diferença entre X_i^N e a taxa *válida* de processamento atual, que é composta das requisições processadas que atenderam o requisito de tempo de resposta. O cálculo detalhado do valor do reembolso e a formulação matemática da taxa válida de processamento serão apresentados na Seção 4.2.1. Para o modo de operação sobrecarregado o SLA define $X_i^S \geq X_i^N$, a taxa de processamento até qual o cliente concorda em pagar uma recompensa ao provedor por servir requisições numa taxa maior que X_i^N. Recompensas também são proporcionais à taxa válida de processamento que *excede* X_i^N.

Contratos tradicionais, com um único alvo de desempenho, requerem que os clientes cujas cargas apresentam alta variabilidade (alta razão entre a carga média e o pico de carga) paguem pelo nível de serviço necessário para satisfazer o pico de carga durante toda a operação, mesmo que somente uma parte da capacidade pela qual o cliente paga seja realmente utilizada na maior parte do tempo. Do ponto de vista de negócio, a abordagem proposta em [3,4] é vantajosa para provedores, que podem oferecer planos de serviço mais atraentes por operarem com maior flexibilidade, e clientes, que pagam por capacidade extra apenas quando necessário, atendendo seu interesse em pagar apenas pelos recursos realmente utilizados [110].

A capacidade das máquinas virtuais em processar transações é limitada pelos requisitos de tempo de resposta do contrato SLA. Neste trabalho, consideramos um requisito sobre a cauda da distribuição do tempo de resposta que diz que uma requisição da classe i não pode exceder um dado limite R_i^{SLA} mais do que $\alpha_i \times 100\%$ das vezes. R_i^{SLA} é o limite do tempo de resposta aceitável para a aplicação e α_i é uma tolerância a violações desse tempo de resposta. Especificamente, o requisito de desempenho pode ser escrito como $P[R_i > R_i^{SLA}] \leq \alpha_i$, onde R_i é o tempo de resposta de uma requisição da classe i. Quanto maiores (menores) R_i^{SLA} e α_i, menos (mais) restritivo será o requisito de desempenho. Especificamente, essa restrição expressa o compromisso entre taxa de processamento e qualidade de serviço. Um valor menor de α_i garante que grande parte das requisições da classe i serão servidas com tempos de resposta curtos. Porém, para atender esse requisito menos requisições são aceitas dentro do sistema e a taxa de processamento é menor. Valores maiores para α_i viabilizam a admissão de um número maior de requisições na infra-estrutura e, por fim, maior taxa de processamento. Porém, requisições admitidas terão tempos de resposta

mais longos com maior frequência. Esse requisito atende demandas atuais de clientes [60] e é muito mais rico que a abordagem tradicional de prover garantias sobre a média do tempo de resposta.

Enfatizamos que essa abordagem não é restrita a apenas dois níveis (normal e sobrecarregado). A abordagem proposta pode ser estendida para contratos SLA com mais de dois níveis, especificando múltiplos alvos de desempenho para acomodar diferentes níveis de demandas para aplicações e clientes.

4.2.1 Cálculo do Lucro do Provedor

Esta seção descreve como as recompensas e reembolsos são computados e como o lucro da execução das aplicações é calculado no final de cada intervalo de controle.

A taxa *válida* de processamento é definida como X_i. Devido à lei do balanço de fluxos [54], a taxa total de processamento é igual a λ_i^{ac}, pois nenhuma requisição é perdida depois de admitida na infra-estrutura. Seja q_i a freqüência relativa das transações da classe i com tempos de resposta menores que R_i^{SLA}. Se o requisito sobre a cauda da distribuição do tempo de resposta foi violado, então $q_i < (1 - \alpha_i)$ e $X_i < \lambda_i^{ac}$. A taxa válida de processamento é dada pelas requisições que atenderam o requisito de desempenho mais a tolerância a falhas permitida: $X_i = q_i\lambda_i^{ac} + \alpha_i(q_i\lambda_i^{ac})$. No caso contrário, $q_i \geq (1 - \alpha_i)$, todas as transações processadas durante o intervalo atenderam ao requisito de desempenho e $X_i = \lambda_i^{ac}$.

Como dito anteriormente, reembolsos são feitos pelo provedor aos clientes para cada classe i sempre que $X_i \leq X_i^N$ devido a falta de capacidade na infra-estrutura para servir requisições. Quando $\lambda_i < X_i^N$, reembolsos ocorrem quando $X_i \leq \lambda_i$ e são proporcionais a $\lambda_i - X_i$. Por outro lado, se $\lambda_i > X_i^N$, o provedor deve atender pelo menos X_i^N requisições por segundo. Se esse requisito não for satisfeito, o reembolso será proporcional a $X_i^N - X_i$. Sumarizando, os reembolsos do provedor feitos ao cliente caso deixe de atender o requisito de taxa de processamento para o modo de operação normal, X_i^N, são proporcionais a $min(\lambda_i, X_i^N) - X_i$. A quantidade reembolsada pelo provedor ao cliente por unidade de taxa de processamento abaixo da necessária é dada por c_i. Desta forma, reembolsos resultantes do serviço de cada classe i ocorrem quando $X_i \leq X_i^N$, no valor de $c_i(min(\lambda_i, X_i^N) - X_i)$.

De forma similar, se $X_i > X_i^N$, o provedor capitaliza recompensas pagas pelo cliente por exceder o requisito de desempenho do modo de operação normal. A re-

compensa é proporcional à taxa de processamento que excede X_i^N, porém limitada ao valor de X_i^S. Denotamos por r_i a quantidade paga pelo cliente ao provedor por unidade de taxa de processamento acima de X_i^N. Desta forma, recompensas são pagas ao provedor para cada classe i quando $X_i > X_i^N$, no valor de $r_i(min(X_i, X_i^S) - X_i^N)$.

Dado esse modelo de negócio, o objetivo do provedor é maximizar o lucro total decorrente dos reembolsos e recompensas de todas as classes. O lucro de uma classe i é denotado por g_i e definido abaixo:

$$g_i = \begin{cases} -c_i \left(min(\lambda_i, X_i^N) - X_i\right) & X_i \leq X_i^N \\ r_i \left(min(X_i, X_i^S) - X_i^N\right) & X_i > X_i^N \end{cases} \tag{4.1}$$

O lucro total, calculado ao final de cada intervalo de controle, é dado por:

$$\sum_{i=1}^{N} g_i \tag{4.2}$$

4.3 Modelos de Desempenho e Otimização

O arcabouço apresentado em [3,4] propõe um modelo de desempenho que modela infra-estruturas com uma camada, onde cada classe de aplicação é modelada como uma fila M/M/1 ou M/G/1/PS. O modelo de desempenho provê estimativas da utilização da máquina virtual de cada classe e da cauda da distribuição do tempo de resposta, que segue uma distribuição exponencial. Assim, a probabilidade de violação do SLA é dada por:

$$P[R_i \geq R_i^{SLA}] = e^{-(1/d_i - \lambda_i^{ac})R_i^{SLA}} \tag{4.3}$$

O modelo de otimização proposto em [3, 4] objetiva maximizar o lucro do provedor, definido pelo modelo de negócio e expresso pela Equação 4.2. Suas variáveis de decisão, f_i e λ_i^{ac}, estão sujeitas a várias restrições, dentre elas o limite sobre a utilização das máquinas virtuais, $\rho_i \leq \nu_i$, e o requisito sobre a cauda da distribuição do tempo de resposta, $P[R_i \geq R_i^{SLA}] \leq \alpha_i$.

Estes modelos foram apresentados brevemente pois são um caso específico dos novos modelos propostos no Capítulo 5 para infra-estruturas multicamadas.

5 *Gerenciamento de Capacidade de Infra-Estruturas Multicamadas Frente a Ataques de Segurança, Custos e Restrições de Energia*

O arcabouço apresentado no Capítulo 4 trata apenas de infra-estruturas com uma camada. Nesta seção apresentamos nossas extensões para capturar infra-estruturas multicamadas, ataques de segurança, custos e restrições de energia. O novo arcabouço de gerenciamento proposto neste trabalho herda o funcionamento cíclico (Seção 4.1) e o modelo de negócio (Seção 4.2) apresentados anteriormente em [3,4]. As extensões propostas a esses aspectos para múltiplas camadas são discutidos na Seção 5.1. O novo modelo de desempenho utilizado para estimar analiticamente o desempenho de infra-estruturas multicamadas é apresentado na Seção 5.2. O modelo de otimização que combina os modelos de negócio e desempenho para encontrar a alocação de capacidade que maximiza o lucro do provedor é apresentado na Seção 5.3.

Ao final da Seção 5.3, o leitor já terá compreendido o funcionamento do arcabouço durante períodos normais de operação. As Seções 5.4 e 5.5 estendem os modelos de negócio, desempenho e otimização para períodos nos quais a infra-estrutura opera sob ataques de segurança e restrições de energia, respectivamente.

A notação completa de todos os parâmetros e variáveis do arcabouço é apresentada nas Tabelas 5.1, 5.2 e 5.3 ao final deste capítulo.

5.1 Extensões para Múltiplas Camadas

O arcabouço autônomo apresentado nesta dissertação herda do trabalho prévio em [3,4] o controle cíclico (Seção 4.1), funcionando da mesma forma como mostrado na Figura 4.1. A cada intervalo de controle, o gerenciador de capacidade utiliza

os parâmetros do sistema, os contratos SLA e a previsão de carga para o próximo intervalo para calcular a melhor alocação de capacidade da infra-estrutura.

Todos os parâmetros do sistema são estendidos para múltiplas camadas. A utilização máxima planejada para a máquina virtual associada à classe i na camada j é denotada $\nu_{i,j}$. A demanda média da classe i quando executando na capacidade total da camada j é definida como $d^*_{i,j}$. De forma similar, a capacidade alocada, a utilização e a demanda média por recursos da classe i na infra-estrutura compartilhada da camada j, são denotadas $f_{i,j}$, $\rho_{i,j}$ e $d_{i,j}$, respectivamente.

5.2 Modelo de Desempenho

Esta seção descreve um modelo analítico de filas que estima as métricas de desempenho utilizadas pelo gerenciador de capacidade. As métricas estimadas são a utilização dos recursos por cada classe i em cada camada j, $\rho_{i,j}$, e a probabilidade de violação do requisito do SLA sobre o tempo de resposta, $P[R_i \geq R_i^{SLA}]$. Como nenhuma requisição é descartada depois de admitida na infra-estrutura, a taxa de processamento de cada classe i é igual a λ_i^{ac}, devido à lei do equilíbrio de fluxos [54].

O modelo proposto supõe que chegadas de requisições para cada classe são dadas por um processo de Poisson [61, 79, 102]. Requisições da classe i aceitas para processamento chegam na primeira camada com taxa λ_i^{ac}, e deixam o sistema após visitar a camada j com probabilidade $p_{i,j}$. Como discutido na Seção 3.4, supõe-se que requisições de cada classe i possuem tempos de serviço exponencialmente distribuídos em cada camada j, com média $d^*_{i,j}$. Nesse contexto, a distribuição exata do tempo de resposta pode ser calculada, resultando em um modelo preciso que não poderia ser derivado para outras distribuições. O impacto do relaxamento dessa premissa será avaliado na Seção 6.5. Modelos preliminares para outras distribuições de tempo de serviço e chegada de requisições são discutidos no Apêndice B, deixando seu aperfeiçoamento para trabalhos futuros.

Dadas essas premissas, o sistema virtualizado multicamadas é modelado como N redes, uma para cada classe, com K filas em seqüência, como mostrado na Figura 5.1. A máquina virtual dedicada a cada classe em cada camada é representada como uma fila M/M/1 com escalonamento FCFS (primeiro-a-chegar primeiro-a-sair) [54]. Argumentamos que escalonamento FCFS é um modelo mais preciso de servidores atuais do que compartilhamento de processador (*processor sharing*), pois servidores

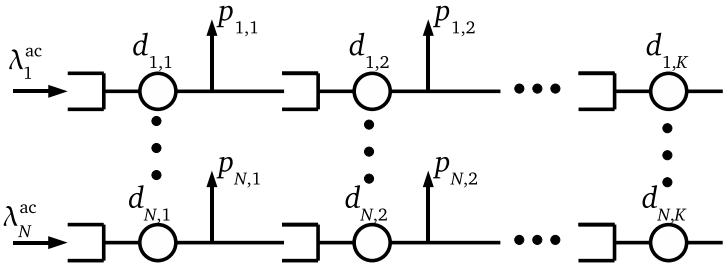


Figura 5.1: Redes de Filas do Modelo de Desempenho

atendem requisições em seqüência, evitando concorrência em dispositivos de entrada e saída tipicamente seqüenciais e alocação simultânea de recursos limitados como memória. Filas M/M/1 com escalonamento FCFS são comumente usadas como um modelo razoável para centros de serviços transacionais [61, 67, 99, 102].

Como discutido na Seção 3.4, uma requisição da classe i tem um tempo de resposta no sistema igual à soma dos tempos de residência em cada uma das filas, que são supostas independentes umas das outras. Essa premissa de independência entre as filas troca precisão por simplicidade e aplicabilidade.

Dadas as probabilidades de roteamento de uma requisição, $p_{i,j}$, a taxa efetiva de chegada de requisições da classe i na camada j, $\lambda_{i,j}^e$, é dada por:

$$\lambda_{i,j}^e = \lambda_i^{ac} \prod_{k=1}^{j-1} (1 - p_{i,k}) \tag{5.1}$$

Note que $\lambda_{i,1}^e = \lambda_i^{ac}$. Além disso, se $p_{i,j} = 0$ para $i \in [1, N]$ e $j \in [1, K)$, então $\lambda_{i,j}^e = \lambda_i^{ac}$ para $i \in [1, N]$ e $j \in [1, K]$.

A utilização da máquina virtual da classe i na camada j, $\rho_{i,j}$, pode ser estimada como o produto do seu tempo médio de serviço pela sua taxa efetiva de chegada de requisições [54]:

$$\rho_{i,j} = \lambda_{i,j}^e d_{i,j} \tag{5.2}$$

O componente mais desafiador do modelo de desempenho é a estimativa da probabilidade de uma requisição da classe i violar o requisito de tempo de resposta do SLA, $P[R_i \geq R_i^{SLA}]$. O tempo de resposta de uma requisição da classe i é dado pela soma dos tempos de residência em cada camada, $R_{i,j}$, ou seja, $R_i = \sum_{j=1}^{K} R_{i,j}$. Note

que $R_{i,j}$ é o tempo de resposta de uma fila M/M/1, que é exponencialmente distribuído com parâmetro $\gamma_{i,j} = (1/d_{i,j}) - \lambda^{e}_{i,j}$ [54]. Além disso, a soma de K variáveis independentes exponencialmente distribuídas segue uma distribuição hipoexponencial [98]. Assim, a probabilidade do tempo de resposta de uma requisição da classe i violar o requisito de tempo de resposta é igual à probabilidade de uma variável com distribuição hipoexponencial ser maior que R_i^{SLA}. Em outras palavras, $P[R_i \geq R_i^{SLA}]$ é igual ao complemento da distribuição acumulada de uma variável hipoexponencial com parâmetros $\gamma_{i,j}$, dado por:

$$P[R_i \geq R_i^{SLA}] = \sum_{j=1}^{K} \left(\prod_{k=1,k\neq j}^{K} \frac{\gamma_{i,k}}{\gamma_{i,k} - \gamma_{i,j}} \right) e^{-\gamma_{i,j} R_i^{SLA}} \tag{5.3}$$

Devido ao fato desse modelo de desempenho baseado em filas capturar o paralelismo que pode existir em sistemas multicamadas, espera-se que a Equação 5.3 seja mais precisa do que aplicar qualquer uma das aproximações de camada única apresentadas em [3, 4] à nossa plataforma multicamadas alvo. Adaptando o modelo baseado em camada única para o cenário multicamadas, uma aproximação possível é substituir o parâmetro d_i na Equação 4.3 por $\sum_{j=1}^{K} d_{i,j}$.

Em contrapartida, a Equação 5.3 é mais complexa que a Equação 4.3 utilizada em [3,4], o que pode comprometer a convergência e o tempo de solução do modelo de otimização (Seção 5.3.1). Assim, também é considerada uma aproximação derivada da Desigualdade de Chebyshev [54], que provê o seguinte limite superior para a probabilidade de violações da classe i:

$$P[R_i \geq R_i^{SLA}] \leq \frac{VAR[R_i]}{(R_i^{SLA} - E[R_i])^2} \tag{5.4}$$

Onde $E[R_i]$ e $VAR[R_i]$ são a média e variância do tempo de resposta do sistema para a classe i, respectivamente, e são iguais às somas das métricas correspondentes de cada componente exponencial da distribuição hipoexponencial do tempo de resposta [98]. Em outras palavras,

$$E[R_i] = \sum_{j=1}^{K} E[R_{i,j}]$$

e

$$VAR[R_i] = \sum_{j=1}^{K} VAR[R_{i,j}]$$

Esta dissertação não considera a Desigualdade de Markov [54], mais simples, que depende apenas da média do tempo de resposta, pois este limite superior para a probabilidade de violação é geralmente muito impreciso, levando a decisões de alocação excessivamente conservadoras e menos efetivas, conforme observado em [3, 4].

5.3 Modelo de Otimização

O modelo de otimização descrito nesta seção é o componente central do gerenciador de capacidade proposto. No início de cada intervalo de controle, utilizando a carga de trabalho prevista pelo preditor e estimativas derivadas do modelo de desempenho, o modelo calcula a alocação de capacidade e taxa de requisições admitidas que maximizam o objetivo de negócio do provedor. O objetivo de negócio do provedor é maximizar o lucro total obtido dos reembolsos e recompensas para todas as classes, dado pela Equação 4.2.

O modelo de otimização é mostrado na Figura 5.2. A sua principal variável de decisão é o vetor $f_{i,j}$, a fração da capacidade da camada j associada à máquina virtual responsável pela classe i. A segunda variável de decisão é a taxa de requisições que pode ser admitida no sistema, λ_i^{ac}, e ainda satisfazer os requisitos de qualidade do contrato SLA. O conjunto de restrições, discutidos a seguir, será estendido na Seção 5.5 para capturar restrições de energia.

A *restrição (a)* diz que a taxa válida de processamento do sistema é igual à taxa de requisições admitidas. Isto deve-se primeiro à lei do balanço de fluxos [54], já que todas as requisições aceitas são efetivamente servidas pela infra-estrutura. Em segundo lugar, isto deve-se à restrição (h), que força todas as requisições admitidas a atenderem o requisito de qualidade do SLA.

A *restrição (b)* estabelece um limite inferior óbvio à variável de decisão λ_i^{ac} e limita a taxa de requisições admitidas para cada classe à taxa de chegadas prevista e à taxa máxima de processamento até a qual o provedor pode capitalizar quando a classe opera em modo sobrecarregado (vide Seção 4.2).

A *restrição (c)* define a taxa efetiva de chegada para cada classe em cada camada.

$$
\begin{aligned}
max. \quad & \sum_{i=1}^{N} g_i(\lambda_i^{ac}) & & \\
s.a. \quad & X_i = \lambda_i^{ac} & \forall i \in [1,N] \quad & (a) \\
& 0 \leq \lambda_i^{ac} \leq min(\lambda_i, X_i^S) & \forall i \in [1,N] \quad & (b) \\
& \lambda_{i,j}^{e} = \lambda_i^{ac} \prod_{k=1}^{j-1} (1 - p_{i,k}) & \forall i \in [1,N], \forall j \in [1,K] \quad & (c) \\
& f_{i,j} \geq 0 & \forall i \in [1,N], \forall j \in [1,K] \quad & (d) \\
& \sum_{i=1}^{N} f_{i,j} \leq 1 & \forall j \in [1,K] \quad & (e) \\
& d_{i,j} = d_{i,j}^{*} / f_{i,j} & \forall i \in [1,N], \forall j \in [1,K] \quad & (f) \\
& \rho_{i,j} = \lambda_{i,j}^{e} d_{i,j} \leq \nu_{i,j} & \forall i \in [1,N], \forall j \in [1,K] \quad & (g) \\
& P[R_i \geq R_i^{SLA}] \leq \alpha_i & \forall i \in [1,N] \quad & (h)
\end{aligned}
$$

Figura 5.2: Modelo de Otimização do Gerenciador de Capacidade

A *restrição (d)* impõe um limite inferior à variável de decisão $f_{i,j}$. A *restrição (e)* limita a capacidade alocada a 100% da capacidade disponível na infra-estrutura. A *restrição (f)* define o tempo médio de serviço da classe i em sua máquina virtual na camada j, baseado na capacidade alocada, $f_{i,j}$, e no tempo médio de serviço medido em um ambiente pré-produção, $d_{i,j}^{*}$.

A *restrição (g)* define e limita a utilização da máquina virtual da classe i na camada j à utilização máxima planejada, $\nu_{i,j}$, evitando degradação da qualidade de serviço devido a sobrecargas.

A *restrição (h)* expressa o requisito de tempo de resposta do SLA. Duas variantes do modelo são criadas através do uso das expressões nas Equações 5.3 e 5.4, para calcular a probabilidade de violação do requisito de tempo de resposta, $P[R_i \geq R_i^{SLA}]$.

A função objetivo expressa o objetivo de negócio do provedor, dado pela soma, para todas as classes, do lucro resultante dos reembolsos e recompensas individuais, expresso na Equação 4.1. Note que g_i aumenta com λ_i^{ac}. Porém, λ_i^{ac} não pode aumentar indefinidamente pois é limitada pela taxa de chegadas de requisições e pela taxa máxima até onde o cliente concorda em pagar uma recompensa para o provedor (restrição (b)), pela utilização máxima planejada dos recursos (restrição (g)), e, acima de tudo, pelo requisito do tempo de resposta do SLA (restrição (h)). Essas duas últimas

restrições indiretamente ligam os valores de λ_i^{ac} às principais variáveis de decisão $f_{i,j}$. Isto é, para aumentar a taxa de requisições admitidas de uma aplicação é necessário alocar mais capacidade a ela.

O modelo de otimização apresentado pode ser acoplado a outros modelos de desempenho, como os discutidos no Apêndice B, para prever as métricas de desempenho $\rho_{i,j}$ e $P[R_i \geq R_i^{SLA}]$ das restrições (g) e (h) com diferentes níveis de precisão.

5.3.1 Problemas de Convergência

Esta seção apresenta as dificuldades encontradas para garantir a convergência e otimalidade do modelo proposto. A otimalidade de um modelo garante que a solução encontrada é ótima, isto é, não existe outros valores das variáveis de decisão para os quais o valor da função objetivo seja maior que o da solução encontrada. A convergência diz respeito ao fato de vários solucionadores, em repetidas execuções, sempre convergirem para a solução ótima. Nesse âmbito, os maiores desafios para otimalidade e convergência do modelo de otimização são a função objetivo linear por partes e a restrição (h) sobre o requisito do tempo de resposta, que é não-linear.

A solução de problemas de otimização lineares por parte geralmente incorre no uso de variáveis de decisão binárias, o que torna o problema de otimização muito mais complexo [35]. Porém, se a função objetivo linear por partes for côncava, então ela pode ser expressa como um conjunto de restrições lineares que podem ser facilmente resolvidas [35]. O modelo proposto terá uma função objetivo côncava se $c_i \geq r_i$ para $i \in [1, N]$. Esta dissertação considera essas configurações pois o atendimento de requisições extras recebidas quando o serviço opera em modo sobrecarregado é secundário, daí, $c_i \geq r_i$. Além disso, funções objetivo côncavas avaliadas em restrições convexas são condições necessárias e suficientes para que um ótimo local seja também global [84]. Nesse contexto, algoritmos de descida, que diminuem o valor da função objetivo a cada iteração, têm convergência garantida [84]. Outros tipos de funções caem em outra vertente da otimização, para as quais não existe algoritmo que garantidamente encontra a solução ótima. Para o caso onde a função objetivo não é côncava, duas alternativas de aproximação podem ser utilizadas. Primeiro, uma variável binária pode ser usada para combinar reembolsos e recompensas em apenas uma expressão, resultando num modelo não linear com variáveis inteiras. Segundo, pode-se aproximar a função objetivo por um polinômio.

A probabilidade de violação do tempo de resposta do SLA (restrição (h)) derivada da Desigualdade de Chebyshev (Equação 5.4), apesar de ser não linear no domínio das variáveis de decisão, é uma função convexa razoavelmente simples. Assim, o modelo de otimização resultante pode ser facilmente resolvido.

A expressão derivada da distribuição hipoexponencial (Equação 5.3), por outro lado, é muito mais complexa, resultando em um modelo de otimização muito mais desafiador. Em particular, a distribuição não está definida sempre que dois dos seus parâmetros $\gamma_{i,j}$ possuem valores idênticos, tornando o problema sem solução. Várias estratégias podem ser utilizadas para remediar esse problema. Primeiro, a distribuição hipoexponencial pode ser aproximada por um polinômio, o que leva a uma perda de precisão. Segundo, como a distribuição hipoexponencial não é definida para regiões bem conhecidas, pode-se executar uma instância do solucionador para cada região complementar do domínio do problema onde a função é bem definida e depois tomar como ótimo global a maior solução encontrada. Esta estratégia foi implementada e testada com sucesso para um número pequeno de camadas (2) e classes de aplicação (até 4). Porém, essa solução não escala para um grande número de classes e camadas, pois o número de regiões complementares do domínio para as quais precisamos solucionar o modelo é igual a $(K!)^N$.

Outra alternativa é aproximar termos da distribuição hipoexponencial com parâmetros iguais por uma distribuição de Erlang. Essa aproximação é assintoticamente exata, pois a soma de variáveis exponenciais igualmente distribuídas possui distribuição de Erlang [98]. Esta abordagem foi escolhida devido à sua melhor escalabilidade, discutida na Seção 6.1.

Com as modificações descritas acima, todas as execuções do modelo, para diferentes solucionadores, convergem para a mesma solução e satisfazem condições de Karush-Kuhn-Tucker necessárias para que a solução seja ótima [84].

Por fim, a restrição (h) sobre o tempo de resposta tem um efeito indesejável sobre as soluções encontradas em alguns cenários específicos. Se uma classe de aplicação i não é rentável, ou a infra-estrutura não tem capacidade para atendê-la, o solucionador pode decidir que é melhor não admitir nenhuma de suas requisições ($\lambda_i^{ac} = 0$). Nesse caso, nenhuma capacidade ($f_{i,j}$) deveria ser alocada à classe i. Porém, a fórmula que combina a restrição (h) com as Equações 5.3 e 5.4 força a capacidade alocada à classe i, $f_{i,j}$, ser maior do que zero mesmo que nenhuma requisição seja admitida.

Para tratar esse problema, é inicialmente resolvida uma instância do modelo com

todas as N classes. Depois, são consideradas N novas instâncias do problema de otimização, removendo uma classe diferente da instância original por vez. Considera-se que as classes removidas possuem $\lambda_i^{ac} = f_{i,j} = 0$. Caso alguma dessas novas instâncias resulte em solução superior à melhor solução atual, a melhor solução é substituída e a pesquisa continua recursivamente a partir daquela instância. Esse método pode ser visto como uma pesquisa em árvore do tipo ramificar e limitar (*branch and bound*). Para limitar ainda mais a pesquisa, esta pode ser cancelada se for encontrada uma solução próxima o suficiente, por exemplo 90%, de um limite superior para a solução ótima. Esse limite pode ser calculado a partir do próprio modelo de otimização desconsiderando a restrição (h).

A implementação do modelo de otimização em AMPL [35], uma linguagem de modelagem para programação matemática, bem como o pseudocódigo da pesquisa em árvore descrita acima são discutidos no Apêndice C.

5.4 Gerenciamento Frente a Ataques de Segurança

Os modelos de negócio, desempenho e otimização descritos nas Seções 4.2, 5.2 e 5.3, respectivamente, compõem o novo arcabouço. Juntos, eles resolvem o problema de gerenciamento de capacidade para ambientes multicamadas em períodos de operação normal, desconsiderando custos de energia. Esta seção estende os modelos de negócio e otimização apresentados anteriormente para capturar o impacto de ataques de segurança nas decisões de gerenciamento.

O objetivo principal é desenvolver soluções simples que permitam entender alguns dos principais compromissos do gerenciamento de capacidade de uma infraestrutura compartilhada sob ataques de segurança. Devido ao impacto negativo desses ataques no lucro do provedor e na qualidade do serviço provido ao cliente, são considerados, além do interesse do provedor, os interesses dos clientes. Do ponto de vista do cliente, os três componentes do SLA – taxa válida de processamento, tempo de resposta e custo por requisição – são avaliados para as diferentes soluções de contratos SLA adaptativos que o cliente pode optar por usar durante um ataque.

A modelagem dos ataques no arcabouço de gerenciamento de capacidade é apresentada na Seção 5.4.1. A Seção 5.4.2 discute as diferentes soluções incluindo os contratos SLA adaptativos.

5.4.1 Modelagem de Ataques no Arcabouço

Esta dissertação foca em ataques de segurança que visam comprometer a qualidade do serviço recebido por usuários de uma *aplicação* através da sobrecarga dos recursos utilizados pela aplicação específica por meio de, por exemplo, um fluxo de requisições ilegítimas. Essas requisições ilegítimas, apesar de consumir recursos da infra-estrutura, não geram retorno financeiro, lucro, para o provedor. Esses ataques também impactam a qualidade do serviço provido ao cliente, causando diminuição da taxa válida de processamento e aumento do tempo de resposta, devido à capacidade da infra-estrutura ocupada com requisições ilegítimas.

Exemplos de ataques que têm como alvo uma aplicação específica incluem inundação de requisições HTTP [51] e SPAM [43]. A detecção desses ataques pode ser feita através da construção de um perfil da carga típica, para que desvios significativos do comportamento esperado possam ser detectados e caracterizados [39,62]. Minimizar o impacto desses ataques na infra-estrutura é particularmente interessante pois, mesmo que um ataque possa ser detectado, requisições ilegítimas não podem ser individualmente bloqueadas antes de serem processadas, pois são idênticas e indistinguíveis de uma requisição legítima. Este trabalho não objetiva detectar ou combater esses ataques, mas capturar seu impacto nas decisões de gerenciamento e, em última instância, nos interesses do provedor e dos clientes.

Ataques de segurança que atingem uma aplicação específica são alvos interessantes de análise no contexto de gerenciamento de capacidade exatamente porque este decide a alocação de capacidade e taxa de processamento de cada aplicação. Quando alguma aplicação está sob ataque, o gerenciador de capacidade pode configurar a infra-estrutura de forma a reduzir a degradação e o impacto do ataque. Outros tipos de ataques de segurança que atingem a infra-estrutura como um todo, como inundação de banda de rede, estão fora do escopo de atuação de gerenciadores de capacidade, pois normalmente requerem mecanismos de defesa implantados nos roteadores da Internet [38].

Para capturar o impacto primário de um ataque à classe de aplicação i, o arcabouço é estendido como segue. A taxa total de requisições para a classe i, λ_i, é separada em λ_i^+, a taxa de requisições legítimas, e λ_i^-, a taxa de requisições ilegítimas. Isto é, $\lambda_i = \lambda_i^+ + \lambda_i^-$. Requisições legítimas são feitas por usuários reais e resultam em lucro para o provedor, enquanto requisições ilegítimas são decorrentes de um ataque e não resultam em lucro.

Como as requisições ilegítimas não podem ser distinguidas das legítimas e algumas são admitidas no sistema, a taxa de processamento da classe i é decomposta em legítima e ilegítima, dadas por $\lambda_i^{ac}(\lambda_i^+/\lambda_i)$ e $\lambda_i^{ac}(\lambda_i^-/\lambda_i)$, respectivamente. Apenas requisições legítimas são consideradas para fins do cálculo da taxa de processamento válida. Assim, o serviço da fração legítima das requisições da classe i implica em um custo extra ao provedor, pois requisições ilegítimas também serão admitidas no sistema, consumindo recursos sem gerar lucro. Isso permite a criação de soluções de gerenciamento de capacidade interessantes a explorar quanto aos compromissos entre os interesses do provedor e dos clientes, quando uma ou mais aplicações são alvos deste tipo de ataque.

Requisições legítimas e ilegítimas possuem o mesmo tempo médio de serviço em cada camada ($d_{i,j}^*$), como é de se esperar de ataques que procuram imitar o comportamento de usuários reais, como em [51]. Uma discussão breve da modelagem de ataques com demandas heterogêneas, como ataques semânticos [30], é apresentada no Apêndice A.

Dado $\phi_i = \lambda_i^+/\lambda_i$, a fração de requisições da classe i sobre as quais o provedor capitaliza, o lucro do provedor adquirido pelo serviço prestado à classe i pode ser redefinido como:

$$g_i^s = \begin{cases} -c_i\left(min(\lambda_i, X_i^N) - \phi_i X_i\right) & \phi_i X_i \leq X_i^N \\ r_i\left(min(\phi_i X_i, X_i^S) - X_i^N\right) & \phi_i X_i > X_i^N \end{cases} \tag{5.5}$$

5.4.2 Estratégias para Gerenciamento de Capacidade

Nesta dissertação são propostas e avaliadas quatro soluções de gerenciamento de capacidade criadas a partir das extensões do arcabouço discutidas na Seção 5.4.1. Cada solução define um conjunto diferente de fatores que são considerados durante o gerenciamento de capacidade. As duas primeiras objetivam avaliar os compromissos relevantes e as vantagens da modelagem explícita de ataques de segurança no arcabouço de gerenciamento de capacidade. São elas:

Indiferente aos Ataques (IA): Nesta solução, o provedor não está ciente que a classe de aplicação i está sob ataque. Isto é, λ_i^- é desconhecido. A tarefa de gerenciamento de capacidade é feita usando o arcabouço original, e, em particular, estimativas do lucro do provedor são calculadas usando a Equação 4.1, apesar

do lucro ser dado efetivamente pela Equação 5.5.

Ciente dos Ataques (CA): Nesta solução, o gerenciamento de capacidade é feito usando o novo arcabouço estendido descrito na Seção 5.4.1. Em particular, estimativas do lucro do provedor são calculadas usando a Equação 5.5.

Também são consideradas duas outras soluções baseadas em contratos SLA adaptativos. Durante um ataque, o cliente e o provedor podem concordar em usar um contrato SLA adaptativo. Esses SLA adaptativos enriquecem o modelo de negócio, permitindo maior flexibilidade. Face a um ataque, o cliente vitimado pode concordar em pagar mais por cada requisição legítima atendida dentro do SLA original, ou relaxar seus requisitos de desempenho. SLA adaptativos podem beneficiar ambos clientes, que terão mais requisições legítimas atendidas, e provedores, que irão obter maior lucro ou mais flexibilidade. Assim, esses contratos podem ajudar a encontrar um melhor compromisso no caso de interesses conflitantes. As duas próximas soluções aplicam essas estratégias, possibilitando avaliar seus principais compromissos e potenciais benefícios.

Custo Adaptativo (CA-C): Face a um ataque, o cliente vitimado pode concordar em pagar um custo por requisição legítima atendida que é inflacionado de acordo com o ataque. O custo inflacionado por requisição legítima pago pelo cliente, c_i^s, é dado por $c_i^s = c_i(1 + u_i^s(\lambda_i^- / \lambda_i^+))$, onde u_i^s define o valor que o cliente aceita pagar a mais pelas requisições ilegítimas. A recompensa por requisição legítima servida em excesso de X_i^N, r_i^s, é calculada da mesma forma. O inflacionamento descrito acima é equivalente a dizer que, no total, o cliente paga o custo original, c_i e r_i, por todas requisições legítimas e uma fração u_i^s das requisições ilegítimas atendidas. Quanto maior o valor de u_i^s, maior o incentivo ao provedor para dedicar capacidade à vítima do ataque. No limite, quando $u_i^s = 1$, o cliente paga por todas as requisições ilegítimas.

Tempo de Resposta Adaptativo (CA-R): Nesta solução, o cliente concorda em relaxar seu requisito do tempo de resposta, R_i^{SLA}, por um fator proporcional ao peso do ataque, λ_i^- / λ_i. O requisito de tempo de resposta durante ataques, $R_i^{SLA,s}$, é definido como $R^{SLA,s} = R_i^{SLA}(1 + w^s(\lambda^- / \lambda_i)$, onde w^s é uma constante que define quanto o requisito do tempo de resposta pode ser relaxado. Com o relaxamento do requisito do tempo de resposta, o provedor é capaz de atender

um maior número de requisições, o que aumenta a taxa válida de processamento e serve de incentivo para o provedor alocar mais capacidade à vítima.

O arcabouço de gerenciamento apresentado não é limitado apenas às alternativas apresentadas, e pode ser utilizado com outros SLA adaptativos. O projeto de tais contratos é deixado para trabalhos futuros.

5.5 Custos e Restrições de Energia

Esta seção apresenta as extensões propostas para incluir custos e restrições de energia, cada vez mais importantes atualmente [41], nas decisões de gerenciamento de capacidade. Como na Seção 5.4, o foco é desenvolver soluções simples para compreender os principais compromissos resultantes de custos e restrições de energia. A modelagem desses aspectos é apresentada na Seção 5.5.1. Diferentes soluções de gerenciamento de capacidade, inclusive com contratos SLA adaptativos, são propostas na Seção 5.5.2.

5.5.1 Modelagem de Energia no Arcabouço

Os custos de energia incluem os gastos que o provedor tem ao manter a infra-estrutura ligada. Infra-estruturas de hospedagem e seus sistemas de resfriamento podem consumir vários megawatts de potência [23]. A adição de custos de energia nas decisões de gerenciamento de capacidade resulta em um novo compromisso não trivial entre o custo derivado da alocação e do uso de uma certa quantidade de recursos a uma aplicação e o lucro resultante de suas requisições. Para reduzir custos de energia, o provedor pode desligar uma fração da infra-estrutura quando a utilização dos recursos estiver baixa, concentrando a carga de trabalho nos recursos que ficaram ligados e, conseqüentemente, aumentando suas utilizações. Para cada classe, se o custo de energia resultante da alocação de capacidade a ela for maior do que o lucro obtido servindo suas requisições, o provedor pode decidir por desligar parte da infra-estrutura e diminuir a taxa de processamento da classe de aplicação.

A infra-estrutura pode estar sujeita a períodos de restrições de energia, isto é, períodos quando a disponibilidade de energia para o provedor ficará comprometida [22]. Variações de disponibilidade de energia, planejadas ou não, podem ser devidas a problemas na rede de distribuição, falta de reservas energéticas ou outros

fatores. Por outro lado, durante restrições de energia a redução do consumo de energia, e conseqüentemente na alocação total de recursos, é compulsória. Logo, é função do gerenciador de capacidade calcular a melhor alocação da capacidade restante disponível entre as aplicações, minimizando a degradação do lucro do provedor. A modelagem de restrições de energia no arcabouço é ainda mais crítica para períodos de carga pesada, onde os prejuízos podem ser maiores.

O arcabouço é estendido através da definição de e_j, o custo de energia, por unidade de tempo, decorrente da operação da camada j em sua capacidade total. A unidade de e_j é custo (em moeda financeira) por segundo. O custo total de energia por unidade de tempo associado a cada camada j é então dado por $e_j \sum_{i=1}^{N} f_{i,j}$.

A primeira premissa feita por esta modelagem é que a capacidade alocada em cada camada é contínua. Isso é uma simplificação, pois a quantidade de processadores em uma infra-estrutura física é discreta, e mesmo tecnologias de controle dinâmico de voltagem de processadores [11,26] são limitadas a um conjunto finito de possíveis valores de voltagem. Porém, mesmo simplificado, esse modelo permite capturar os compromissos principais introduzidos pela inclusão dos custos de energia. Em uma implantação real, a quantidade de recursos (servidores) a serem desligados poderia ser escolhida como o valor discreto inferior mais próximo do valor contínuo calculado pelo modelo.

A outra premissa dessa modelagem é que os custos de energia aumentam linearmente com a capacidade alocada. Isso deve-se ao fato de que o consumo de energia é uma função do número de componentes ligados. No caso de infra-estruturas heterogêneas constituídas por diferentes tipos de *hardware*, ou métodos de controle dinâmico de voltagem onde a capacidade de processamento (freqüência do processador) é função do cubo da potência utilizada [23], essa modelagem é uma aproximação. No caso de camadas com componentes heterogêneos, o arcabouço proposto ainda poderia operar considerando primeiro os recursos com menor razão entre poder de processamento e consumo de energia. A avaliação dessa aproximação e soluções mais sofisticadas são deixadas como trabalhos futuros.

Essa modelagem de energia já foi adotada em outros trabalhos prévios [10], embora esta dissertação seja o primeiro esforço para sua aplicação em plataformas multicamadas e requisitos sobre a cauda da distribuição do tempo de resposta.

A expressão do lucro do provedor para cada aplicação i quando custos de energia são considerados, g_i^e, é então dada por:

$$g_i^e = g_i - \sum_{j=1}^{K} e_j f_{i,j} \tag{5.6}$$

Onde g_i foi definido na Equação 4.1.

As Equações 5.5 e 5.6 podem ser combinadas para expressar o lucro do provedor quando a infra-estrutura está sob ataque de segurança *e* custos de energia são considerados, através da substituição de g_i por g_i^s.

Para capturar explicitamente restrições de energia, foi adicionado ao modelo de otimização mais uma restrição sobre a capacidade total alocada através de todas as camadas da infra-estrutura:

$$\sum_{j=1}^{K} \left(s_j \sum_{i=1}^{N} f_{i,j} \right) \leq C \tag{5.7}$$

Onde C é a capacidade total disponível na infra-estrutura e a constante s_j normaliza a capacidade de cada camada com relação à capacidade total C, necessária quando camadas possuem diferentes capacidades de processamento. Alternativamente, as constantes s_j podem ser interpretadas como a fração de C que pertence à camada j.

5.5.2 Estratégias para Gerenciamento de Capacidade

São consideradas três soluções distintas para avaliação de custos e restrições de energia. As duas primeiras possibilitam a avaliação dos compromissos e benefícios da modelagem de custos e restrições de energia no arcabouço. A última apresenta um contrato SLA adaptativo para períodos de restrição de energia.

Indiferente à Energia (IE): Nesta solução, o gerenciamento de capacidade usa o arcabouço original, utilizando a Equação 4.1 para estimar o lucro de cada classe, e não captura custos de energia. Além de ignorar custos de energia, esta solução não lida com restrições de energia por não incluir a Equação 5.7 nas restrições do modelo de otimização.

Ciente de Energia (CE): Nesta solução, a extensão descrita na Seção 5.5.1 é utilizada pelo arcabouço no gerenciamento de capacidade. Em particular, o lucro do provedor é calculado pela Equação 5.6. Além disso, a capacidade total alocada satisfaz a restrição sobre a capacidade total disponível descrita pela Equação 5.7,

respeitando restrições de energia.

Tempo de Resposta Adaptativo (CE-R): Durante períodos de restrições de energia, e conseqüente redução de capacidade, o cliente pode concordar em relaxar seu requisito de tempo de resposta para aliviar uma possível redução na taxa de processamento de sua classe de aplicação devido à falta de capacidade. Nesse cenário, o requisito relaxado de tempo de resposta de uma classe i utilizando o SLA adaptativo quando a infra-estrutura passa por restrições de energia é definido como $R_i^{SLA,e}$. Para cada classe i fazemos $R_i^{SLA,e} = R_i^{SLA}(1 + w_i^e S)$, onde S é a fração da capacidade da infra-estrutura que precisa ser desligada e w_i^e é um fator escolhido pelo cliente que quantifica o quanto ele aceita relaxar seu requisito de tempo de resposta. O valor de w_i^e depende da aplicação e da duração do atraso que é aceitável para suas requisições. Dado S, a capacidade total da infra-estrutura disponível para atender requisições durante períodos de restrição de energia, utilizada na Equação 5.7, é dada por $C(1 - S)$.

Este trabalho não considera um esquema de custos adaptativos durante períodos de restrições de energia para evitar injustiças. No caso de um ataque de segurança, o cliente pode optar por pagar mais para reduzir o impacto negativo do ataque em sua aplicação. Porém, isto não incorre em degradação de serviço para as outras aplicações. Logo, uma solução baseada em custos adicionais pode apenas beneficiar o cliente vitimado, sem impactar os demais. Em contrapartida, num cenário de restrição de energia, um cliente que pague a mais certamente receberá uma fração maior dos recursos, ao custo de uma degradação do serviço recebido pelas outras classes. Apesar de vantajoso para o provedor de infra-estrutura por resultar em lucros maiores, essa solução não seria justa com os donos das outras classes de aplicação.

Os contratos adaptativos projetados para ataques de segurança (na Seção 5.4) e energia (nesta seção) são uma decisão tomada apenas entre o cliente e o provedor. A avaliação do impacto da utilização desses contratos SLA adaptativos na satisfação dos usuários finais da aplicação – que pode ser afetada, por exemplo, devido a um aumento do tempo de resposta – é deixada como trabalho futuro. Além dessa avaliação, novos modelos de negócio que considerem explicitamente o usuário final podem ser propostos para modelar com maior realismo e flexibilidade o relacionamento entre provedor, clientes e usuários.

Este capítulo apresentou um arcabouço de gerenciamento de capacidade para infra-estruturas multicamadas. Foram apresentados um modelo de desempenho preciso e um complexo modelo de otimização. Indo além dos objetivos tradicionais de melhorar o desempenho, foram modelados ataques de segurança, custos e restrições de energia. Os próximos capítulos farão uma avaliação do novo arcabouço.

Parâmetros da Infra-Estrutura	
Símbolo	Descrição
N	Número total de classes de aplicação hospedadas ($i \in [1, N]$).
K	Número total de camadas da infra-estrutura ($j \in [1, K]$).
Variáveis de Decisão do Modelo de Otimização	
Símbolo	Descrição
$f_{i,j}$	Fração da capacidade da camada j alocada à classe i.
λ_i^{ac}	Taxa de chegadas da classe i aceitas para processamento ($\lambda_i^{ac} \leq \lambda_i$).
Parâmetros do Sistema	
Símbolo	Descrição
λ_i	Taxa de chegadas da classe i prevista para o próximo intervalo.
$p_{i,j}$	Probabilidade de uma requisição da classe i completar seu serviço e deixar a infra-estrutura após visitar a camada j ($p_{i,K} = 1$).
$\lambda_{i,j}^{e}$	Taxa efetiva de chegada de requisições da classe i na camada j.
$d_{i,j}^{*}$	Demanda média de uma requisição da classe i na camada j executando em sua capacidade total.
$d_{i,j}$	Demanda média de uma requisição da classe i na camada j executando na infra-estrutura compartilhada ($d_{i,j} = d_{i,j}^{*} / f_{i,j}$).
$\rho_{i,j}$	Utilização da máquina virtual da classe i na camada j.
$\nu_{i,j}$	Máxima utilização planejada da máquina virtual da classe i na camada j ($\rho_{i,j} \leq \nu_{i,j}$).
Parâmetros do Contrato SLA	
Símbolo	Descrição
X_i^N	Taxa de processamento da classe i a ser atendida pelo provedor no modo de operação normal.
X_i^S	Taxa de processamento limite da classe i para o modo de operação sobrecarregado ($X_i^N \leq X_i^S$).
R_i^{SLA}	Requisito de tempo de resposta da classe i.
α_i	Tolerância da classe i a violações do tempo de resposta.
c_i	Valor reembolsado por unidade de taxa de processamento da classe i abaixo do requisito do modo de operação normal (X_i^N).
r_i	Valor recebido por unidade de taxa de processamento da classe i acima de X_i^N no modo de operação sobrecarregado.

Tabela 5.1: Sumário dos Parâmetros do Arcabouço

Parâmetros de Segurança	
Símbolo	Descrição
λ_i^+	Taxa de chegada de requisições legítimas para a classe i.
λ_i^-	Taxa de chegada de requisições ilegítimas para a classe i.
u_i^-	Constante de inflacionamento do custo de uma requisição no contrato SLA de custo adaptativo para a classe i (CA-C).
w_i^-	Constante de inflacionamento do requisito do tempo de resposta no contrato SLA adaptativo para a classe i (CA-R).
Parâmetros de Custos e Restrições de Energia	
Símbolo	Descrição
e_j	Custo de energia da operação da capacidade total da camada j.
C	Capacidade total da infra-estrutura.
s_j	Constante de normalização da capacidade da camada j em relação a C ($\sum_{j=1}^{K} s_j = C$).
S	Fração da infra-estrutura que precisa ser desligada devido a restrições de energia.
w_i^e	Constante de inflacionamento do requisito do tempo de resposta no contrato SLA adaptativo para a classe i (CE-R).

Tabela 5.2: Sumário dos Parâmetros de Segurança e Energia

Variáveis do Lucro do Provedor	
Símbolo	Descrição
g_i	Balanço das recompensas e reembolsos para a classe i.
g_i^s	Balanço das recompensas e reembolsos para a classe i durante ataques de segurança.
g_i^e	Balanço das recompensas e reembolsos para a classe i considerando custos de energia.

Tabela 5.3: Sumário das Variáveis de Lucro do Provedor

6 *Avaliação do Arcabouço de Gerenciamento de Capacidade em Infra-Estruturas Multicamadas*

Nesta seção é avaliado o novo arcabouço de gerenciamento de capacidade. O principal objetivo é quantificar seus benefícios e entender os principais compromissos relevantes ao gerenciamento de infra-estruturas multicamadas. Para tanto, ele é comparado com o arcabouço prévio baseado num modelo de recurso único [3, 4] e com alocação de capacidade estática em vários cenários.

As seguintes estratégias são consideradas na avaliação:

Hipoexponencial: Nesta variante do novo arcabouço é utilizada a distribuição hipoexponencial (Equação 5.3) para estimar a probabilidade de violação do requisito do tempo de resposta do SLA (restrição (h) na Figura 5.2).

Chebyshev: Nesta variante do novo arcabouço, a Desigualdade de Chebyshev (Equação 5.4) é utilizada para estimar a probabilidade de violação do requisito do tempo de resposta do SLA.

Camada única: No arcabouço de camada única [3,4], a probabilidade de violação do requisito é estimada a partir da distribuição exponencial do tempo de resposta de uma fila M/M/1 (Equação 4.3). O tempo médio de serviço de cada classe i é igual à soma dos tempos médios de serviço em cada camada ($\sum_{j=1}^{K} d_{i,j}^{*}$).

Estático: Para alocação estática de capacidade, é escolhida a melhor alocação em cada camada para a carga considerada. Uma fração fixa da capacidade total da camada j é alocada para a classe i. Esta fração é proporcional a sua utilização média ao longo de toda a carga de trabalho pela classe i. A distribuição hipoexponencial (Equação 5.3) é usada para estimar a taxa máxima de requisições

que podem ser admitidas para processamento na infra-estrutura, λ_i^{ac}, sem violar o requisito do SLA sobre o tempo de resposta. Assim, como a estratégia hipoexponencial, a estática é baseada numa representação mais precisa da infra-estrutura. Entretanto, a alocação definida no início do experimento é mantida fixa durante toda sua execução.

As três primeiras estratégias são autônomas. A diferença principal entre elas está no modelo de desempenho que distingue entre uma ou múltiplas camadas e diferentes aproximações.

A métrica principal de comparação utilizada neste capítulo é o lucro do provedor obtido com cada estratégia, dado pela Equação 4.2. Nos experimentos, é considerada uma infra-estrutura com duas camadas ($K = 2$), aplicável a uma plataforma com um servidor fronteiriço (como servidores HTTP e de aplicação) e recursos de apoio (como um banco de dados ou rede de armazenamento).

Foi construído um simulador dirigido por eventos que modela o sistema como seqüências de filas, cada seqüência com K centros. Ele recebe como entrada um registro de carga (*logs*) para N classes de aplicação. Para as estratégias autônomas, o simulador é acoplado a um solucionador do modelo de otimização que é chamado ao final de cada intervalo de controle, como ilustrado na Figura 4.1. O solucionador calcula o vetor de alocação de capacidade, $f_{i,j}$, e a taxa de chegada de requisições admitida, λ_i^{ac}, para o próximo intervalo.

O simulador implementa um controle de admissão justo que admite cada requisição da classe i com probabilidade λ_i^{ac}/λ_i. Assim, a premissa de que chegadas seguem um processo de Poisson continua válida para as requisições admitidas no sistema. Esta é uma abordagem conservadora se comparada a outros mecanismos que objetivam minimizar a variância do tempo entre chegadas [54].

Como o foco é na eficiência das novas soluções, as estratégias são comparadas em um cenário otimista para compreender os principais compromissos: é suposto que não há limitação no tempo para adaptar o sistema e que um método de previsão de carga perfeito é utilizado, onde a taxa de chegada do próximo intervalo é conhecida *a priori*. A seleção e avaliação de métodos de previsão de carga práticos, dentre as várias técnicas de séries temporais com diferentes graus de precisão disponíveis [2], são deixadas como trabalho futuro.

Além disso, é suposto que cada requisição admitida visita as duas camadas, e que

a utilização máxima planejada para todas as máquinas virtuais é 95% (ou seja, $p_{i,1} = 0$ e $\nu_{i,j} = 0,95$ para $i \in [1,N]$ e $j \in [1,K]$).

Durante cada intervalo, o simulador coleta o tempo de resposta de cada requisição, a taxa de processamento de cada classe e a utilização de cada máquina virtual. Essas informações são usadas para calcular o lucro do provedor. O simulador foi validado comparando seu tempo de resposta com o calculado pelo modelo analítico (Equação 5.3), com erros abaixo de 1%. Todos os resultados mostrados são médias de 5 execuções (20 na Seção 6.1), com desvio padrão menor que 2% das médias.

Na Seção 6.1 é discutida a escalabilidade do arcabouço de gerenciamento de capacidade proposto. Resultados de simulação para cargas sintéticas e para cargas mais realistas são mostrados nas Seções 6.2 e 6.3, respectivamente. A Seção 6.4 discute um cenário onde ambas cargas de trabalho e capacidade disponível na infra-estrutura são variadas dinamicamente. Finalmente, a Seção 6.5 apresenta um estudo da sensibilidade do arcabouço à premissa de tempos de serviço exponenciais.

Este capítulo não considera ataques de segurança, custos ou restrições de energia. Especificamente, $\lambda_i^- = e_j = S = 0$, para $i \in [1,N]$ e $j \in [1,K]$. A avaliação do impacto desses fatores é apresentada no Capítulo 7.

6.1 Desempenho e Escalabilidade

O modelo de otimização foi implementado e testado em AMPL [35], uma linguagem de modelagem para programação matemática. Vários diferentes solucionadores (*solvers*) [40,93,101] foram utilizados, e todos eles convergiram para a mesma solução para todas as entradas utilizadas.

A escalabilidade do arcabouço multicamadas foi avaliada para configurações com até 60 classes. Um número maior não foi usado devido a uma limitação da quantidade de variáveis de decisão existente na versão de estudante do AMPL, usada neste trabalho devido ao elevado custo das licenças para versões comerciais.

É avaliada apenas a estratégia hipoexponencial devido à sua maior complexidade e maiores tempos de solução. As outras estratégias consideradas são mais simples e requerem menor esforço computacional. Os experimentos são conduzidos usando o solucionador não-linear SNOPT [40], em um computador com um processador AMD Sempron 2400 de 2 GHz e 512 MB de RAM.

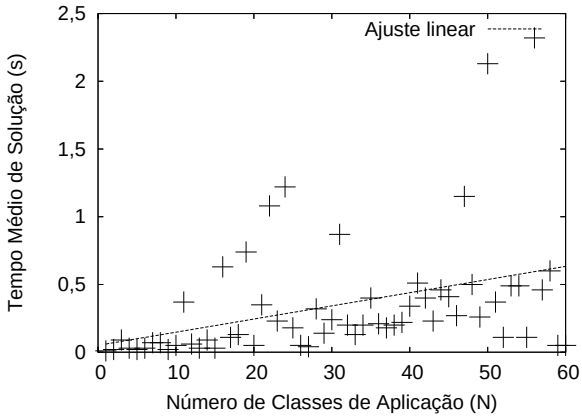


Figura 6.1: Escalabilidade do Arcabouço de Gerenciamento de Capacidade

A Figura 6.1 mostra o tempo médio de solução quando o número de classes de aplicação aumenta. As variações no tempo de solução para diferentes valores de N devem-se principalmente à quantidade de iterações necessárias ao solucionador para encontrar a solução ótima. O ajuste linear dos dados indica que o tempo médio de solução, tipicamente abaixo de 1 segundo, aumenta com um fator pequeno do número de classes. Argumenta-se então que o arcabouço de gerenciamento proposto escala bem para cenários práticos e pode ser usado para gerenciamento de capacidade em tempo real.

6.2 Avaliação com Cargas Sintéticas

Esta seção apresenta resultados de simulações para cargas sintéticas com duas classes de aplicação. São utilizados dois cenários para ilustrar os principais compromissos e benefícios do novo arcabouço. O cenário 1 (Seção 6.2.1) considera uma carga de trabalho leve, enquanto o cenário 2 (Seção 6.2.2) considera uma carga pesada.

6.2.1 Cenário 1 – Carga Leve

Neste cenário, requisições para cada classe chegam de acordo com um processo de Poisson não homogêneo em degraus, como mostrado na Figura 6.2. Taxas de chegada variam de 0 a 1000, com degraus e períodos com duração de 1000 e 10000 segundos, respectivamente. Ambas as cargas possuem perfis idênticos com um deslocamento nos períodos. Este é um cenário interessante para abordagens autônomas, que são capazes de realocar capacidade ociosa em máquinas virtuais com carga baixa

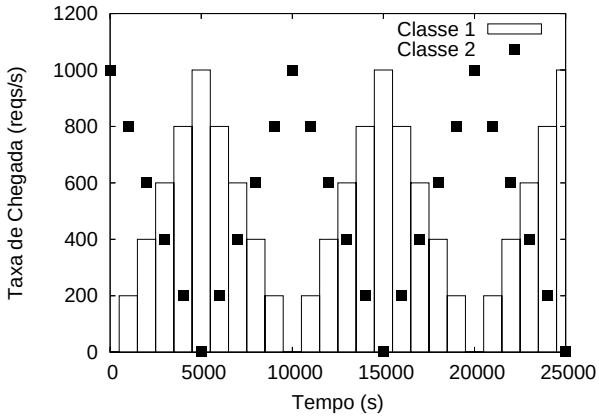


Figura 6.2: Carga Sintética

Classe i	Cenário 1		Cenário 2	
	$d^*_{i,1}$ (ms)	$d^*_{i,2}$ (ms)	$d^*_{i,1}$ (ms)	$d^*_{i,2}$ (ms)
1	0,6	0,4	1	0,7
2	0,4	0,6	0,7	1

Tabela 6.1: Tempo Médio de Serviço por Camadas nos Cenários 1 e 2

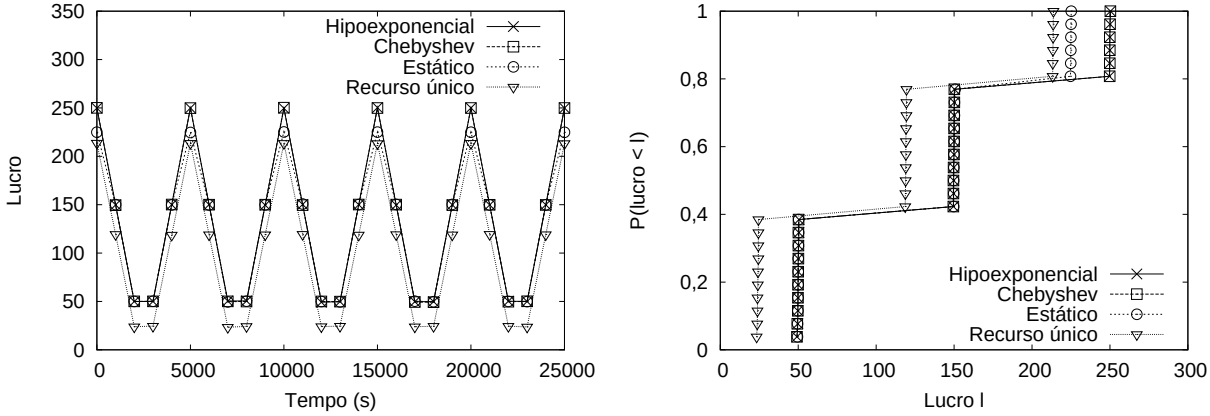
para máquinas virtuais sobrecarregadas, aumentando a taxa de processamento e atendendo requisitos do SLA. O intervalo de controle é configurado como 1000 segundos. Assim, o gerenciador de capacidade autônomo é chamado nos instantes que ocorrem as mudanças de taxa de chegada, que são conhecidas *a priori*.

Os tempos médios de serviço para cada classe em cada camada são mostrados na Tabela 6.1. Os valores de demanda foram escolhidos de forma que a infra-estrutura esteja levemente superdimensionada para atender a carga de trabalho agregada nas estratégias multicamadas. Note que as classes 1 e 2 têm maior demanda por recursos nas camadas 1 e 2, respectivamente. Nesse caso, a abordagem autônoma multicamadas é capaz de alocar dinamicamente, para cada classe, mais recursos na camada onde ela necessita mais. No cenário 1 o desbalanceamento do tempo de serviço não é muito significativo. Por último, os valores dos parâmetros de negócio são mostrados na Tabela 6.2. Ambas as classes têm parâmetros de negócio iguais, pois o interesse primário é a avaliação da eficiência das estratégias analisadas.

A Figura 6.3-a mostra o lucro total obtido pelo provedor para cada estratégia ao longo da simulação. O padrão repetido da curva é resultante do comportamento periódico das cargas. Os picos correspondem aos períodos onde as classes de aplicação têm taxas de chegada complementares. Nesses instantes, o provedor pode dedicar

Cenário	R_i^{SLA}	X_i^N	X_i^S	c_i	r_i	α_i
1	0,1 s	500 req/s	1200 req/s	1,0	0,5	0,1
2	0,1 s	500 req/s	1200 req/s	1,0	0,5	0,1

Tabela 6.2: Valores dos Parâmetros do Modelo de Negócio



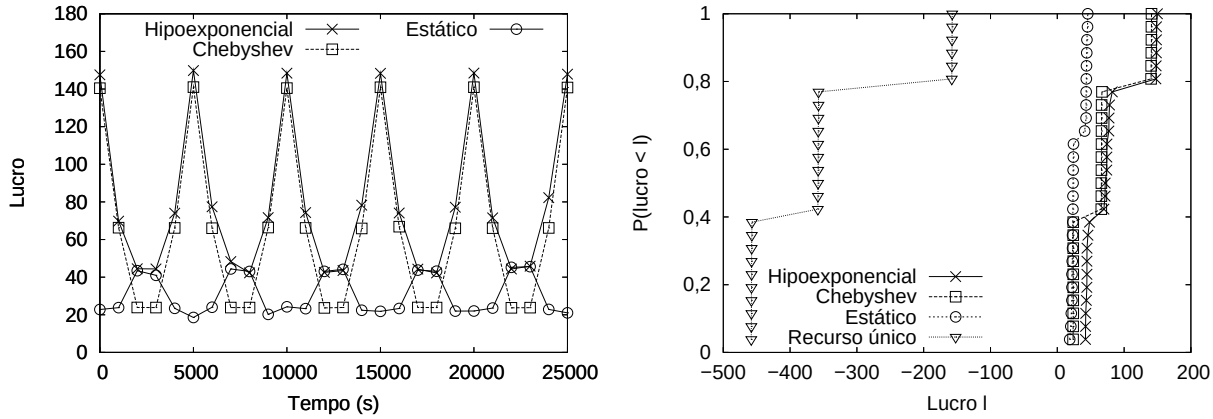
(a) Lucro ao Longo do Tempo (b) Distribuição Acumulada do Lucro

Figura 6.3: Lucro do Provedor no Cenário 1

mais capacidade à classe com maior taxa de chegadas e capitalizar sobre sua operação em modo sobrecarregado. Os vales correspondem aos períodos onde as classes têm taxas de chegadas similares, utilizando capacidade do provedor para satisfazer requisitos do modo de operação normal, o que não resulta em recompensas.

As abordagens hipoexponencial e de Chebyshev resultam em lucros similares ao longo da simulação. Interessante notar que ambas resultam apenas em ganhos marginais (11%) sobre a estratégia estática quando as classes têm cargas complementares. Esse resultado é a despeito deste ser o melhor cenário para estratégias autônomas, capazes de realocar capacidade da máquina virtual subutilizada para a sobrecarregada. Isto deve-se ao fato da carga imposta a cada camada ser muito leve ao longo da simulação. Assim, a habilidade de adaptar-se não desempenha um papel central, e quase todas as requisições são admitidas no sistema por todas as estratégias. O lucro é ditado principalmente pela capacidade do provedor em capitalizar sobre aplicações executando em modo sobrecarregado, que é igual para todas as estratégias.

Por outro lado, na estratégia de recurso único [3,4], o tempo médio de serviço para cada classe é 1 milissegundo, fazendo a infra-estrutura levemente subdimensionada para suportar a carga de trabalho agregada. A utilização média do sistema na estraté-



(a) Lucro ao Longo do Tempo (b) Distribuição Acumulada do Lucro

Figura 6.4: Lucro do Provedor no Cenário 2

gia de camada única, considerando as duas classes, é dada pela multiplicação da taxa de chegada agregada (1000 reqs/s) pela demanda média no recurso único (1 ms), que é 1. O ganho no lucro obtido pelas novas estratégias multicamadas sobre a de recurso único varia de 17% (nos picos) a 103% (nos vales). De fato, quando as classes têm cargas similares, a estratégia de recurso único, baseada num modelo simplificado do sistema, é significativamente superada até mesmo pela estratégia estática. A Figura 6.3-b sumariza esses resultados, mostrando as distribuições acumuladas do lucro do provedor para todos os intervalos. As estratégias autônomas multicamadas resultam num ganho médio de lucro de 28% sobre a estratégia de recurso único.

6.2.2 Cenário 2 – Carga Pesada

O cenário 2 é caracterizado por cargas mais pesadas e desbalanceadas. Os valores das demandas são mostrados na Tabela 6.1 e foram escolhidos de forma a serem mais desbalanceados entre camadas e fazer a infra-estrutura um pouco subdimensionada mesmo para os novos métodos multicamadas. O perfil da carga é idêntico ao mostrado na Figura 6.2, porém as taxas de chegadas variam de 200 a 1200 requisições por segundo. Como o foco é na relação custo-benefício dos métodos, os parâmetros de negócio são mantidos iguais aos do cenário 1, mostrados na Tabela 6.2. A duração do intervalo de controle é mantida em 1000 segundos, mas é variada no final da seção para verificar a sensibilidade do arcabouço ao seu valor.

As Figuras 6.4-a e 6.4-b mostram o lucro do provedor obtido por cada estratégia. A estratégia hipoexponencial resulta em maior lucro que a abordagem baseada na

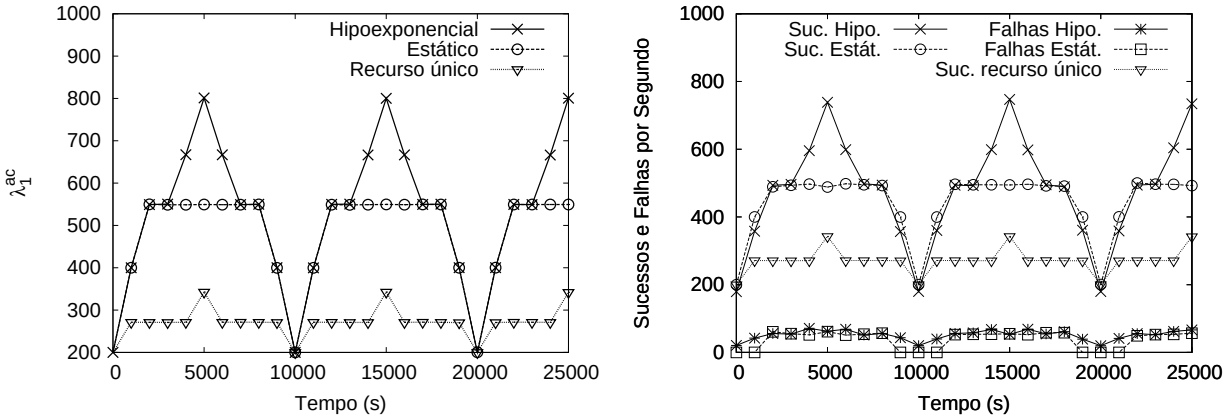
Desigualdade de Chebyshev, quando a taxa de chegadas das classes estão balanceadas. Nesses instantes, a taxa de chegadas das duas aplicações é maior que X_i^N, o que torna o fato da infra-estrutura estar subdimensionada mais crítico. O ganho médio, ao longo de toda a simulação, do lucro da estratégia hipoexponencial sobre a de Chebyshev é 20%. Como esperado, o erro resultante da utilização da Desigualdade de Chebyshev, uma aproximação, fica mais significativo para cargas mais pesadas.

Ao contrário do cenário anterior, ambas estratégias autônomas significativamente superam a estratégia estática (em até 580%) nos intervalos onde as classes têm taxas de chegada complementares. Além disso, a estratégia de recurso único, com lucros flutuando entre -458 e -157 (omitidos na Figura 6.4-a), é superada pelas outras três estratégias por ordens de magnitude.

Em suma, os compromissos ilustrados acima podem ser compreendidos a partir da identificação de dois fatores primários que impactam a eficiência das estratégias de gerenciamento de capacidade: a habilidade de adaptar a mudanças da carga de trabalho e a precisão do modelo de desempenho. Alocação de capacidade fixa penaliza significativamente a estratégia estática para cargas pesadas e heterogêneas (e.x.: cenário 2). O modelo de desempenho impacta em primeira ordem as decisões sobre alocação de capacidade e controle de admissão.

Em ambos cenários analisados (e num cenário omitido onde as aplicações têm cargas balanceadas e homogêneas), a estratégia de recurso único é significativamente penalizada pelo seu modelo de desempenho mais simplista, e conseqüentemente mais impreciso. Por outro lado, as estratégias hipoexponencial e estática utilizam a distribuição hipoexponencial do tempo de resposta de uma seqüência de filas M/M/1. Pode ser facilmente mostrado, para tempos médios de serviço fixos, que a média do tempo de resposta calculado na primeira abordagem (exponencial) é sempre maior que o da segunda (hipoexponencial) [98]. Assim, para satisfazer a restrição sobre o tempo de resposta, dada por uma função exponencial que cresce com a utilização dos recursos, a abordagem de recurso único é forçada a fazer decisões de alocação e controle de admissão mais conservadoras, resultando por fim em lucros significativamente menores, até mesmo se comparados à abordagem estática.

Essa conclusão é ilustrada nas Figuras 6.5-a e 6.5-b, que mostram as taxas de requisições admitidas, taxas de processamento válida (sucessos) e violações do requisito de tempo de resposta do SLA por segundo (falhas) para a classe de aplicação 1. Curvas de λ_i^{ac} e sucessos para a abordagem de Chebyshev, omitidas, estão entre as das



(a) Taxa de Chegada Admitida ao Longo do Tempo (b) Sucessos e Falhas (violações) do SLA

Figura 6.5: Detalhe do Desempenho da Infra-Estrutura para o Cenário 2

estratégias hipoexponencial e estática. A Figura 6.6 mostra a distribuição acumulada do tempo de resposta das requisições durante toda a simulação.

As decisões de alocação de capacidade e controle de admissão feitas pelas estratégias hipoexponencial e estática, obtidas com o modelo hipoexponencial mais preciso e agressivo, resultam em um maior número de falhas do SLA. Porém, como mostrado na Figura 6.6, a restrição sobre o tempo de resposta do SLA ($P[R_i \geq 0,1] \leq 0,1$) ainda é satisfeita. O novo modelo admite a maior quantidade de requisições para processamento que não resulta em violação do SLA. Por outro lado, devido ao modelo de desempenho menos preciso, a abordagem baseada na Desigualdade de Chebyshev, mesmo admitindo um número maior de requisições que a abordagem estática, tem uma distribuição de tempo de resposta mais à esquerda. Modelos menos precisos admitem menos requisições para processamento, o que resulta em filas menores em cada camada, diminuindo o tempo de resposta. Conseqüentemente, a quantidade de falhas da abordagem de Chebyshev e de recurso único é quase nula, como pode ser visto na Figura 6.6.

Por último, a fim de verificar o impacto de erros na predição da carga de trabalho do próximo intervalo, foram executadas simulações usando diferentes durações para o intervalo de controle no cenário 2. São escolhidas durações para o intervalo de controle que não coincidem com os instantes onde a carga muda. Para duração dos intervalos de controle de 300 e 600 segundos, a redução no lucro médio obtido, comparado com os resultados da Figura 6.4-a, é apenas 5% (8%) e 11% (15%), respectivamente, para a estratégia hipoexponencial (Chebyshev). Logo, a nova solução é

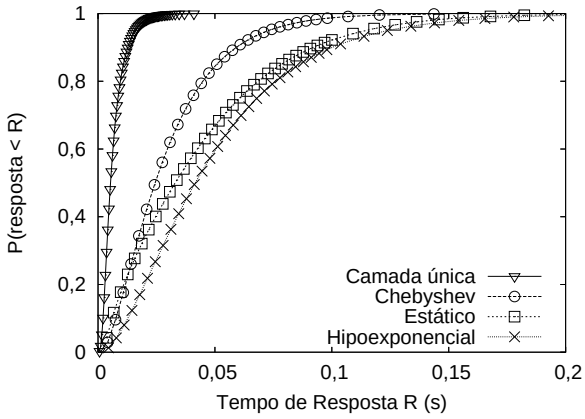


Figura 6.6: Distribuição Acumulada do Tempo de Resposta no Cenário 2

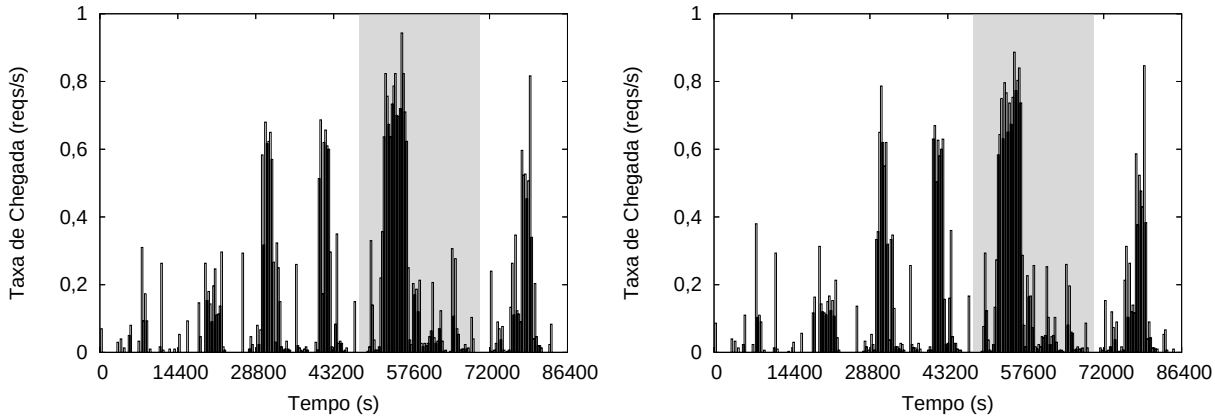
razoavelmente robusta a erros de predição da carga de trabalho.

6.3 Avaliação com Cargas Realistas

Nesta seção, as diferentes estratégias de gerenciamento de capacidade são avaliadas para perfis de carga mais realistas. Novas cargas de trabalho são construídas utilizando registros de acesso contendo o número de requisições recebidas, a cada intervalo de 5 minutos, para 4 aplicações reais de comércio eletrônico. Os registros têm uma duração de três meses correspondentes ao período de 23/11/2004 a 23/2/2005. Acordos de confidencialidade impedem a divulgação da fonte dos dados.

Todos os quatro registros possuem perfis de carga similares, com picos aproximadamente nos mesmos instantes, diminuindo as oportunidades para realocação de recursos de aplicações com carga leve para as sobrecarregadas. Taxas de chegadas variam muito, com pico de 17 requisições por segundo e uma média de 0,078 requisições por segundo. Existem vários intervalos de 5 minutos onde as aplicações não recebem requisições. Cargas realistas são construídas supondo que as chegadas de requisições seguem processos Poisson não-homogêneos, com taxas podendo mudar a cada intervalo de 5 minutos, conforme os registros. As Figuras 6.7-a e 6.7-b mostram a variação da taxa de chegada para duas das aplicações num dia típico.

Dois novos cenários de configuração são considerados na análise. No cenário 3, são executadas simulações com as 4 classes de aplicação construídas a partir dos registros. O cenário 4, cuja carga é construída duplicando cada um dos registros originais e deslocando as requisições em cada réplica 6 horas para o futuro, tem uma



(a) Aplicação 1 (b) Aplicação 3

Figura 6.7: Registo de Acesso para Duas Aplicações de Comércio Eletrônico

Cenário / Média dos Tempos de Serviço (s)		Classe i							
		1	2	3	4	5	6	7	8
3	$d^*_{i,1}$	0,5	2,0	1,5	3,0	–	–	–	–
	$d^*_{i,2}$	3,0	1,5	2,0	0,5	–	–	–	–
4	$d^*_{i,1}$	0,25	1,25	0,75	0,9	0,85	1,0	0,5	1,5
	$d^*_{i,2}$	1,5	0,5	1,0	0,85	0,9	0,75	1,25	0,25

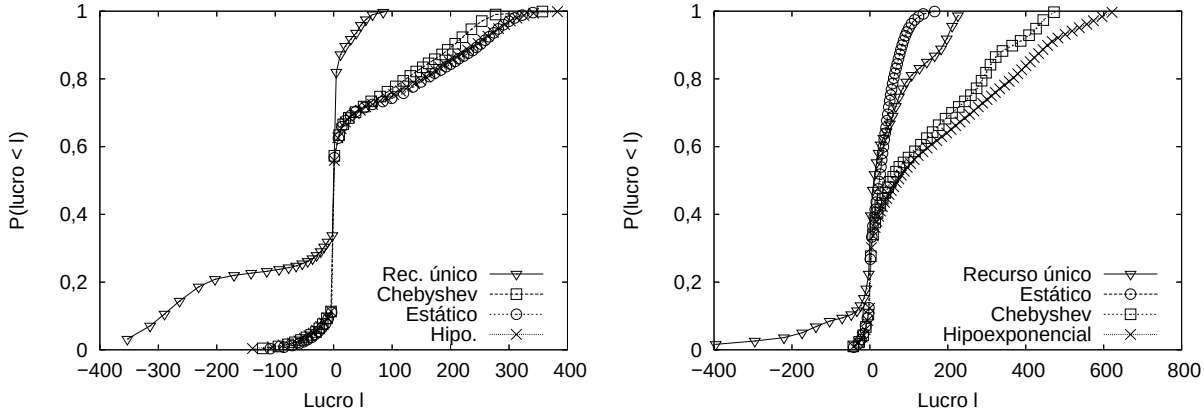
Tabela 6.3: Tempo Médio de Serviço Por Camadas nos Cenários 3 e 4

carga com 8 aplicações. A Tabela 6.3 mostra o tempo médio de serviço das classes em cada camada, escolhidos de forma a fazê-las subdimensionadas para atender a carga agregada. Cada aplicação tem uma demanda total por requisição igual a 3,5 e 1,75 segundos nos cenários 3 e 4, respectivamente. Note que a quantidade de processamento total requisitada pelos cenários 3 e 4 à infra-estrutura é idêntica.

A Tabela 6.4 mostra os parâmetros do modelo de negócio. Requisitos de taxa de processamento do SLA são fixos em ambos cenários pois têm a mesma variação em suas cargas de trabalho. O modo de operação normal exige uma taxa de processamento, X_i^N, um pouco acima da média das taxas de chegada. O valor de X_i^S é consideravelmente maior para alojar os grandes picos existentes nas cargas. O requisito do tempo de resposta, R_i^{SLA}, é escolhido como 60 vezes o valor da demanda total de uma requisição. Os custos das requisições, c_i e r_i, no cenário 4 são a metade dos valores escolhidos para o cenário 3 de forma a normalizar o lucro máximo potencial do provedor; pois no cenário 4 são processadas o dobro de requisições, cada uma com metade da demanda do cenário 3. A duração do intervalo de controle é configurada

Cenário	R_i^{SLA}	X_i^N	X_i^S	c_i	r_i	α_i
3	210 s	0,08 req/s	10 req/s	3500	1750	0,1
4	105 s	0,08 req/s	10 req/s	1750	875	0,1

Tabela 6.4: Valores dos Parâmetros do Modelo de Negócio



(a) Cenário 3 (b) Cenário 4

Figura 6.8: Distribuição Acumulada dos Lucros para Cargas Realistas

como 5 minutos.

As distribuições acumuladas do lucro do provedor para os cenários 3 e 4 são mostradas nas Figuras 6.8-a e 6.8-b, respectivamente. A estratégia hipoexponencial leva aos maiores lucros em ambos os casos. No cenário 3, a estratégia estática é tão boa quanto a hipoexponencial, pois os perfis muito similares das quatro cargas deixam pouco espaço para ganhos através do gerenciamento *dinâmico* de capacidade. Note, porém, a degradação significativa da estratégia estática no cenário 4. Isso por que há mais oportunidades de ganhos através da alocação dinâmica de capacidade entre as 8 classes consideradas, devido ao deslocamento de 6 horas nos perfis de carga. Nesse cenário, a abordagem hipoexponencial resulta em um ganho médio no lucro diário de 429% sobre a abordagem estática.

Como no cenário 2, a abordagem hipoexponencial tem melhor relação custo-benefício que a abordagem de Chebyshev, resultando em ganhos médios no lucro diário de 20% e 26% nos cenários 3 e 4, respectivamente. Isso deve-se ao fato da infra-estrutura estar subdimensionada para atender todas as requisições, onde o modelo de desempenho mais preciso admite uma maior quantidade de requisições para processamento, resultando em maiores lucros. Mais uma vez, a estratégia de recurso

único é superada por ordens de grandeza.

Por fim, apesar dos parâmetros das Tabelas 6.3 e 6.4 terem sido escolhidos de forma que o lucro do provedor fosse similar nos dois cenários, a Figura 6.8 mostra que o lucro das estratégias autônomas é maior no cenário 4. Isso deve-se ao fato de que uma maior quantidade de requisições é servida através da realocação de capacidade entre as aplicações que têm perfis diferentes de carga.

6.4 Variando a Capacidade da Infra-Estrutura

Nesta seção, a aplicabilidade das estratégias autônomas é avaliada quando a capacidade total disponível em uma das camadas diminui subitamente. Esse pode ser o caso, por exemplo, de pane do *hardware* ou manutenção não programada na infra-estrutura. Durante esse período, a capacidade local disponível para servir requisições das aplicações hospedadas diminui. Outras camadas não afetadas ainda podem dedicar sua capacidade total às aplicações.

Simulações com a mesma carga e parâmetros usados no cenário 3 foram executadas. Porém, a simulação é limitada ao dia mostrado na Figura 6.7, e a capacidade da camada 1 é reduzida em 25% durante o período de 6 horas sombreado. Durante o período de capacidade reduzida, os tempos médios de serviço de cada classe i são inflacionados de acordo com a nova capacidade total, ou seja, $d^*_{i,1} = d^*_{i,1}/0.75$.

A Figura 6.9 mostra os lucros obtidos para as estratégias hipoexponencial e de recurso único ao longo do tempo. Não mostramos a distribuição acumulada para enfatizar a degradação no lucro durante o período de capacidade reduzida. Claramente, a abordagem hipoexponencial multicamadas é muito mais robusta, resultando em lucros significativamente maiores mesmo quando a capacidade da camada 1 está reduzida. Os lucros médios das estratégias hipoexponencial e de recurso único durante toda a simulação são 52 e -106, respectivamente. Durante o período de capacidade reduzida os lucros médios são -1 e -172, respectivamente. O comportamento da estratégia de Chebyshev é similar ao da hipoexponencial.

Este cenário foi planejado como uma análise inicial da operação do arcabouço de gerenciamento de capacidade em situações adversas. Essa análise é estendida no capítulo seguinte para ataques de segurança, custos e restrições de energia.

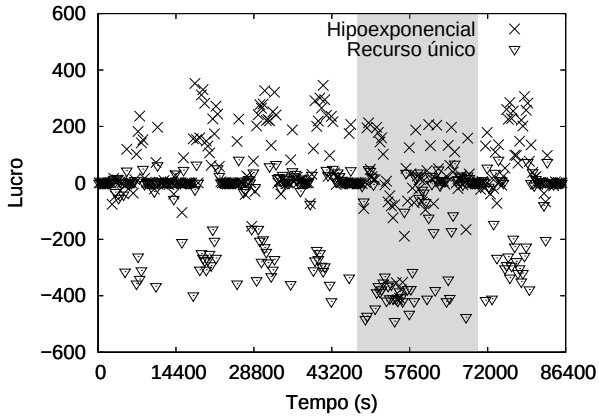


Figura 6.9: Lucro do Provedor ao Longo do Tempo com Redução de Capacidade

6.5 Sensibilidade do Arcabouço a Alta Variabilidade do Tempo de Serviço

Esta seção faz uma análise da sensibilidade do arcabouço a uma de suas premissas principais: tempos de resposta em cada camada exponencialmente distribuídos.

Foram executados experimentos para a configuração do cenário 2, carga sintética e pesada, assumindo que a distribuição dos tempos de serviço têm uma variabilidade maior, seguindo uma distribuição lognormal [98] com vários coeficientes de variação (CV). Para enfatizar o impacto do aumento da variância dos tempos de serviço, é considerado um cenário pessimista no qual o valor original de R_i^{SLA} no cenário 2 é dividido por 3 (ou seja, $R_i^{SLA} = 0{,}034$ segundos), diminuindo o limite para atrasos no tempo de serviço. Todos os outros parâmetros do cenário 2 são mantidos. Também são apresentados resultados para tempos de serviços exponencialmente distribuídos, que servem como base de comparação.

São consideradas as estratégias hipoexponencial e Chebyshev, baseadas em modelos de filas M/M/1. Também é considerada a estratégia abaixo:

Chebyshev G/G/1 Esta estimativa usa o modelo de desempenho preliminar apresentado no Apêndice B, baseado em filas G/G/1. A Desigualdade de Chebyshev é utilizada (Equação 5.4) para estimar $P[R_i \geq R_i^{SLA}]$, mas $E[R_i]$ e $VAR[R_i]$ são calculados a partir das métricas correspondentes dos tempos de residência em cada camada. Os tempos de residência em cada camada são estimados a partir das distribuições do tempo de serviço (lognormal) e do tempo de espera na fila. Métricas do tempo de espera são estimadas usando os primeiros três momentos

Distribuição do Tempo de Serviço	Valores Médios					
	Hypoexp.		Cheb. M/M/1		Cheb. G/G/1	
	Lucro	Prob.	Lucro	Prob.	Lucro	Prob.
Exponencial	869	10%	784	1.2%	784	1,2%
Lognormal (CV=2)	747	27%	783	8.5%	631	1,5%
Lognormal (CV=3)	634	38%	732	16%	573	1,6%
Lognormal (CV=5)	492	52%	642	26%	454	1,7%
Lognormal (CV=10)	341	67%	515	41%	187	1,3%

Tabela 6.5: Lucro Médio e Probabilidade de Violação do SLA para Tempos de Serviço com Alta Variabilidade

da distribuição do tempo de serviço, como detalhado no Apêndice B. Também foram executados experimentos com um modelo baseado em filas M/G/1, com resultados qualitativamente similares (por isso são omitidos).

Note que as duas estratégias baseadas em filas M/M/1 se apresentam como aproximações ao supor que o tempo de serviço em cada camada é exponencial, introduzindo erros nos resultados. Além disso, as saídas de requisições de uma máquina virtual, que serve de entrada para a próxima, só seguem um processo de Poisson se os tempos de serviço forem exponencias [54]. Quando os tempos de serviço seguem outra distribuição, as chegadas na segunda (e demais) camada não seguem um processo de Poisson. Este fato também não é capturado pelas estratégias baseadas em filas M/M/1.

A Tabela 6.5 mostra valores médios do lucro e da probabilidade de violação do requisito de tempo de resposta do SLA, ao longo de toda a simulação, para as várias combinações de modelo e distribuição do tempo de serviço. Ao contrário dos resultados apresentados nas seções anteriores, os valores de lucro apresentados aqui levam em consideração os pagamentos realizados pelos clientes como parte do SLA, evitando assim valores negativos de lucro, comuns nestes experimentos devido ao aumento da variância dos tempos de serviço.

As decisões de alocação muito agressivas realizadas pela estratégia hipoexponencial, que resultam em uma probabilidade de falhas próxima de α_i no cenário exponencial, deixam pouco espaço para acomodar a variância adicional introduzida pela distribuição lognormal de tempo de serviço. Com o aumento do CV, a probabilidade de violação cresce e o lucro diminui, rapidamente. A estimativa de Chebyshev utilizando filas M/M/1 é mais conservadora que a Hipoexponencial. Como ela admite uma quantidade menor de requisições para processamento, as requisições têm

tempos de resposta menores (Figura 6.6), portanto, dão mais espaço para erros decorrentes do aumento da variância. De fato, a estimativa de Chebyshev atende o SLA ($P[R_i \geq R_i^{SLA}] \leq \alpha_i$) para tempos de serviço com coeficientes de variação menores do que 3, com um pequeno impacto no lucro do provedor. Por último, a estimativa de Chebyshev utilizando filas G/G/1 sempre atende o SLA. Isso é possível pois o modelo mais preciso captura a maior variação dos tempos de serviço com distribuição lognormal e configura a infra-estrutura de acordo. Porém, essa configuração consiste de um controle de admissão muito conservador, devido às imprecisões das aproximações utilizadas e à grande variação dos tempos de serviço, que, por fim, resulta em lucros muito baixos.

Resultados similares foram obtidos com outras distribuições de tempo de serviço, como Normal e Weibull. Assim, no caso de tempos de serviço não-exponenciais, argumenta-se que o novo arcabouço, utilizando a simples Desigualdade de Chebyshev e assumindo filas M/M/1 obtém resultados satisfatórios para valores de coeficientes de variação menores que 3. O desenvolvimento de modelos para tempos de serviço com variação maior é deixado como trabalho futuro.

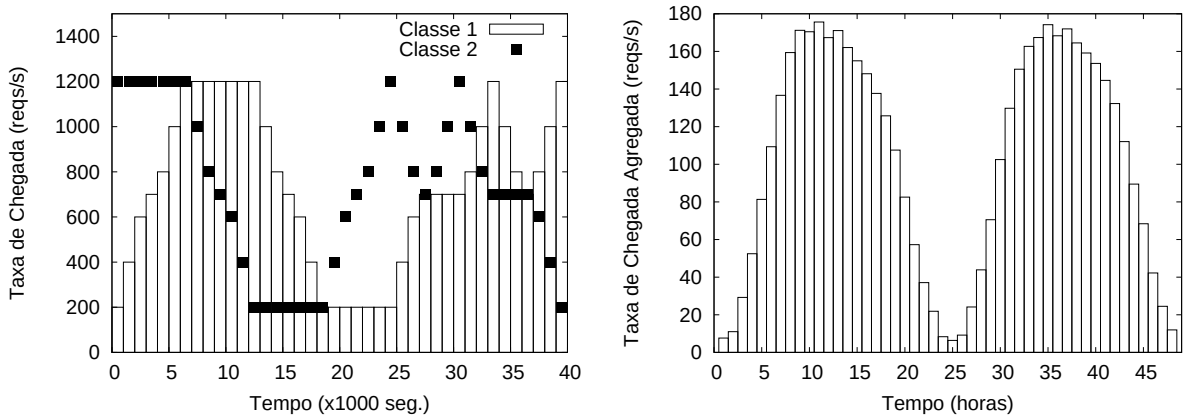
7 *Avaliação do Arcabouço de Gerenciamento de Capacidade Face Ataques de Segurança, Custos e Restrições de Energia*

Esta seção apresenta resultados de simulação para analisar e melhor entender os principais compromissos que surgem no gerenciamento de capacidade de infra-estruturas sob ataques de segurança (Seção 7.2) e restrições de energia (Seção 7.3). A análise leva em consideração interesses do provedor e dos clientes. O interesse do provedor é caracterizado como seu lucro, e os interesses dos clientes são expressos pela taxa de processamento válida, pela distribuição do tempo de resposta e pelo custo por requisição legítima servida. A carga de trabalho e parâmetros de configuração do sistema utilizados em nossa avaliação são descritos na Seção 7.1.

7.1 Carga de Trabalho e Parâmetros do Sistema

Como no Capítulo 6, são considerados dois cenários descrevendo a carga de trabalho e configuração do sistema. Ambos os cenários consistem de infra-estruturas com duas camadas ($K = 2$). Requisições sempre visitam as duas camadas, ou seja, $p_{i,1} = 0$. Novamente, supõe-se que classes de aplicação têm requisitos de SLA e do modelo de negócio iguais, dando ênfase no gerenciamento de capacidade. A utilização máxima planejada para uma máquina virtual é 95% ($\nu_{i,j} = 0,95$). A carga de trabalho é conhecida *a priori*, e o módulo de gerenciamento de capacidade é executado sempre que a taxa de requisições (legítimas ou ilegítimas) muda.

Neste capítulo, o lucro do provedor é calculado contabilizando o pagamento feito pelo cliente como parte do SLA, que é reembolsado pelo provedor no caso de violações do SLA. Dessa forma, o lucro do provedor é sempre positivo. Os lucros são



(a) Cenário 1 (b) Cenário 2: Carga Agregada

Figura 7.1: Cargas de Trabalho

apresentados assim para evitar valores negativos, muito mais comuns em cenários com ataques de segurança e restrições de energia. Todos os resultados apresentados são médias de 5 execuções com desvio padrão menor que 2% das médias.

O cenário 1 é composto de uma carga de trabalho sintética para duas classes de aplicações, com requisições chegando de acordo com os processos de Poisson não-homogêneos em degraus mostrados na Figura 7.1-a. Essa nova carga sintética nos permite avaliar o comportamento do sistema em um número maior de situações, por exemplo períodos onde uma classe tem taxa de chegada fixa e a outra varia (aumenta ou diminui), capturando compromissos relevantes para ataques de segurança que não são capturados com a carga sintética utilizada no Capítulo 6. Cada degrau tem uma duração de 1000 segundos. Note que as duas classes têm a mesma taxa média de chegada de requisições ao longo da carga, 732,5 requisições por segundo.

Os tempos médios de serviço são mostrados na Tabela 7.1. Eles são escolhidos de forma a tornar a infra-estrutura subdimensionada para atender a carga agregada durante a maior parte do tempo. Existe um curto intervalo ao redor do segundo 19000 quando a infra-estrutura tem capacidade suficiente para atender toda a carga. A Tabela 7.2 mostra os requisitos do SLA para o tempo de resposta e taxa de processamento. O valor de R_i^{SLA} é escolhido como 20 vezes a demanda total de uma requisição, considerando todas as camadas, $\sum_{j=1}^{K} d_{i,j}^{*}$.

O cenário 2 é construído com registros de acesso para um portal Web real. Esse registro contém o número de chegadas de requisições por hora durante um período de seis meses entre 1/1/2006 e 30/06/2006. Se comparada à carga real utilizada

Cenário	Média dos Tempos de Serviço (ms)	Classe i					
		1	2	3	4	5	6
1	$d^*_{i,1}$	0,9	0,6	–	–	–	–
	$d^*_{i,2}$	0,6	0,9	–	–	–	–
2	$d^*_{i,1}$	12	8	12	8	12	8
	$d^*_{i,2}$	8	12	8	12	8	12

Tabela 7.1: Tempo Médio de Serviço Por Camadas

Cenário	R_i^{SLA}	X_i^N	X_i^S	c_i	r_i	α_i
1	0,03 s	500 req/s	1000 req/s	1	0,5	0,1
2	0,4 s	18,4 req/s	36,8 req/s	1	0,5	0,1

Tabela 7.2: Valores dos Parâmetros do Modelo de Negócio

no Capítulo 6, esta carga é preferida por ser mais recente e ter um número significativamente maior de acessos. Cargas realistas são construídas dividindo o registro em 6 sub-registros, um por mês, e tomando os acessos de cada sub-registro como as chegadas de uma classe de aplicação diferente. É suposto que a taxa de chegada de cada classe segue um processo de Poisson não-homogêneo com taxa dada pelo sub-registros respectivo.

Apesar da carga ter duração de um mês, são mostrados resultados apenas para os dois dias exibidos na Figura 7.1-b, objetivando facilitar a exposição dos resultados. A carga apresenta forte padrão diurno de acesso. A média, o mínimo e o máximo das taxas de chegada para cada classe são mostrados na Tabela 7.3, onde λ_i^+ representa a taxa de chegada de requisições legítimas.

As demandas médias, mostradas na Tabela 7.1, são escolhidas de forma a fazer a infra-estrutura subdimensionada para atender a carga média agregada. Os parâmetros do modelo de negócio são mostrados na Tabela 7.2. Novamente, o valor de R_i^{SLA} é escolhido como 20 vezes a demanda total em todas as camada de uma requisição.

λ_i^+ (reqs/s)	Classe i					
	1	2	3	4	5	6
Mínimo	0,1	0,6	0,5	0,3	0,6	0,7
Médio	24	13	18	20	13	18
Máximo	42	27	41	39	25	34

Tabela 7.3: Características da Carga de Trabalho Realista

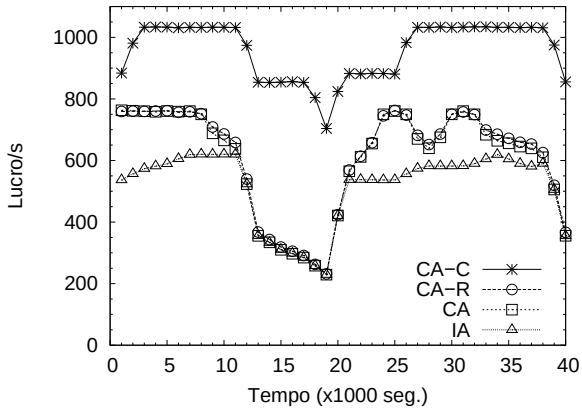


Figura 7.2: Lucro do Provedor no Cenário 1

O valor de X_i^N é escolhido um pouco acima da taxa média de chegada de requisições. X_i^S tem um valor duas vezes maior.

7.2 Ataques de Segurança

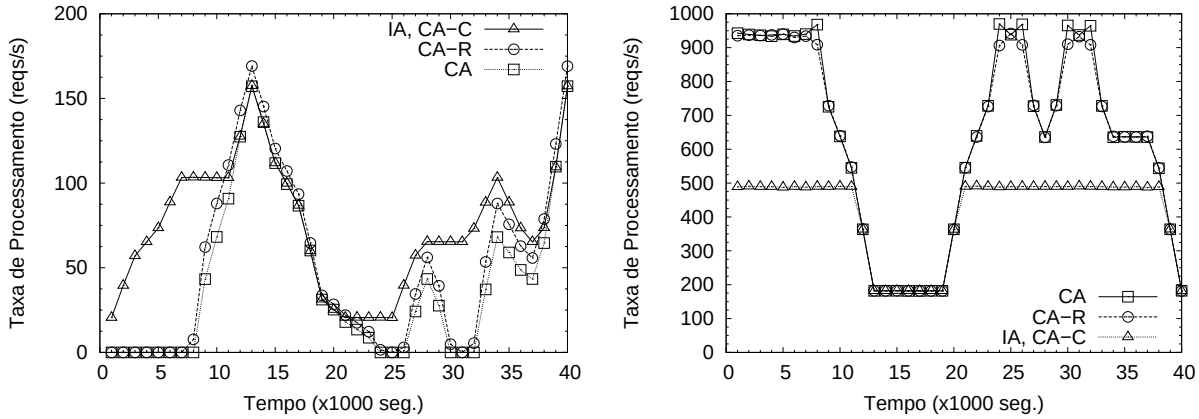
Esta seção apresenta os resultados e compromissos relativos a ataques de segurança. Experimentos com a carga sintética e real são discutidos nas Seções 7.2.1 e 7.2.2, respectivamente. Esta seção não considera custos de energia, fazendo $e_1 = e_2 = 0$.

7.2.1 Cenário 1 – Carga Sintética

Primeiro são discutidos os resultados mais relevantes para o cenário 1, acrescido de um ataque para a classe 1 com uma taxa de 5000 requisições por segundo ($\lambda_1^- = 5000$) durante todo o intervalo de simulação, isto é, o peso do ataque à classe 1, λ_1^- / λ_1, varia de 0,81 a 0,96. A classe 2 não está sob ataque.

A Figura 7.2 mostra o lucro do provedor para as quatro estratégias consideradas. As Figuras 7.3-a e 7.3-b mostram a taxa de processamento válida das classes 1 e 2, respectivamente, com diferentes limites no eixo y devido à redução na taxa válida de processamento da classe 1 resultante do ataque.

De forma geral, as principais diferenças entre as estratégias Indiferente ao Ataque (IA) e Ciente do Ataque (CA) se devem principalmente à capacidade da estratégia CA de deslocar capacidade da classe 1, sob ataque, para capitalizar sobre uma maior taxa de processamento da classe 2. A estratégia CA evita desperdiçar capacidade com



(a) Classe 1 (b) Classe 2

Figura 7.3: Taxa de Processamento Válido das Aplicações no Cenário 1

requisições ilegítimas, aumentando assim o valor total do lucro do provedor e a taxa de processamento de requisições legítimas.

Quanto mais pesada a carga de trabalho da classe 2, maior a fração de recursos transferidas para ela e, conseqüentemente, maiores os ganhos de lucro de CA sobre IA. Por exemplo, nos instantes 5000 e 30000 quando a carga da classe 2 está alta (Figura 7.1-a), o lucro do provedor na estratégia CA é significativamente maior que da estratégia IA (Figura 7.2), ao custo de uma redução na taxa de processamento da classe 1 (Figura 7.3-a) e aumento da taxa de processamento da classe 2 (Figura 7.3-b). Porém, se a carga de trabalho da classe 2 é baixa, não existe nenhum benefício no deslocamento de capacidade e ambas as soluções são equivalentes. Por exemplo, no intervalo entre os segundos 12000 e 18000 a carga da classe 2 é muito baixa (Figura 7.1-a) e não há benefício em deslocar capacidade para ela; as taxas de processamento de cada classe são similares nas estratégias IA e CA (Figura 7.3), o que, por fim, leva a lucros similares (Figura 7.2).

Se o ataque é muito pesado, ou seja, se λ_i^- / λ_i for muito próximo de 1, a classe vítima pode ser totalmente desligada pela estratégia CA para maximizar os ganhos obtidos da classe 2. Por exemplo, no experimento realizado isto acontece até o segundo 7000 (repare a Figura 7.3-a), devido à baixa taxa de chegada de requisições legítimas (Figura 7.1-a).

Ao longo de toda a simulação, a estratégia CA resulta em ganhos de lucro significativos sobre IA (16% na média). Da perspectiva do cliente, custos e tempos de resposta não são alterados, mas a taxa de processamento da classe de aplicação vítima

é severamente penalizada (41%), enquanto a classe 2 beneficia-se de capacidade extra, atingindo taxa de processamento válida total 34% maior (Figura 7.3).

A atenção é agora direcionada para as estratégias com contratos SLA adaptativos, sobre as quais clientes e provedores podem concordar para reduzir as penalidades na taxa de processamento válido das classes vítimas, discutidas acima. Na estratégia de Custo Adaptativo (CA-C), o maior valor possível para u_i^s é utilizado (isto é, $u_i^s = 1$), que é um cenário limite equivalente a dizer que o cliente paga por todas as requisições legítimas e ilegítimas processadas. Nesse caso, ao contrário da estratégia CA, a estratégia CA-C não desloca capacidade da classe vítima, pois o custo extra pago completamente elimina o impacto do ataque no lucro do provedor. De fato, o lucro da estratégia CA-C é o maior das quatro soluções, na média 59% maior que o da estratégia CA. Além disso, a taxa de processamento da vítima é tão alta quanto na estratégia IA e, na média, 70% maior que a obtida com CA. Porém, esses ganhos vêm juntos de um custo por requisição legítima processada que é quase 7 vezes maior, na média, o que pode ser interessante apenas para as aplicações mais críticas.

A estratégia de Tempo de Resposta Adaptativo (CA-R) tira vantagem do relaxamento do tempo de resposta ($w_1^s = 5$) para admitir um maior número de requisições da classe 1, aumentando a utilização das máquinas virtuais. Comparada com CA, essa estratégia aumenta a taxa de processamento legítima da vítima em 18%, na média, com mínimo impacto na taxa de processamento da classe 2, ao contrário de CA-C. Em contrapartida, o 90^o percentil dos tempos de resposta aumenta de 0,03s para 0,13s. Se o atraso maior for aceitável, CA-R pode ser uma opção interessante com bom custo-benefício para aplicações sobre ataque. Do ponto de vista do provedor, CA-R resulta em lucros um pouco maiores (1,2%) que a estratégia CA.

Esses resultados estão sumarizados na Tabela 7.4, que mostra as médias do lucro do provedor por intervalo de controle, o custo por requisição e a taxa de processamento válida dos clientes. Em resumo, quando comparado à estratégia Indiferente ao Ataque (IA), a estratégia Ciente do Ataque (CA) leva a maiores lucros através do aumento da taxa de processamento da classe 2, enquanto a taxa de processamento da vítima decresce significativamente. A estratégia de Custo Adaptativo (CA-C) resulta nos maiores lucros, e a classe vítima tem a mesma taxa de processamento que na estratégia IA, mas com um custo por requisição significativamente maior. Por fim, a estratégia de Tempo de Resposta Adaptativo (CA-R) consegue aumentar o lucro e a taxa de processamento, ao custo de tempos de resposta mais longos para a classe

Estratégia	Valores Médios				
	Lucro/s	Classe 1		Classe 2	
		Custo/Req	Reqs/s	Custo/req	Reqs/s
IA	518	1	74,7	0,97	456
CA	604	1	44,0	0,92	689
CA-C	959	6,89	74,7	0,97	455
CA-R	611	1	51,9	0,91	687

Tabela 7.4: Sumário do Impacto de Ataques de Segurança no Cenário 1

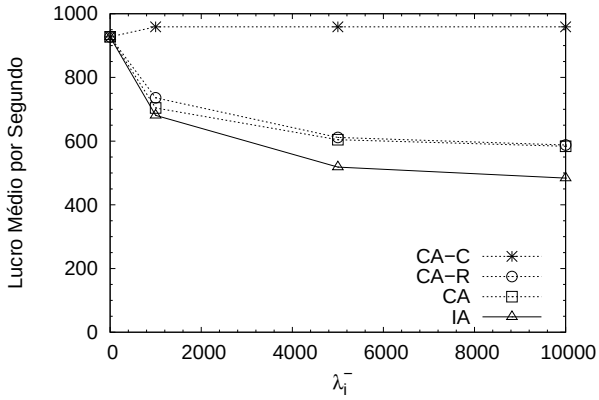


Figura 7.4: Impacto da Taxa de Ataque no Lucro do Provedor

vítima (não mostrado).

O impacto da variação da taxa de ataque é avaliado a seguir. A Figura 7.4 mostra que a diferença entre as estratégias CA e IA aumenta com λ_i^-, como esperado. Menos intuitivo é o pequeno aumento no lucro do provedor durante ataques para a estratégia CA-C. Isto deve-se ao fato de que, quando existir, capacidade ociosa da infra-estrutura é utilizada para servir, e capitalizar sobre, requisições ilegítimas. Na carga sintética considerada isso acontece por volta do instante 19000. Quanto mais capacidade ociosa tiver disponível, maiores serão os ganhos no lucro atingidos pela estratégia CA-C.

Sobre as diferenças entre as estratégias CA e CA-R, vale a pena mencionar três pontos chaves. Primeiro, diferenças do lucro e taxa válida de processamento dependem do peso do ataque. Quanto mais pesado o ataque for, menor será a fração de requisições legítimas servidas, e menor será o benefício de relaxar o R_i^{SLA} da vítima. A Figura 7.4 mostra que o lucro da solução CA-R se aproxima do lucro da estratégia CA à medida que λ_i^- aumenta. As curvas da taxa válida de processamento para ambas as classes têm comportamento similar.

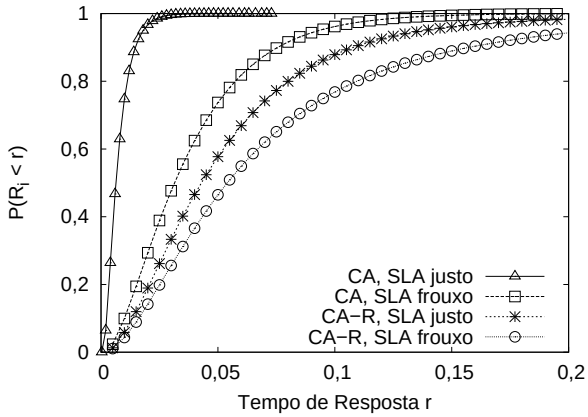


Figura 7.5: Impacto da Estratégia CA-R no Tempo de Resposta

Segundo, quanto mais justo o R_i^{SLA} original (isto é, quanto menor ele for), maior o impacto de relaxá-lo. Como a relação entre a utilização de uma máquina virtual e o tempo de resposta das requisições é dado pela distribuição hipoexponencial (Equação 5.3), os ganhos de relaxar o R_i^{SLA} original são maiores quando ele está antes do joelho da curva. Esse fato é ilustrado na Figura 7.5, que mostra as distribuições de tempo de resposta para ambas estratégias CA e CA-R com R_i^{SLA} original igual a $10d$ (justo) e $50d$ (frouxo), onde $d = \sum_{i=1}^{K} d_{i,j}^{*}$. A distância entre as curvas das estratégias CA e CA-C em cada caso (justo e frouxo), nos fornece indicação do aumento da utilização das máquinas virtuais, e conseqüentemente, taxa válida de processamento. O relaxamento do R_i^{SLA} justo dá muito mais espaço para melhoria. De fato, os ganhos médios na taxa válida de processamento da vítima resultantes da estratégia CA-R vão de 77% para o SLA justo a 4% para o SLA frouxo, sem impacto significativo na taxa de processamento da classe 2. De forma similar, ganhos no lucro obtido pela estratégia CA-R sobre CA vão de 3,1% a 0,5% do SLA justo ao frouxo.

Por fim, todos os resultados mostrados são para um fator de relaxamento $w_i^s = 5$. Valores maiores levam a resultados similares pois as utilizações das máquinas virtuais já estão próximas do máximo permitido ($\nu_{i,j}$). Valores menores podem ser utilizados para selecionar um compromisso entre CA-R e CA.

7.2.2 Cenário 2 – Carga Realista

Esta seção considera o cenário 2, constituído de cargas realistas. Considera-se ainda a ocorrência de 4 ataques com durações e taxas λ_i^- derivadas de [47]. Ataques incidem sobre as classes 1, 2, 4 e 5, cada um com início (a partir do início da

Classe i	Início	Duração (min)	λ_i^- (reqs/s)
1	00h10m44s	10	24
2	17h00m00s	30	261
3	–	–	–
4	36h26m32s	10	518
5	03h00m00s	300	62
6	–	–	–

Tabela 7.5: Características dos Ataques Realistas

Intervalo	Classe i	Métrica	Valores Médios			
			IA	CA	CA-C	CA-R
Todos Ataques	$i \in [1,6]$ (Média)	Lucro/s	37,7	48,1	53,6	48,7
		Custo/req	0,96	0,92	1,36	0,91
		90% R_i (s)	0,39	0,39	0,39	0,78
		Reqs/s	39,4	52,4	39,3	53,8
Ataque na Classe 5	$i = 4$	Custo/req	1	0,89	1	0,89
		90% R_i (s)	0,39	0,39	0,39	0,39
		Reqs/s	13,4	17,4	13,4	17,3
	$i = 5$	Custo/req	1	1	3,15	1
		90% R_i (s)	0,39	0,39	0,39	2,32
		Reqs/s	7,0	2,5	7,0	3,3

Tabela 7.6: Sumário do Impacto de Ataques de Segurança no Cenário 2

simulação), duração e taxa de chegada de requisições ilegítimas mostrados na Tabela 7.5. Objetivando enfatizar os compromissos relativos aos ataques, a Tabela 7.6 mostra os resultados médios – lucro do provedor, custo por requisição, 90^o do tempo de resposta e taxa de processamento – para o intervalo total onde pelo menos uma das classes está sobre ataque, ignorando assim intervalos quando não ocorre ataque. Os valores médios obtidos durante o intervalo de tempo quando a classe 5 estava sobre ataque, isto é, entre 03h00m e 03h05m, também são mostrados.

Os mesmos compromissos observados no cenário 1, gerado a partir de cargas sintéticas e discutidos na Seção 7.2.1, podem também ser observados neste cenário, mais realista. A estratégia Ciente do Ataque (CA) melhora significativamente o lucro do provedor ao custo de uma penalização da taxa de processamento válida da vítima. A estratégia de Custo Adaptativo (CA-C) elimina as penalidades com um aumento significativo nos custos, enquanto a estratégia de Tempo de Resposta Adaptativo (CA-R) provê melhoria para a vítima com um comprometimento do seu tempo de resposta. Os ganhos de CA-R são modestos devido ao SLA original frouxo.

7.3 Custos e Restrições de Energia

Esta seção apresenta os resultados e compromissos relativos a custos e restrições de energia. Experimentos com a carga sintética e real são discutidos nas Seções 7.3.1 e 7.3.2, respectivamente. Esta seção não considera ataques de segurança, fazendo $\lambda_i^- = 0$ para $i \in [1, N]$.

7.3.1 Cenário 1 – Carga Sintética

O principal objetivo desta seção é mostrar os compromissos chaves que surgem quando custos de energia são considerados no gerenciamento de capacidade, comparando as estratégias Ciente de Energia (CE) e ciente de energia com Tempo de Resposta Adaptativo (CE-R) quando economia de energia é desejável ($S = 0$) e quando é obrigatória ($S > 0$). Os benefícios de simplesmente considerar custos de energia, isto é, as diferenças entre as estratégias Ciente de Energia (CE) e Indiferente à Energia (IE), são discutidos no final desta seção.

Camadas têm capacidades iguais ($s_1 = s_2 = 1$, $C = 2$), e custos de energia por camada são escolhidos como uma porcentagem dos ganhos máximos possíveis, dados por $c_i C/(d^*_{i,1} + d^*_{i,2})$, a taxa máxima de processamento vezes o maior valor pago por cada unidade de processamento. Dado um custo total de energia fixo, duas configurações são consideradas, a saber: custos homogêneos fixos em 18% (ou seja, $e_1 = e_2 = 240$), e custos heterogêneos de 30% na primeira camada ($e_1 = 400$) e 6% na segunda ($e_2 = 80$). Os resultados mostrados continuam válidos para outros custos de energia. Esses valores elevados são escolhidos de forma a enfatizar os compromissos.

A avaliação começa considerando a solução CE para $S = 0$. O lucro do provedor, mostrado na Figura 7.6, é igual para ambas configurações homogênea e heterogênea pois o custo total de energia é fixo. A Figura 7.7-a mostra a taxa de processamento válido por classe para cada configuração. A taxa de processamento da classe 2 no cenário homogêneo (omitida) é similar ao da classe 1, deslocada no tempo de acordo com a carga de trabalho. Para o caso heterogêneo, porém, é mais vantajoso desligar mais recursos da camada mais cara (camada 1), removendo capacidade da classe com maior demanda (classe 1) naquela camada. Conseqüentemente, a classe 1 é penalizada, como mostrado na Figura 7.7-a, com sua taxa de processamento decrescendo em até 8%. Ocasionalmente, a redução da taxa de processamento da classe 1 pode levar a alguns dos recursos da camada 2 serem deslocados para a classe 2, favorecendo-a.

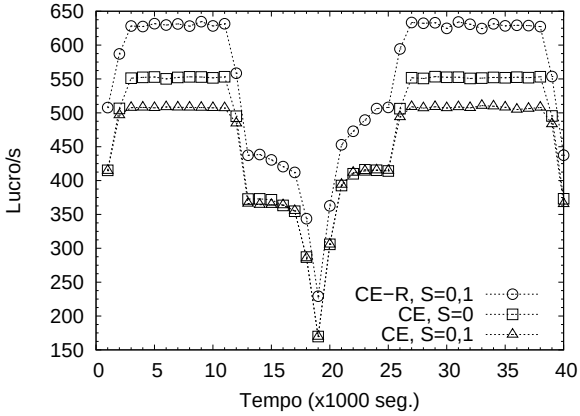
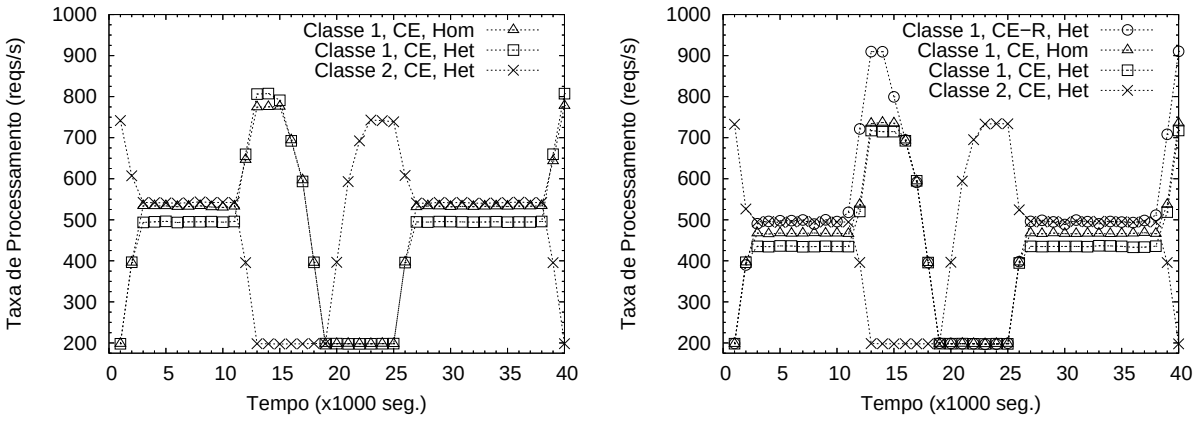


Figura 7.6: Lucro do Provedor Considerando Custos e Restrições de Energia



(a) $S = 0$ (b) $S = 0{,}1$

Figura 7.7: Taxa de Processamento Considerando Custos e Restrições de Energia

Solução	Configuração	Valores Médios				
		Lucro/s	λ_1^{ac}	λ_2^{ac}	$\sum_i f_{i,1}$	$\sum_i f_{i,2}$
CE	$S = 0$ / Homogêneo	473	491	491	0,94	0,94
	$S = 0$ / Heterogêneo	476	475	490	0,89	0,97
CE	$S = 0,1$ / Homogêneo	444	447	447	0,87	0,87
	$S = 0,1$ / Heterogêneo	451	427	461	0,83	0,91
CE-R	$S = 0,1$ / Homogêneo	546	518	519	0,84	0,84
	$S = 0,1$ / Heterogêneo	549	486	548	0,82	0,86

Tabela 7.7: Sumário do Impacto de Custos e Restrições de Energia no Cenário 1

Esses resultados são sumarizados na Tabela 7.7, que mostra valores médios, ao longo de toda a simulação, do lucro, taxa admitida de requisições e capacidade alocada em cada camada. Note que a alocação de recursos nas camadas para as duas classes, mostradas nas duas colunas mais à direita, é feita de forma distinta no caso da configuração heterogênea, onde uma maior fração da camada 1 é desligada.

Quando existe uma restrição de energia de 10% ($S = 0,1$), o lucro e as taxas de processamento de ambas as classes diminuem devido à capacidade reduzida, como mostrado nas Figuras 7.6 e 7.7-b. Os mesmos compromissos observados para $S = 0$ valem neste cenário. Porém, na configuração heterogênea, CE é ainda mais agressivo em remover capacidade da classe 1 na camada 1. As Figuras 7.7-a e 7.7-b mostram que a taxa de processamento da classe 2 é menos penalizada com a redução de capacidade (6% contra 10% da classe 1, na média). Além disso, a Tabela 7.7 mostra que, comparando as duas configurações para $S = 0,1$, onde a taxa média de processamento da classe 1 decresce 5%, a da classe 2, com menor demanda na camada mais cara, é favorecida com um aumento de 3% no caso heterogêneo.

Agora é analisada a eficácia da estratégia CE-R, baseada em contratos SLA com tempo de resposta adaptativos, quando $S = 0,1$. O fator de inflacionamento w_i^e é fixado em 25 para ambas as classes. O lucro e a taxa de processamento da classe 1 estão mostrados nas Figuras 7.6 e 7.7-b, respectivamente. Resultados médios são sumarizados na Tabela 7.7. Comparado à estratégia CE, o relaxamento do tempo de resposta leva a 22% de aumento no lucro para ambas as configurações. No caso heterogêneo, as taxas média de processamento das classes 1 e 2 aumentam em 14% e 19%, respectivamente. Ambas as classes têm taxa de processamento 16% maior na configuração homogênea. Estes ganhos vêm ao custo de um inflacionamento de 3 vezes no 90^o percentil do tempo de resposta.

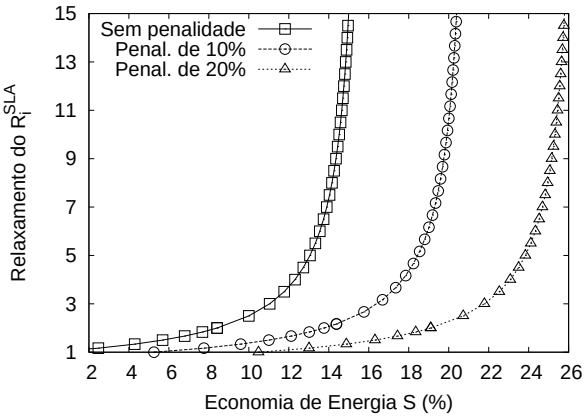


Figura 7.8: Impacto de CE-R na Taxa de Processamento

Para compreender melhor o compromisso entre o relaxamento do R_i^{SLA} e a taxa de processamento válida quando há restrições de energia, apenas a classe 1 é considerada, executando sobre a configuração homogênea, com $\lambda_1^{ac} = 330$ requisições por segundo, para vários valores de S (redução de capacidade). A Figura 7.8 mostra, para valores de S crescentes, o relaxamento $w_1^e S$ que precisa ser usado para que a taxa de processamento da classe 1 não seja penalizada (isto é, seja mantida fixa). Por exemplo, para reduzir 10% da energia utilizada ($S = 0,1$), $R_1^{SLA,e}$ precisa ser 2,5 vezes maior que R_i^{SLA}. Para pequenas reduções de energia ($S < 6\%$), um pequeno relaxamento do tempo de resposta é suficiente para garantir que não ocorra penalizações. Porém, a quantidade de relaxamento necessária cresce para infinito rapidamente. Nesse exemplo, reduções de energia maiores que 18% não são possíveis sem penalizar a taxa de processamento, pois a taxa de processamento máxima atingível é limitada pela utilização das máquinas virtuais (que por sua vez são limitadas por $\nu_{i,j}$) e, conseqüentemente, pela capacidade alocada. A Figura 7.8 também mostra curvas com penalizações de 10% e 20% na taxa de processamento, ilustrando os compromissos entre o fator de relaxamento e a degradação da taxa de processamento.

A respeito dos resultados na Tabela 7.7, o valor de $w_i^e = 25$ utilizado é grande o suficiente para garantir que não haverá penalizações na taxa de processamento. Valores menores de w_i^e levam a resultados intermediários, convergindo para o comportamento da estratégia Ciente de Energia (CE) quando $w_i^e = 0$.

A respeito dos benefícios de se considerar custos de energia, ou seja, ganhos da estratégia Ciente de Energia (CE) sobre a estratégia Indiferente à Energia (IE), os experimentos realizados mostram que, como em trabalhos anteriores [10], a estratégia CE provê grandes ganhos através do desligamento de recursos ociosos durante períodos

Solução	Configuração	Valores Médios				
		Lucro/s	λ_1^{ac}	λ_2^{ac}	$\sum_i f_{i,1}$	$\sum_i f_{i,2}$
IE	$S = 0$ / Homogêneo	904	732	732	1,0	1,0
CE	$S = 0$ / Homogêneo	1088	725	725	0,27	0,27

Tabela 7.8: Impacto de Considerar Custos de Energia para Cargas Leves

de carga leve, aumentando a utilização dos recursos. Resultados para um cenário 1 com carga mais leve, no qual os valores de $d_{i,j}^*$ e R_i^{SLA} foram divididos por 1,5, são mostrados na Tabela 7.8. Nesse cenário, a estratégia CE economiza 73% de energia, aumentando o lucro do provedor em 20%.

Arcabouços de gerenciamento de capacidade que objetivam economizar energia através do desligamento de componentes precisam considerar também o impacto do tempo de reinício (*reboot*) dos componentes depois que eles são desligados. Durante períodos de reinício os componentes consomem energia mas não são usados para servir requisições, o que pode impactar o lucro do provedor. Esse impacto depende da freqüência com a qual os componentes são ligados e desligados, o que está associado à variabilidade da alocação de capacidade, e, conseqüentemente, à estabilidade do sistema. Uma análise preliminar do impacto do tempo de reinício foi feita para o cenário 1. Mesmo numa configuração pessimista onde o tempo de reinício do *hardware* é metade do intervalo de controle, foram constatadas perdas no lucro de no máximo 4%. Em cenários onde a duração do tempo de reinício é uma fração pequena do intervalo de controle, o impacto no lucro é ainda menor.

7.3.2 Cenário 2 – Carga Realista

Para o cenário 2 são consideradas as mesmas configurações do cenário 1. Considera-se casos em que economia de energia é desejável ($S = 0$) e obrigatória ($S = 0,1$). Considera-se também configurações com custo total de energia fixo, com custos homogêneos iguais a 5% ($e_1 = e_2 = 5$) e custos heterogêneos iguais a 8,35% ($e_1 = 8,35$) e 1,65% ($e_2 = 1,65$). Os valores idênticos de custo e e_i devem-se ao fato do lucro potencial máximo, $c_i C/(d_{i,1}^* + d_{i,2}^*)$, ser igual a 10 no cenário 2. Para a estratégia com Tempo de Resposta Adaptativo (CE-R) foi usada uma constante de inflacionamento $w_i^e = 25$ para todas as aplicações.

A Tabela 7.9 mostra valores médios ao longo da simulação do lucro do provedor, taxa de processamento total de todas as classes e capacidade alocada em cada

Solução	Configuração	Valores Médios			
		Lucro/s	$\sum_i \lambda_i^{ac}$	$\sum_i f_{i,1}$	$\sum_i f_{i,2}$
CE	$S = 0$ / Homogêneo	40,7	54,0	0,87	0,88
	$S = 0$ / Heterogêneo	40,7	54,1	0,84	0,93
CE	$S = 0,1$ / Homogêneo	37,9	48,1	0,78	0,82
	$S = 0,1$ / Heterogêneo	38,0	48,0	0,74	0,89
CE-R	$S = 0,1$ / Homogêneo	51,5	60,8	0,74	0,75
	$S = 0,1$ / Heterogêneo	51,7	60,3	0,69	0,82

Tabela 7.9: Sumário do Impacto de Custos e Restrições de Energia no Cenário 2

camada. Novamente, os mesmos compromissos discutidos para a carga sintética podem ser observados nesse cenário mais realista. Na configuração heterogênea, o arcabouço desliga uma maior quantidade de recursos da camada mais cara (camada 1); esta diferença de alocação aumenta quando há restrições de energia. Na configuração heterogênea, classes com menor demanda por recurso nas camadas mais caras são favorecidas (classes 2, 4 e 6) e atingem maiores taxas de processamento que as demais classes (não mostrado). Por último, a estratégia CE-R com a alta constante de inflacionamento $w_i^e = 25$ consegue servir uma maior quantidade de requisições e aumentar o lucro do provedor, ao custo de um inflacionamento de 3,5 vezes no 90^o percentil do tempo de resposta.

8 Conclusões e Trabalhos Futuros

Com o crescimento global da Internet ao longo dos últimos anos, um número crescente de empresas vem utilizando terceirização de capacidade computacional como uma abordagem financeiramente atrativa para hospedar seus serviços. O gerenciamento dessas infra-estruturas de hospedagem se tornou uma tarefa árdua para humanos devido à complexidade que surge da grande quantidade de fatores envolvidos: requisitos de qualidade, alta variabilidade de carga, contratos SLA (possivelmente flexíveis) e múltiplas camadas. Isso evidencia a necessidade de sofisticados arcabouços para gerenciar de forma autônoma a capacidade disponível.

Nesta dissertação foi apresentado um novo arcabouço autônomo de gerenciamento de capacidade para infra-estruturas virtualizadas multicamadas que visa maximizar o objetivo de negócio do provedor. Construído a partir de um arcabouço prévio de camada única [3, 4], o novo arcabouço compartilha com o anterior o modelo de negócio de dois níveis. Porém, ele combina um modelo de desempenho de múltiplas filas muito mais preciso, que captura contenção em camadas específicas em uma aplicação e o paralelismo inerente a arquiteturas multicamadas, e um modelo de optimização estendido muito mais desafiador.

O arcabouço de gerenciamento proposto vai além dos objetivos tradicionais de desempenho capturando outros fatores que impactam a operação da infra-estrutura como ataques de segurança, custos e restrições de energia. Frente a esses fatores, o objetivo do arcabouço é minimizar o impacto negativo que eles causam no lucro do provedor. São considerados ainda contratos SLA adaptativos, que visam diminuir o impacto desses fatores nos objetivos do cliente.

Foram executados experimentos de simulação em vários cenários de configuração, cobrindo diferentes áreas do espaço de projeto. Cargas sintéticas foram usadas para levantar e apresentar os principais compromissos relevantes em cada cenário. Os compromissos discutidos para as cargas sintéticas também foram observados nas

cargas realistas. Isso indica que as conclusões obtidas podem ser estendidas para cenários práticos.

As principais conclusões são discutidas na Seção 8.1. Possibilidades de extensão e trabalhos futuros são discutidos na Seção 8.2.

8.1 Conclusões

As principais conclusões obtidas a partir dos experimentos realizados, apresentados nos Capítulos 6 e 7, são:

- O arcabouço autônomo multicamadas escala bem para cenários práticos. Com o tempo de solução do modelo de otimização tipicamente abaixo de 1 segundo, ele pode ser utilizado em tempo real.
- A nova abordagem autônoma é significativamente mais eficaz do que alocação estática de recursos para cargas de trabalho pesadas e heterogêneas.
- O arcabouço prévio de camada única, mais simples e impreciso, leva a decisões de alocação muito conservadoras que, por fim, comprometem sua eficácia. Ele é inferior às abordagens autônomas multicamadas e em alguns casos até mesmo à alocação estática de capacidade.
- A análise da sensibilidade do arcabouço à premissa de tempos de serviço exponencialmente distribuídos levou à conclusão que ele pode ser utilizado para distribuições de tempos de serviço com variabilidade moderada (coeficientes de variação menor que 3).
- Durante períodos em que alguma aplicação está sob ataque de segurança, o arcabouço desloca capacidade da classe sobre ataque para outras, aumentando o lucro do provedor (em até 16% na estratégia Ciente do Ataque (CA)) ao custo de uma redução na taxa de processamento válida da classe vítima.
- Quando a carga das aplicações hospedadas está leve, o arcabouço desliga recursos para economizar energia. Além disso, o arcabouço minimiza o impacto de restrições de energia no lucro do provedor, favorecendo aplicações com menor demanda por recursos na camada com maior custo de energia.

- Estratégias com SLA dinâmicos são alternativas razoáveis para reduzir a degradação na taxa de processamento válida. O impacto de um ataque pode ser mitigado por meio do aumento do custo de cada requisição processada. O relaxamento do tempo de resposta também reduz a degradação devido a um ataque ou restrição de energia, apesar dos seus benefícios dependerem do SLA original, do fator de relaxamento e das utilizações máximas.

8.2 Trabalhos Futuros

Trabalhos futuros estendendo o arcabouço apresentado podem ser divididos em duas direções: aprimoramento do arcabouço proposto e o seu enriquecimento por meio da modelagem de novos fatores. Alguns tópicos interessantes são discutidos abaixo.

- O arcabouço proposto nesta dissertação trata apenas de cargas transacionais. Para cargas onde os usuários acessam o serviço através de uma sessão, uma seqüência de requisições interdependentes, o arcabouço é apenas uma aproximação. A modelagem explícita de sessões de usuários trará maior precisão para esses tipos de carga, e abrirá caminho para modelos de negócio mais ricos, como a priorização de sessões, ou requisições dentro do contexto de uma sessão, com maior potencial de lucro para o provedor [8, 66].

- No arcabouço proposto, toda a capacidade destinada a uma classe é utilizada para o processamento de apenas uma requisição. Em várias aplicações executando em aglomerados (*clusters*) ou múltiplos nós de processamento, porém, a capacidade disponível é utilizada para servir várias requisições em paralelo, a uma velocidade menor. A modelagem específica desse comportamento, por exemplo através de filas M/M/m [54], pode aumentar a precisão do arcabouço para esse tipo de aplicação.

- Ainda com relação à modelagem da infra-estrutura, os modelos de desempenho e otimização podem ser flexibilizados para permitir deslocamento de capacidade entre as camadas, isto é, retirar capacidade de uma camada e adicionar em outras. Isto permitiria soluções ainda melhores para cenários onde aplicações têm contenção por recursos em camadas distintas.

- Novos contratos SLA adaptativos podem ser projetados para capturar fatores relacionados ao usuário final no modelo de negócio. Esses contratos relaxam métricas de qualidade de serviço que impactam a satisfação do usuário final, a utilidade que a aplicação tem para ele e, por fim, a quantia que ele está disposto a pagar pelo serviço [50]. Um modelo de negócio unificado que capture a satisfação do usuário final pode ser mais realista.

- Métodos alternativos de solução do problema de maximização do lucro do provedor, como programação dinâmica ou pesquisa combinatória, podem ser considerados caso complicações no modelo de otimização adicionadas por extensões futuras sejam muito difíceis de contornar.

- Com relação à modelagem de ataques de segurança, a modelagem de ataques semânticos, brevemente discutida no Apêndice A, é um fator importante visto o potencial impacto destes ataques [30]. Além disso, a modelagem de métodos de defesa ativos que utilizam recursos da infra-estrutura para minimizar o impacto do ataque [105, 106] pode trazer ganhos significativos em cenários onde estas defesas estão implantadas.

- Com relação à modelagem de custo de energia, tecnologias de controle dinâmico de freqüência do processador podem ser consideradas como um meio de aumentar a economia de energia. A modelagem dessas tecnologias implica na adição de uma relação entre consumo de energia e freqüência de processamento que é não-linear. Além disso, o desgaste do *hardware* resultante do ato de ligá-lo ou desligá-lo, discos rígidos em especial, deve ser capturado pois a substituição de componentes resulta em significativo custo adicional.

Apesar dos resultados positivos obtidos na avaliação realizada por meio de simulação nesta dissertação, estes precisam ser confirmados em uma implantação real. Nesse sentido, a prototipação do arcabouço de gerenciamento de capacidade em um sistema real, permitiria medir diretamente o desempenho da infra-estrutura, que é simplificada no simulador. Esta prototipação é um trabalho futuro interessante e poderia ser feita utilizando o Xen [16] como camada de virtualização.

APÊNDICE A – Modelando Ataques Semânticos

O modelo de desempenho apresentado na Seção 5.2 para ataques de segurança supõe que os tempos de serviço de requisições legítimas e ilegítimas seguem a mesma distribuição, e particularmente têm o mesmo valor médio. Esse modelo não captura ataques semânticos ou de inundação, que possuem demandas por serviço significativamente diferentes das requisições legítimas. Como apresentado em [30], existem ataques onde as requisições são construídas de forma a gastarem mais recursos que as requisições normais. Existem ainda ataques que enviam uma inundação de pacotes pequenos [71], que demandam poucos recursos individualmente mas causam impacto devido à grande quantidade. Este apêndice propõe uma modelagem desses tipos de ataques. Sua integração aos modelo de desempenho e otimização, bem como sua avaliação, são deixados como trabalho futuro.

Nesse cenário, requisições de cada classe i são aceitas para processamento na infra-estrutura com taxa λ_i^{ac}. Uma fração $\phi_i = \lambda_i^+ / \lambda_i$ dessas requisições terá uma demanda média por recursos em cada camada j dada por $d_{i,j}^+$. A fração restante, $1 - \phi_i$, corresponde a requisições ilegítimas e terão uma demanda média por recursos dada por $d_{i,j}^-$. Esse cenário é ilustrado na Figura A.1.

Supondo que o tempo de serviço das requisições legítimas e ilegítimas sejam dados por distribuições exponenciais com média $d_{i,j}^+$ e $d_{i,j}^-$, respectivamente, para $i \in [1,N]$ e $j \in [1,K]$, então o tempo de serviço de uma requisição (área sombreada na figura), segue uma distribuição hiperexponencial [98], dada por:

$$P[X \leq x] = \phi_i(1 - e^{-x/d_{i,j}^+}) + (1 - \phi_i)(1 - e^{-x/d_{i,j}^-}) \tag{A.1}$$

Uma alternativa a esta modelagem é usar uma fila M/M/1 para modelar o comportamento do centro de serviço mostrado na Figura A.1. A média do tempo de serviço de requisições legítimas e ilegítimas da classe i na camada j, a ser usada para

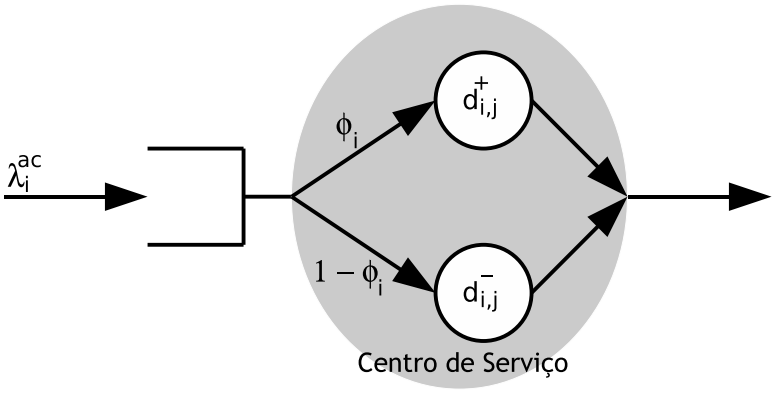


Figura A.1: Centro de Serviço Durante Ataques com Demandas Diferentes

parametrizar a fila M/M/1, é dada por $\phi_i d_{i,j}^{+} + (1-\phi_i) d_{i,j}^{-}$. Porém, a distribuição hiperexponencial tem variância maior que a distribuição exponencial [98] e, conseqüentemente, essa alternativa pode ser imprecisa. Como visto na Seção 6.5, imprecisões são significativas se o coeficiente de variação da distribuição do tempo de serviço for maior do que 3. Esse pode ser o caso, por exemplo, de ataques muito pesados com valores de $d_{i,j}^{-}$ muito diferentes de $d_{i,j}^{+}$. Para capturar esse tipo de ataques, são necessários modelos mais complexos com filas M/G/1 ou G/G/1, como os discutidos no Apêndice B, cujo projeto e avaliação deixamos como trabalho futuro.

APÊNDICE B – Modelos de Desempenho Para Distribuições Genéricas

Um modelo de desempenho baseado em filas M/M/1 pode não ser preciso para cenários onde a distribuição do tempo entre chegadas ou de serviço não é exponencial. O objetivo deste apêndice é dar uma direção inicial para a modelagem de distribuições de tempos entre chegadas e tempos de serviço genéricas no arcabouço através de um modelo de desempenho baseado em seqüências de filas G/G/1, utilizando aproximações existentes na literatura. Um modelo baseado nessas filas poderia ser utilizado para modelar sistemas com um nível arbitrário de variabilidade.

Para ser utilizado no arcabouço proposto, o modelo de desempenho precisa, em primeiro lugar, ser capaz de calcular a utilização da máquina virtual da classe i na camada j, $\rho_{i,j}$. Para filas G/G/1, a utilização é calculada da mesma forma que para filas M/M/1, resultando $\rho_{i,j} = \lambda^e_{i,j} d_{i,j}$, onde $\lambda^e_{i,j}$ é a taxa de chegada de requisições [54]. Porém, o maior desafio do novo modelo é prover uma estimativa da cauda da distribuição do tempo de resposta, isto é, $P[R_i \geq R_i^{ANS}]$.

Como uma expressão fechada para a cauda da distribuição do tempo de resposta em uma fila G/G/1 não existe [54], faz-se necessário o uso de aproximações. Este apêndice foca na aproximação da cauda da distribuição do tempo de resposta fornecida pela Desigualdade de Chebyshev (Equação 5.4), devido a seus resultados mais precisos que os atingidos pela aproximação dada pela Desigualdade de Markov [3,4].

Para utilizar a Desigualdade de Chebyshev, precisa-se de aproximações para $E[R_i]$ e $VAR[R_i]$, a média e a variância do tempo de resposta de uma requisição. O tempo de resposta é composto do tempo de serviço e do tempo de espera na fila, ou seja, $R_i = S_i + W_i$. Como a média e a variância do tempo de serviço, $E[S_i]$ e $VAR[S_i]$, são conhecidos de sua distribuição, são necessárias apenas aproximações para a média e variância do tempo de espera na fila, $E[W_i]$ e $VAR[W_i]$. Os valores de $E[R_i]$ e $VAR[R_i]$ são dados por $E[R_i] = E[S_i] + E[W_i]$ e $VAR[R_i] = VAR[S_i] + VAR[W_i]$ [98].

B.1 Média e Variância do Tempo de Espera em Filas G/G/1

Para estimar a média e a variação do tempo de espera de filas G/G/1 será utilizada a aproximação apresentada por Whitt [109]. As distribuições de tempos entre chegadas e tempos de serviço em cada fila j são denotados por $A_j(t)$ e $B_j(x)$, respectivamente, como mostrado na Figura B.1. Por exemplo, se $A_1(t)$ e $B_1(x)$ são distribuídos exponencialmente, então a fila 1 é uma fila M/M/1.

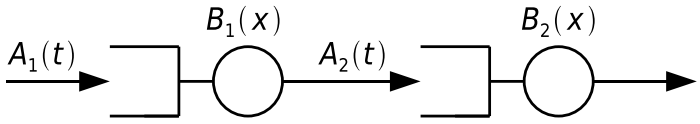


Figura B.1: Rede de Filas em Seqüência

A análise de uma fila individual é apresentada a seguir. Sob a premissa de independência entre os tempos de serviço em cada camada, as equações abaixo podem ser usadas em todas as filas do modelo de desempenho. Para maior clareza, os índices i e j são dispensados. O tempo de espera é denotado por W. As variáveis aleatórias de de $A(t)$ e $B(x)$ são denotadas por t e x, respectivamente. Pela aproximação de Whitt, tem-se que:

$$\mathrm{E}[W] \approx \psi \frac{(c_B^2 + c_A^2)}{2} E_{M/M/1}[W] \tag{B.1}$$

Onde $c_A^2 = \sigma_A^2/(E[A])^2$ e $c_B^2 = \sigma_B^2/(E[B])^2$ são os coeficientes de variação das distribuições de tempos entre chegadas e tempos de serviço, respectivamente. $E_{M/M/1}[W]$ é o tempo médio de espera de uma fila M/M/1, com o mesmo tempo médio de serviço que a fila G/G/1. ψ é dado por:

$$\psi = \begin{cases} \dfrac{c_B^2 \Psi(\frac{c_A^2 + c_B^2}{2}) + 4(c_A^2 - c_B^2)}{4c_A^2 - 3c_B^2} & c_A^2 \geq c_B^2 \\ \dfrac{(3c_A^2 + c_B^2)\Psi(\frac{c_A^2 + c_B^2}{2}) + (c_B^2 - c_A^2)e^{\frac{-2(1-\rho)}{3\rho}}}{2c_A^2 + 2c_B^2} & c_A^2 < c_B^2 \end{cases} \tag{B.2}$$

Onde ρ é a utilização do centro de serviço e Ψ é dado por:

$$\Psi(c^2) = \begin{cases} 1 & c^2 \geq 1 \\ \left(\frac{1}{2}e^{\frac{-2(1-\rho)}{3\rho}} + \frac{1}{2}\right)^{2(1-c^2)} & c^2 < 1 \end{cases} \tag{B.3}$$

Essas fórmulas são usadas para estimar o tempo médio de espera em uma fila G/G/1 dados os coeficientes de variação das distribuições de tempos entre chegadas e tempos de serviço e sua utilização. Nos cenários de maior interesse, como o descrito no Apêndice A, onde o coeficiente de variação do tempo de serviço é maior do que 1, Ψ sempre terá o valor 1, por que seu parâmetro, $(c_A^2 + c_B^2)/2$, sempre será maior que 1. Por completude, expressamos aqui a fórmula para $E_{M/M/1}[W]$, denotando por $\overline{x}$ a média de $B(x)$:

$$E_{M/M/1}[W] = \frac{\rho\overline{x}}{1-\rho} \tag{B.4}$$

Para estimar a variância do tempo de espera, a aproximação de Whitt [109] para o coeficiente de variação do tempo de espera é utilizada:

$$c_W^2 \approx \frac{\frac{4(1-\rho)\overline{x^3}/\overline{x}^3}{3(1+c_B^2)^2} + 2\rho - P[W>0]}{P[W>0]} \tag{B.5}$$

Onde $P[W>0]$ é a probabilidade de enfileiramento (a mesma probabilidade do tempo de espera ser maior que zero), dada abaixo, e $\overline{x^3}$ é o terceiro momento da distribuição do tempo de serviço.

$$P[W>0] \approx \rho^2\pi_1 + (1-\rho^2)\pi_2 \tag{B.6}$$

onde

$$\pi_1 = min\Big\{1, \rho\frac{1-\Phi((1+c_B^2)(1-\rho)/(c_A^2+c_B^2))}{1-\Phi(1-\rho)}\Big\} \tag{B.7}$$

$$\pi_2 = min\Big\{1, \rho\frac{1-\Phi(2(1-\rho)(1+c_A^2))}{1-\Phi(1-\rho)}\Big\} \tag{B.8}$$

e $\Phi(\cdot)$ é a distribuição normal padrão.

Essa aproximação necessita do terceiro momento da distribuição de serviço, $\overline{x^3}$, que pode ser facilmente calculado se $B(x)$ for conhecida ou estimado num ambiente pré-produção. A aproximação também captura a média das distribuições de tempos entre chegadas e tempos de serviço em ρ. Por último, são usados os coeficientes de variação das distribuições, c_A^2 e c_B^2.

De todos esses parâmetros, apenas ρ varia com a capacidade alocada à aplicação, $f_{i,j}$. Os parâmetros $\overline{x^3}$, c_A^2 e c_B^2 são constantes. Essa é uma característica essencial para que esse modelo possa ser utilizado em modelos de otimização. Outro fator importante para a implementação deste modelo de desempenho é que a distribuição normal padrão, $\Phi(\cdot)$, pode ser muito bem aproximada por um polinômio.[1]

Por último, o coeficiente de variação da distribuição de chegada na camada $j+1$ é uma função dos desvios padrão da distribuição de chegada e serviço da camada j, podendo ser aproximado pela fórmula:

$$c_{A_{j+1}}^2 \approx (1-\rho^2)c_{A_j}^2 + \rho^2 c_{B_j}^2 \tag{B.9}$$

B.2 Média e Variância do Tempo de Espera em Filas M/G/1

Esta seção apresenta um modelo de desempenho mais simples que o anterior, baseado em filas M/G/1, para estimar a cauda da distribuição do tempo de resposta. Para essas filas, os momentos da distribuição do tempo de serviço podem ser utilizados para calcular o valor exato do k-ésimo momento da distribuição do tempo de espera, $\overline{w^k}$, segundo a Equação 5.112 em [54]:

$$\overline{w^k} = \frac{\lambda}{1-\rho} \sum_{i=1}^{k} \binom{k}{i} \frac{\overline{x^{i+1}}}{(i+1)} \overline{w^{k-i}} \tag{B.10}$$

onde λ é a taxa de chegada de requisições, exponencialmente distribuída.

Esses momentos podem ser utilizados para calcular $E[W] = \overline{w}$ e $VAR[W] = \overline{w^2} - \overline{w}^2$. Fórmulas para $\overline{w}$ e $\overline{w^2}$ são apresentadas a seguir; note que, como na aproximação de Whitt para filas G/G/1, é necessário o terceiro momento da distribuição do tempo de serviço:

$$\overline{w} = \frac{\lambda \overline{x^2}}{2(1-\rho)} \tag{B.11}$$

$$\overline{w^2} = \frac{\lambda \overline{x^3}}{3(1-\rho)} \tag{B.12}$$

[1] Como o AMPL não possui a função $\Phi(\cdot)$ embutida, foram utilizadas aproximações polinomiais de sexto e décimo-segundo grau para esta função em um protótipo, com resultados satisfatórios.

Apesar da distribuição de chegadas em uma fila M/G/1 ser exponencialmente distribuída, a distribuição de saídas não é, como mostrado na Equação B.9. A única fila que apresenta essa propriedade é a fila M/M/m [54]. Para ser utilizado no arcabouço multicamadas proposto, um modelo baseado em seqüências de filas M/G/1 precisa aproximar a distribuição de saída de cada fila, que serve de entrada para a próxima, como uma distribuição exponencial.

Felizmente, experimentos preliminares indicam que o erro causado por essa aproximação é amenizado pelo erro decorrente do uso da Desigualdade de Chebyshev, pois o primeiro subestima a variabilidade do sistema enquanto o segundo superestima o tempo de resposta. Especificamente, o tempo de resposta superestimado pela Desigualdade de Chebyshev foi suficiente para anular o erro da aproximação em todos os cenários onde a distribuição dos tempos de serviço tinham coeficiente de variação menor do que 15.

APÊNDICE C – Implementação do Modelo de Otimização

Este apêndice apresenta a implementação do modelo de otimização em AMPL [35], uma linguagem de modelagem para programação matemática, na Seção C.1. A Seção C.2 apresenta o pseudocódigo da implementação do algoritmo de procura em árvore que decide qual subconjunto das classes hospedadas deve permanecer ligado para maximizar o lucro do provedor, evitando o problema de alocação de capacidade para classes que não estão sendo processadas descrito na Seção 5.3.1.

C.1 Implementação em AMPL

A implementação do modelo de otimização foi feita em AMPL [35], que facilita muito a solução do problema por permitir a expressão do modelo em alto nível, deixando a solução para solucionadores (*solvers*) especializados de funcionamento de convergência conhecidos na literatura [40, 93, 101].

Parâmetros

A implementação começa com a definição dos parâmetros do arcabouço que servem de entrada para o modelo de otimização. Parâmetros são valores constantes em AMPL e não impactam a complexidade do modelo. Os valores dos parâmetros caracterizam uma instância do problema, e são informados antes do início do processo de solução.

```
set C;                                          # classes.
set K;                                          # centers.
param D      {c in C, k in K} >= 0;             # d_{i,j}.
```

```
param l      {c in C} >= 0;                    # lambda^{+}.
param ldos   {c in C} >= 0;                    # lambda^{-}.
param Rsla   {c in C} >= sum {k in K} D[c,k];  # R^{SLA}.
param Xx     {c in C} >= 0;                    # X^S.
param Xn     {c in C} >= 0, <= Xx[c];          # X^N.
param cost   {c in C} >= 0;                    # c_i.
param rev    {c in C} >= 0;                    # r_i.
param a      {c in C} >= 0, <= 1;              # alpha_i.
param u      {c in C, k in K} >= 0, <= 0.99;   # rho_i.
param e      {k in K} >= 0;                    # e_j.
param cap    >= 0;                             # C.
param s      {k in K} >= 0, <= C;              # s_j.
```

Variáveis de Decisão

Variáveis de decisão são os valores que os solucionadores variam para encontrar a solução ótima. No nosso modelo de otimização as variáveis de decisão são λ_i^{ac} e $f_{i,j}$. O tempo de solução de um modelo de otimização geralmente aumenta com o número de variáveis de decisão que ele considera (ver Figura 6.1). O valor inicial de $\lambda_i^{ac} = 0$ para $i \in [1, N]$. O valor inicial de $f_{i,j} = C/N(\sum_{j=1}^{K} s_j)$ para $i \in [1, N]$ e $j \in [1, K]$, que é simplesmente $1/N$ se não houver restrições de energia, ou seja, $C = \sum_{j=1}^{K} s_j$.

```
var la {c in C} >= 0,
        <= min(l[c] + ldos[c], Xx[c]*(l[c]+ldos[c])/l[c]), := 0;

var f {c in C, k in K} >= 0.01, <= 1, := cap/(sum{k in K}(s[k])*card{C});
```

Função Objetivo

A função objetivo do modelo de otimização, mostrada abaixo, combina g_i^{-} e g_i^{s} para calcular o lucro do provedor quando são considerados custos de energia e ataques de segurança, simultaneamente. O operador de construção de funções lineares-por-partes em AMPL (<<;,>>) é usado. Assim, a função objetivo é transformada num conjunto de restrições lineares se for côncava [35].

```
maximize profit:
```

```
    (sum {c in C} <<min(Xn[c], l[c])*(l[c]+ldos[c])/l[c];
    cost[c]*l[c]/(l[c]+ldos[c]), rev[c]*l[c]/(l[c]+ldos[c])>> la[c])
    - (sum {k in K} (sum {c in C} f[c,k]*e[k]));
```

Restrições

Como várias das restrições do modelo de otimização já foram especificadas acima na definição dos parâmetros e variáveis de decisão, apenas as restrições que não estão relacionadas à definição de variáveis e seus limites é que precisam ser explicitadas para o AMPL. A restrição (e) sobre a capacidade alocada em cada camada e a restrição (g) sobre a utilização de uma máquina virtual são dadas abaixo:

```
subject to utilization {c in C, k in K}:
    la[c]*D[c,k]/(l[c]+ldos[c]) <= u[c,k]*f[c,k];

subject to capacity {k in K}:
    sum {c in C} f[c,k] <= 1;
```

A restrição sobre a capacidade total alocada na infra-estrutura durante restrições de energia, mostrada na Equação 5.7, é escrita da seguinte maneira:

```
subject to totalcapacity:
    sum {k in K} (s[k]*sum {c in C} f[c,k]) <= cap;
```

A última restrição a adicionar na implementação em AMPL do modelo de otimização é a restrição (h) sobre a cauda da distribuição do tempo de resposta. Abaixo está mostrada a restrição usando a distribuição hipoexponencial (Equação 5.3). Sempre que a distribuição não está definida – ou seja, $\gamma_{i,j} = \gamma_{i,k}$ com $j \neq k$ – uma aproximação dada pela distribuição de Erlang é utilizada. Além disso, a restrição implementada abaixo é aplicável apenas para o cenário onde $K = 2$. Para outros valores de K, a restrição precisa ser reescrita.

```
subject to delay {c in C}:
    (if abs(f[c,2]/D[c,2] - f[c,1]/D[c,1]) > 0.000001
    then
        (1/(f[c,2]/D[c,2] - f[c,1]/D[c,1]))*(
```

```
        (f[c,2]/D[c,2] - la[c])*exp(-(f[c,1]/D[c,1] - la[c])*Rsla[c]) -
        (f[c,1]/D[c,1] - la[c])*exp(-(f[c,2]/D[c,2] - la[c])*Rsla[c])
        )
    else
        (exp(-Rsla[c]*((f[c,1]/D[c,1] + f[c,2]/D[c,2])/2 - la[c])) +
            (Rsla[c]*((f[c,1]/D[c,1] + f[c,2]/D[c,2])/2 - la[c])) *
            exp(-Rsla[c]*((f[c,1]/D[c,1] + f[c,2]/D[c,2])/2 - la[c]))
        )
    ) <= a[c];
```

Por último, a restrição sobre a cauda da distribuição do tempo de resposta pode ser feita utilizando a aproximação dada pela Desigualdade de Chebyshev (Equação 5.4). Essa aproximação é muito mais simples que a distribuição exata mostrada acima, mas vem acompanhada de uma restrição adicional para restringir as variáveis de decisão à sua parte convexa.

```
subject to delaydomain {c in C}:
    Rsla[c] - sum {k in K} (1/(f[c,k]/D[c,k] - la[c])) >= 1e-4;

subject to delay {c in C}:
    sum {k in K} (1/((f[c,k]/D[c,k] - la[c])**2)) <=
    a[c]*(Rsla[c] - sum {k in K} (1/(f[c,k]/D[c,k] - la[c])))**2;
```

C.2 Procura em Árvore pela Solução Ótima

Nesta seção é apresentado o pseudocódigo do algoritmo descrito na Seção 5.3.1 para encontrar o conjunto das aplicações que maximiza o lucro do provedor. Ele decide quais aplicações devem ser totalmente desligadas – isto é, as classes que devem ter $\lambda_i^{ac} = f_{i,j} = 0$ para $i \in [1, N]$ e $j \in [1, K]$ – através de uma procura em árvore do tipo ramificar e limitar em todo o espaço de possibilidades. A pesquisa começa a partir do nó raíz da árvore (Algoritmo 1) e continua deste ponto recursivamente como descrito no Algoritmo 2. Uma solução aceitável é aquela que atinge lucro suficientemente próximo, por exemplo 90%, de um limite superior para o lucro do provedor, que pode ser obtido através da solução do modelo de otimização sem a restrição (h).

Algoritmo 1 Procura pelo Conjunto de Aplicações que Maximiza o Lucro do Provedor

Requer: $C \leftarrow$ instância do problema com todas as classes i.
Garante: M é aceitável ou é a melhor solução possível.
  $M \leftarrow$ ramifica_e_limita$(C, -\infty)$

Algoritmo 2 ramifica_e_limita(C, M)

Requer: $C \leftarrow$ uma instância do problema de otimização.
Requer: $M \leftarrow$ a melhor solução conhecida.
  se $C < M$ **então**
    retorne C
  se C é aceitável **então**
    retorne C
  $M \leftarrow C$
  para todo $i \in [1, N]$ **faça**
    se $i \in C$ **então**
      $N \leftarrow$ ramifica_e_limita($C - i$, M)
      se N é aceitável **então**
        retorne N
      senão se $N > M$ **então**
        $M \leftarrow N$
  retorne M

Referências Bibliográficas

[1] MySQL AB. MySQL Database.
`http://www.mysql.com`.

[2] B. Abraham and J. Ledolter. *Statistical Methods for Forecasting*. John Wiley and Sons, 1983.

[3] B. Abrahao. Gerenciamento Autônomo de Capacidade para Centro de Dados da Internet. *Dissertação de Mestrado, Departamento de Ciência da Computação, Universidade Federal de Minas Gerais*, 1(558), 2005.

[4] B. Abrahao, V. Almeida, J. Almeida, A. Zhang, D. Beyer, and F. Safai. Self-Adaptive SLA-Driven Capacity Management for Internet Services. In *10th IEEE/IFIP NOMS*, Vancouver, Canada, Abril 2006.

[5] The Green Grid Administration. The Green Grid.
`http://www.thegreengrid.org`.

[6] Audi AG. R8 OMG!
`http://www.audi.com/audir8/html/index.php???`

[7] J. Agosta, J. Chandrashekar, D. Dash, M. Dave, D. Durham, H. Khosravi, H. Li, S. Purcell, S. Rungta, R. Sahita, U. Savagaonkar, and E. Schooler. Towards Autonomic Enterprise Security: Self-Defending Platforms, Distributed Detection, and Adaptive Feedback. *Intel Technology Journal*, 10(4), 2006.

[8] S. Aiber, D. Gilat, A. Landau, N. Razinkov, A. Sela, and S. Wasserkrug. Autonomic Self-Optimization According to Business Objectives. In *1st IEEE ICAC*, New York, New York, Maio 2004.

[9] I. Akyldiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A Survey on Sensor Networks. *IEEE Communications Magazine*, 40(8), 2002.

[10] J. Almeida, V. Almeida, D. Ardagna, C. Francalanci, and M. Trubian. Resource Management in the Autonomic Service-Oriented Architecture. In *3rd IEEE ICAC*, Dublin, Ireland, Junho 2006.

[11] AMD. PowerNow! Technology Brief.
`http://www.amd.com/us-en/assets/content_type/DownloadableAssets/Power_Now2.pdf`.

[12] M. Arlitt and T. Jin. Workload Characterization of the 1998 World Cup Web Site. Technical Report HPL-1999-35R1, HP Laboratories, Setembro 1999.

[13] S. Axelsson. Intrusion Detection Systems: A Survey and Taxonomy. In *Technical Report, Chalmers University of Technology*, Götenborg, Sweden, Março 2000.

[14] G. Banga, P. Druschel, and J. Mogul. Resource Containers: a New Facility for Resource Management in Server Systems. In *3rd USENIX OSDI*, New Orleans, Louisiana, Fevereiro 1999.

[15] P. Barford, J. Kline, D. Plonka, and A. Ron. A Signal Analysis of Network Traffic Anomalies. In *2nd ACM SIGCOMM Workshop on Internet Measurement*, Marseille, France, Novembro 2002.

[16] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the Art of Virtualization. In *19th ACM SOSP*, Bolton Landing, NY, Outubro 2003.

[17] F. Baskett, M. Chandy, R. Muntz, and F. Palacios. Open, Closed, and Mixed Networks of Queues with Different Classes of Customers. *Journal of the ACM*, 22(2), Abril 1975.

[18] D. Bernstein. SYN Cookies. http://cr.yp.to/syncookies.html, Outubro 1996.

[19] M. Blazek, H. Chong, W. Loh, and J. Koomey. Data Centers Revisited: Assessment of the Energy Impact of Retrofits and Technology Trends in a High-Density Computing Facility. *Journal of Infrastructure Systems*, 10(3), 2004.

[20] N. Bobroff and A. Kochut an K. Beaty. Dynamic Placement of Virtual Machines for Managing SLA Violations. In *10th IEEE/IFIP IM*, Munich, Germany, Maio 2007.

[21] M. Buco, R. Chang, L. Luan, C. Ward, J. Wolf, and P. Yu. Utility Computing SLA Management Based upon Business Objectives. *IBM Systems Journal*, 43(1), 2004.

[22] J. Chase, D. Anderson, P. Thakar, A. Vahdat, and R. Doyle. Managing Energy and Server Resources in Hosting Centres. In *18th ACM SOSP*, Banff, Canada, Outubro 2001.

[23] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam. Managing Server Energy and Operational Costs in Hosting Centers. In *SIGMETRICS '05*, Banff, Alberta, Canada, Junho 2005.

[24] L. Cherkasova and P. Phaal. Session-Based Admission Control: A Mechanism for Peak Load Management of Commercial Web Sites. *IEEE Transactions on Computers*, 51(6), Junho 2002.

[25] Hewlett-Packard Development Company. HP-UX 11i PRM. `http://h20338.www2.hp.com/hpux11i/cache/317534-0-0-0-121.html`.

[26] Intel Corporation. Enhanced Intel SpeedStep Technology. `http://support.intel.com/support/processors/xeon/sb/CS-012641.htm`.

[27] Microsoft Corporation. dotNET Framework. `http://msdn.microsoft.com/netframework`.

[28] Microsoft Corporation. Windows Vista.
`http://www.microsoft.com/windows/products/windowsvista`.

[29] M. Cremonini and D. Nizovtsev. Understanding and Influencing Attackers' Decisions: Implications for Security Investment Strategies. In *5th WEIS*, Cambridge, England, Junho 2006.

[30] S. Crosby and D. Wallach. Denial of Service via Algorithmic Complexity Attacks. In *12th USENIX Security Symposium*, Washington, DC, Agosto 2003.

[31] S. Elnikety, E. Nahum, J. Tracey, and W. Zwaenepoel. A Method for Transparent Admission Control and Request Scheduling in E-Commerce Web Sites. In *13th ACM WWW*, New York, New York, Maio 2004.

[32] O. Etzion, A. Fisher, and S. Wasserkrug. e-CLV: A Modeling Approach for Customer Lifetime Evaluation in e-Commerce Domains, with an Application and Case Study for Online Auctions. In *2004 IEEE EEE*, Taipei, Taiwan, Março 2004.

[33] Apache Software Foundation. HTTP Server Project.
`http://httpd.apache.org`.

[34] The FreeBSD Foundation. The FreeBSD Project.
`http://www.freebsd.org`.

[35] R. Fourer, D. Gay, and B. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Boyd and Fraser, 1993.

[36] R. Ge, X. Feng, and K. Cameron. Performance-constrained Distributed DVS Scheduling for Scientific Applications on Power-aware Clusters. In *SC '05*, Seattle, WA, Novembro 2005.

[37] E. Gelenbe, R. Lent, and Z. Xu. Design and Performance of Cognitive Packet Networks. *Performance Evaluation*, 46(2-3), Outubro 2001.

[38] E. Gelenbe and G. Loukas. A Self-Aware Approach to Denial of Service Defence. *Computer Networks*, 51(5), 2007.

[39] T. Gil and M. Poleto. MULTOPS: A Datastructure for Bandwidth Attack Detection. In *10th USENIX Sec. Symp.*, Washington, DC, Agosto 2001.

[40] P. Gill, W. Murray, and M. Saunders. SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization. *SIAM Review*, 47(1), 2005.

[41] L. Goasduff and C. Forsling. Gartner Urges IT and Business Leaders to Wake up to IT's Energy Crisis, 2006.
`http://www.gartner.com/it/page.jsp?id=496819`.

[42] A. Goldfarb. The Medium-Term Effects of Unavailability. *Quantitative Marketing and Economics*, 4(2), 2006.

[43] L. Gomes, C. Cazita, J. Almeida, V. Almeida, and W. Meira. Workload Models of SPAM and Legitimate E-Mails. *Performance Evaluation*, 64(7), 2007.

[44] PostgreSQL Global Development Group. PostgreSQL Database. `http://www.postgresql.org`.

[45] C. Hsu and W. Feng. A Power-Aware Run-Time System for High-Performance Computing. In *SC '05*, Seattle, WA, Novembro 2005.

[46] D. Huang, Q. Hu, and R. Behara. Economics of Information Security Investment in the Case of Simultaneous Attacks. In *5th WEIS*, Cambridge, England, Junho 2006.

[47] A. Hussain, J. Heidemann, and C. Papadopoulos. A Framework for Classifying Denial of Service Attacks. In *ACM SIGCOMM*, Karlsruhe, Germany, Agosto 2003.

[48] IHC. InterContinental Hotels Club. `http://http://www.ichotelsgroup.com`.

[49] C. Jones, K. Sivalingam, P. Agrawal, and J. Chen. A Survey of Energy Efficient Network Protocols for Wireless Networks. *Wireless Networks*, 7(4), 2001.

[50] T. Kamoto and T. Hayashi. Analysis of Service Provider's Profit by Modeling Customer's Willingness to Pay for IP QoS. In *GLOBECOM 2002*, Taipei, Taiwan, Novembro 2002.

[51] S. Kandula, D. Katabi, M. Jacob, and A. Berger. Botz-4-Sale: Surviving Organized DDoS Attacks That Mimic Flash Crowds. In *2nd NSDI*, Boston, MA, Maio 2005.

[52] N. Kappiah, V. Freeh, and D. Lowenthal. Just In Time Dynamic Voltage Scaling: Exploiting Inter-Node Slack to Save Energy in MPI Programs. In *SC '05*, Seattle, WA, Novembro 2005.

[53] A. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure Overlay Services. In *ACM SIGCOMM*, Pittsburgh, Pennsylvania, Agosto 2002.

[54] L. Kleinrock. *Queueing Systems*. John Wiley and Sons, 1975.

[55] E. Lazowska, J. Zahorjan, and G. Graham K. Sevcik. *Quantitative Systems Performance: Computer Systems Analysis using Queueing Network Models*. Prentice Hall, 1984.

[56] C. Lefurgy, K. Rajamani, F. Rawson, W. Felter, M. Kistler, and T. Keller. Energy Management for Commercial Servers. *IEEE Computer*, 36(12), 2003.

[57] C. Lemuet, J. Sampson, J. Collard, and N. Jouppi. The Potential Energy Efficiency of Vector Acceleration. In *SC '06*, Tampa, FL, Novembro 2006.

[58] M. Lesk. The New Front Line: Estonia under Cyberassault. *IEEE Security & Privacy*, 5(4), 2007.

[59] W. Lin, Z. Liu, C. Xia, and L. Zhang. Optimal Capacity Allocation for Web Systems with End-to-End Delay Guarantees. *Performance Evaluation*, 62(1-4), Outubro 2005.

[60] X. Liu, X. Zhu, S. Singhal, and M. Arlitt. Adaptive Entitlement Control to Resource Containers on Shared Servers. In *9th IFIP/IEEE IM*, Nice, France, Maio 2005.

[61] Z. Liu, M. Squillante, and J. Wolf. On Maximizing Service-Level-Agreement Profits. *SIGMETRICS Performance Evaluation Review*, 29(3), Dezembro 2001.

[62] R. Mahajan, S. Bellovin, S. Floyd, J. Ioannidis, V. Paxon, and S. Shenker. Controlling High Bandwidth Aggregates in the Network. *ACM SIGCOMM Computer Communication Review*, 32, 2002.

[63] J. Markoff and S. Hansell. Hiding in Plain Sight, Google Seeks More Power. *The New York Times*, 2006.

[64] J. Mears and D. Connor. These Energy Efforts are Hot, Hot, Hot. In *Network World*, Junho 2006.

[65] D. Menascé and V. Almeida. *Scaling for E-Business: Technologies, Models, Performance, and Capacity Planning*. Prentice Hall, 2002.

[66] D. Menascé, V. Almeida, R. Fonseca, and M. Mendes. Business-Oriented Resource Management Policies for e-Commerce Servers. *Performance Evaluation*, 42(2-3), Outubro 2000.

[67] D. Menascé and M. Bennani. Autonomic Virtualized Environments. In *2nd IEEE ICAS*, Silicon Valley, CA, Julho 2006.

[68] Microsoft Corporation. Internet Information Services. `http://www.iis.net`.

[69] Sun Microsystems. Java Platform, Enterprise Edition. `http://java.sun.com/j2ee`.

[70] Sun Microsystems. Solaris Operating System. `http://www.sun.com/software/solaris`.

[71] J. Mirkovic and P. Reiher. A Taxonomy of DDoS Attack and DDoS Defense Mechanisms. *SIGCOMM Computer Communication Review*, 34(2), 2004.

[72] D. Moore, C. Shannon, D. Brown, G. Voelker, and S. Savage. Inferring Internet Denial-of-Dervice Activity. *ACM Transactions on Computer Systems*, 24(2), 2006.

[73] W. Morein, A. Stavrou, D. Cook, A. Keromytis, V. Misra, and D. Rubenstein. Using Graphic Turing Tests to Counter Automated DDoS Attacks Against Web Servers. In *10th ACM Conference on Computer and Communications Security*, Washington, DC, Outubro 2003.

[74] MPI. Message Passing Interface. `http://www-unix.mcs.anl.gov/mpi/`.

[75] MySpace.com. A Place for Friends. `http://www.myspace.com`.

[76] NCMEC. National Center for Missing & Exploited Children. `http://www.ichotelsgroup.com`.

[77] Oracle. Oracle Database. `http://www.oracle.com/database`.

[78] Linux Kernel Organization. The Linux Kernel Archives. `http://www.kernel.org`.

[79] V. Paxson and S. Floyd. Wide Area Traffic: the Failure of Poisson Modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, 1995.

[80] H. Perros and K. Elsayed. Call Admission Control Schemes: A Review. *IEEE Communications Magazine*, 34(11), 2003.

[81] F. Popovici and J. Wilkes. Profitable Services in an Uncertain World. In *ACM/IEEE SC*, Seattle, Washington, Novembro 2005.

[82] Rackspace. Managed Hosting. `http://www.rackspace.com`.

[83] M. Rappa. The Utility Business Model and the Future of Computing Services. *IBM Systems Journal*, 43(1), 2004.

[84] T. Rockafellar. *Convex Analysis*. Princeton University Press, 1996.

[85] M. Roesch. SNORT: Lightweight Intrusion Detection for Networks. In *13th LISA*, Seattle, Washington, Novembro 1999.

[86] J. Rolia, L. Cherkasova, M. Arlitt, and A. Andrzejak. A Capacity Management Service for Resource Pools. In *ACM 5th WOSP*, Palma, Spain, Julho 2005.

[87] J. Rolia, L. Cherkasova, and C. McCarthy. Configuring Workload Manager Control Parameters for Resource Pools. In *10th IEEE/IFIP NOMS*, Vancouver, Canada, Abril 2006.

[88] J. Rolia and K. Sevcik. The Method of Layers. *IEEE Transactions on Software Engineering*, 21(8), Agosto 1995.

[89] J. Ross and G. Westerman. Preparing for Utility Computing: The Role of IT Architecture and Relationship Management. *IBM Systems Journal*, 43(1), Janeiro 2004.

[90] C. Schuba, I. Krsul, M. Kuhn, E. Spafford, A. Sundaram, and D. Zamboni. Analysis of a Denial of Service Attack on TCP. In *IEEE Symposium on Security and Privacy*, Oakland, California, Maio 1997.

[91] A. Snoeren, C. Partridge, L. Sanchez, C. Jones, F. Tchakountio, S. Kent, and T. Strayer. Hash-Based IP Traceback. In *ACM SIGCOMM*, San Diego, California, Agosto 2001.

[92] O. Spatscheck and L. Petersen. Defending Against Denial of Service Attacks in Scout. In *3rd USENIX OSDI*, New Orleans, Louisiana, Fevereiro 1999.

[93] P. Spellucci. An SQP Method for General Nonlinear Programs Using Only Equality Constrained Subproblems. *Mathematical Programming*, 3(82), 1998.

[94] M. Steinder, I. Whalley, D. Carrera, I. Gaweda, and D. Chess. Server Virtualization in Autonomic Management of Heterogeneous Workloads. In *10th IEEE/IFIP IM*, Munich, Germany, Maio 2007.

[95] Symantec. Trends for Julho–Dezembro 06. *Symantec Internet Security Threat Report*, 11(1), 2007.

[96] Akamai Technologies. Web Application Performance Management. `http://www.akamai.com`.

[97] A. Totok and V. Karamcheti. Improving Performance of Internet Services Through Reward-Driven Request Prioritization. In *14th IEEE WQoS*, New Haven, Connecticut, Junho 2006.

[98] K. Trivedi. *Probability and Statistics with Reliability, Queuing and Computer Science Applications*. John Wiley and Sons, 2002.

[99] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi. An Analytical Model for Multi-Tier Internet Services and its Applications. In *ACM SIGMETRICS*, Banff, Canada, Junho 2005.

[100] B. Urgaonkar, P. Shenoy, A. Chandra, and P. Goyal. Dynamic Provisioning of Multi-tier Internet Applications. In *2nd IEEE ICAC*, Seattle, Washington, Junho 2005.

[101] R. Vanderbei. LOQO: An Interior Point Code for Quadratic Programming. *Optimization Methods and Software*, 11(1–4), 1999.

[102] D. Villela, P. Pradhan, and D. Rubenstein. Provisioning Servers in the Application Tier for E-Commerce Systems. *ACM Transactions on Internet Technology*, 7(1), Fevereiro 2007.

[103] VMware. Virtualization, Virtual Machine & Virtual Server Consolidation. `http://www.vmware.com`.

[104] M. Vrable, J. Ma, J. Chen, D. Moore, E. Vandekieft, A. Snoeren, G. Voelker, and S. Savage. Scalability, Fidelity, and Containment in the Potemkin Virtual Honeyfarm. In *20th ACM SOSP*, Brighton, United Kingdom, Outubro 2005.

[105] M. Walfish, M. Vutukuru, H. Balakrishnan, D. Karger, and S. Shenker. DDoS Defense by Offense. In *ACM SIGCOMM*, Pisa, Italy, Setembro 2006.

[106] X. Wang and M. Reiter. Defending Against Denial-of-Service Attacks with Puzzle Auctions. In *IEEE Symposium on Security and Privacy*, Oakland, California, Maio 2003.

[107] A. Whitaker, R. Cox, M. Shaw, and S. Gribble. Rethinking the Design of Virtual Machine Monitors. *IEEE Computer*, 38(5), 2005.

[108] A. Whitaker, M. Shaw, and S. Gribble. Scale and Performance in the Denali Isolation Kernel. In *5th USENIX OSDI*, Boston, Massachusetts, Dezembro 2002.

[109] W. Whitt. Approximations for the GI/G/m Queue. *Production and Operations Management*, 2(2), 1993.

[110] J. Wilkes, J. Mogul, and J. Suermondt. Utilification. In *11th ACM SIGOPS European Workshop*, Leuven, Belgium, Setembro 2004.

[111] Z. Xu and G. Bochmann. A Probabilistic Approach for Admission Control to Web Servers. In *SPECTS*, San Jose, California, Julho 2004.