BRUNO DIAS ABRAHÃO

GERENCIAMENTO AUTÔNOMO DE CAPACIDADE PARA CENTRO DE DADOS DA INTERNET

Belo Horizonte
02 de setembro de 2005

Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

GERENCIAMENTO AUTÔNOMO DE CAPACIDADE PARA CENTRO DE DADOS DA INTERNET

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.

BRUNO DIAS ABRAHÃO

Belo Horizonte
02 de setembro de 2005

Federal University of Minas Gerais
Instituto de Ciências Exatas
Graduate Program in Computer Science

AUTONOMOUS BUSINESS-DRIVEN CAPACITY MANAGEMENT FOR INTERNET DATA CENTERS

Dissertation presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

BRUNO DIAS ABRAHÃO

Belo Horizonte
September 2, 2005

UNIVERSIDADE FEDERAL DE MINAS GERAIS

FOLHA DE APROVAÇÃO

Gerenciamento Autônomo de Capacidade para Centro de Dados da Internet

BRUNO DIAS ABRAHÃO

Dissertação defendida e aprovada pela banca examinadora constituída por:

Ph. D. Virgílio A. F. Almeida – Orientador
Universidade Federal de Minas Gerais

Ph. D. Jussara Marques de Almeida
Universidade Federal de Minas Gerais

Ph. D. Antonio Alfredo F. Loureiro
Universidade Federal de Minas Gerais

Belo Horizonte, 02 de setembro de 2005

Resumo Estendido

1 Introdução

Serviços *on-line* tornaram-se fundamentais para as empresas realizarem seus negócios tradicionais através da Internet e implementarem processos informatizados de forma distribuída geograficamente. Devido ao custo e à complexidade de manutenção desses serviços, é crescente a opção dessas empresas pela terceirização de infra-estrutura computacional através de provedores de serviços que hospedam aplicações em centros de dados da Internet [34].

Os centros de dados da Internet são plataformas de computação compartilhadas que permitem o gerenciamento flexível de capacidade, com o objetivo de acomodar de forma efetiva as demandas específicas de cada cliente e minimizar custos. Do ponto de vista do cliente, a terceirização de recursos computacionais representa uma alternativa financeiramente atrativa. Em contraste à compra de hardware e software, contratação de pessoal capacitado e constantes manutenções e atualizações tecnológicas, são firmados contratos de hospedagem com provedores, que, por compartilharem a infra-estrutura computacional com vários clientes, conseguem obter benefício da economia em escala e, portanto, são capazes de oferecer um serviço mais vantajoso financeiramente que a hospedagem independente.

Os contratos firmados entre as partes são baseados em níveis de serviço [13]. Nesse acordo são estabelecidos, além da taxa de hospedagem, os requisitos de desempenho das aplicações e valores de multas contratuais acarretadas pela violação desses. Uma vez que a receita de um provedor provém principalmente desses contratos, a habilidade de obter lucros depende fundamentalmente da maturidade do gerenciamento dessas plataformas [34]. Conseqüentemente, os provedores direcionam seus esforços para otimizar a capacidade computacional disponível, ao mesmo tempo em que procuram satisfazer os requisitos de desempenho estabelecidos pelos clientes, evitando, assim, multas contratuais. Entretanto, a realização dessa tarefa de forma a considerar os requisitos, características e cargas de trabalho dos vários serviços hospedados simultaneamente, além de contemplar os objetivos de negócios do provedor, apresenta

enorme complexidade.

Nesse trabalho, apresentamos uma proposta de gerenciamento de capacidade que têm por finalidade orquestrar a plataforma computacional a favor dos objetivos de negócios do provedor, em vez de simplesmente acomodar os serviços de forma satisfatória. Na próxima subseção, descrevemos a abordagem em detalhes.

1.1 Descrição do Problema

Nesse trabalho, é proposto um modelo de gerenciamento de capacidade computacional para centro de dados da Internet no qual os objetivos de negócios do provedor são priorizados. À luz da hipótese de que o provedor é capaz de maximizar seu potencial financeiro, e ao mesmo tempo garantir a execução estável dos serviços hospedados, através do uso dessa abordagem, a formulação do modelo envolve questões de negócios e de sistema. Em contraste às propostas de gerenciamento mais populares na literatura, em que a satisfação dos requisitos de desempenho das aplicações é buscada, o modelo de gerenciamento de capacidade dirigido à negócios toma decisões de alocação e soluciona conflitos de necessidade de recursos orientado aos objetivos financeiros do provedor.

Entretanto, além da alta complexidade da arquitetura desses sistemas, características no domínio de negócios exacerbam o desafio apresentado pelo problema em consideração. Novas demandas de clientes introduzem dificuldades sem precedentes para o provedor. Primeiramente, há um recente crescimento de interesse em contratos de hospedagem onde clientes pagam proporcionalmente à utilização [28, 32, 44]. Conseqüentemente, os provedores de serviços são forçados a reformular planos de negócios para oferecer a contabilidade de serviço que contempla essa categoria de contrato. Em outro aspecto, a grande competitividade dos provedores resultou em contratos que prometem limites estritos na probabilidade das métricas de desempenho violarem um determinado alvo, ao invés de apenas garantir a média dos valores [23]. Ainda referente aos contratos, uma vez que a satisfação dos requisitos de desempenho das aplicações em relação a apenas uma métrica isoladamente não garante a qualidade da experiência dos usuários, os clientes vêm demonstrando interesse em garantias compostas de múltiplas métricas.

Paralelamente aos desafios do nível de negócios, a satisfação dos requisitos de desempenho dos serviços têm implicações diretas na receita obtida dos contratos de serviço [13]. Entretanto, aumentando ainda mais a complexidade do ambiente, os serviços da Internet apresentam cargas de trabalho com grande flutuação de intensidade no tempo, de tal forma que a necessidade de capacidade dessas aplicações se altera

drasticamente a todo o instante. Adicionalmente, as cargas de trabalho dos serviços apresentam ocasionais picos de demanda de curta duração [25], que podem acarretar multas severas ao provedor, caso esse não seja capaz de adaptar o sistema rapidamente para lidar com tais situações. Assim, o modelo de gerenciamento deve interagir com a plataforma, produzindo novas decisões de alocação de capacidade e configurações em tempo real.

Por último, a quantidade de variáveis e o ritmo das mudanças subjuga a capacidade humana de gerenciamento de tais sistemas. Dessa forma, para que seja possível maximizar o aproveitamento dos recursos, faz-se necessário o emprego de uma abordagem autônoma para reconfiguração dinâmica.

Na próxima subseção, são apresentadas nossas contribuições para lidar com esses desafios, contemplando as características supra-citadas através da abordagem de gerenciamento dirigida à negócios.

1.2 Contribuições

Os fatores descritos na subseção anterior têm direta influência no gerenciamento de capacidade de serviços da Internet. À luz dos desafios mencionados, apresentamos aqui as contribuições desse trabalho, que propõe um modelo para a combinação dos objetivos de negócios de alto-nível com decisões no nível de sistemas.

No âmbito de negócios, para permitir a contabilização de serviço proporcional à utilização, é proposto um modelo de múltiplos níveis para a especificação de requisitos de desempenho. Ilustramos esse ponto com o estudo de contratos de dois níveis, que definem dois modos de operação: normal e pico. O modo de operação normal corresponde ao nível de serviço contratado pelos clientes para satisfazer suas demandas usuais, enquanto no modo de operação pico é estabelecido um limite de nível de serviço, acima do nível normal, até o qual os clientes têm interesse em receber capacidade extra para evitar a degradação de desempenho de suas aplicações devido a ocasionais picos de demanda. O contrato de dois níveis funciona em conjunto com um modelo de custo baseado em multas, acarretadas ao provedor quando os requisitos do modo de operação normal são violados, e recompensas, recebidas pelo provedor quando o nível de serviço provido excede o nível estabelecido como normal. A proposta do modelo baseia-se na contabilidade de utilização através de métricas que oferecem um significado mais próximo da realidade dos clientes, isto é, o desempenho alcançado pelos serviços, ao invés de simplesmente medir a utilização de recursos individuais (ex: CPU, rede, disco etc.).

No nível de sistemas, é proposto um modelo de desempenho para mapear as características dos serviços, seus requisitos de desempenho e suas cargas de trabalho recebidas

em necessidades de capacidade. Em vista dos requisitos probabilísticos de desempenho, a previsão da distribuição de probabilidade do tempo de resposta, representa um obstáculo ao se empregar modelos de filas. Isto é devido a basicamente dois fatores: i) os resultados exatos para a derivação dessa probabilidade são conhecidos somente para alguns tipos de filas, os quais não podem ser aplicados para capturar realisticamente o comportamento dos diferentes tipos de serviços da classe considerada e ii) alguns dos resultados disponíveis fazem do modelo de otimização de difícil (ou ineficiente) solução. Como conseqüência, aproximações alternativas desses resultados são propostas para que possam ser avaliadas sob duas diferentes políticas de escalonamento, sob as quais as aplicações podem executar: primeiro a chegar, primeiro a ser servido (PCPS) e compartilhamento de processador (CP).

Visando de mapear os objetivos de negócios de alto nível em diretivas de configuração de sistema, a formulação de um problema de otimização combina os níveis de negócios e de sistemas do problema para gerar decisões de alocação de capacidade e configurações para a plataforma de forma a maximizar o resultado do balanço entre multas e recompensas do ponto de vista do provedor. Completando a abordagem, para que o sistema apresente rápida adaptabilidade em resposta à mudanças, o modelo de otimização é incorporado em uma estrutura autônoma de reconfiguração.

Finalmente, através da análise experimental, avaliamos a eficácia da abordagem proposta em relação aos objetivos fiananceiros do provedor e ao nível de confiabilidade oferecido aos serviços.

2 Centro de Dados da Internet

No capítulo 2, o ambiente considerado nessa dissertação, o centro de dados da Internet, é apresentado. Essa plataforma é composta de grandes agrupamentos de dispositivos de processamento e armazenamento, interligados através de uma rede de alta velocidade, e possui mecanismos para prover o isolamento do desempenho das aplicações hospedadas através dos mecanismos de virtualização [18].

O mecanismo de virtualização permite particionar a capacidade dos dispositivos e isolar as aplicações. Assim, ao invés de usar os recursos físicos diretamente, os serviços usam as partições isoladas, ou recursos virtuais, cuja capacidade representa uma fração da capacidade do dispositivo físico correspondente, permitindo o provedor contrair ou expandir a capacidade alocada para os serviços de forma flexível.

Nesse trabalho, consideramos os serviços da Internet transacionais, caracterizados por aplicações que recebem requisições, normalmente de uma população infinita de usuários, para recuperar ou atualizar informações [25]. Esses serviços são compostos

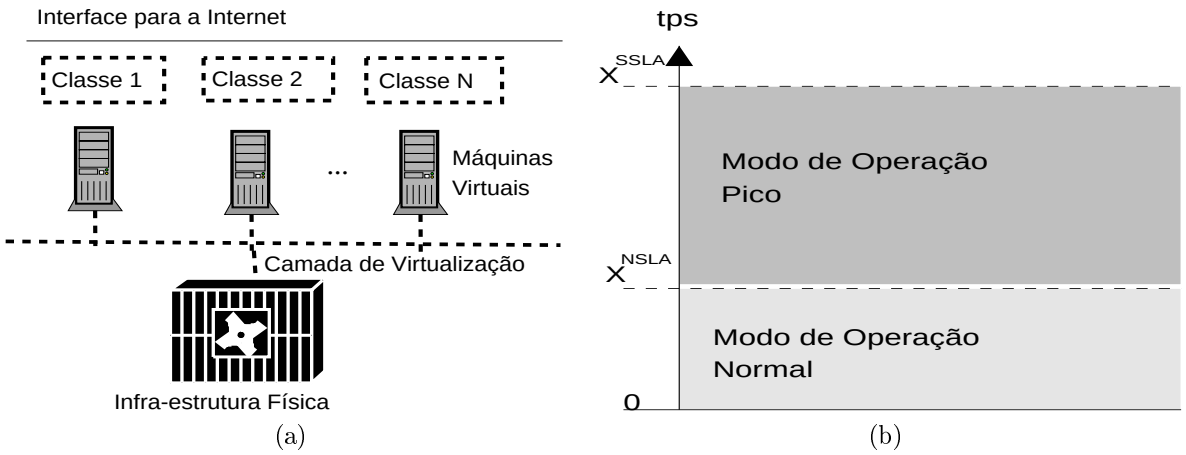


Figura 1: a) Plataforma de hospedagem do centro de dados da Internet. b) Exemplo de escala arbitrária com os escopos dos modos normal e pico.

de diferentes tipos de transação, sujeitas a diferentes cargas de trabalho, com diferentes demandas de serviço nos recursos e executadas por componentes de software independentes. Esses componentes são denotados nesse trabalho *classes de aplicações*. Sob a premissa de que suas funcionalidades são independentes daquelas de outras classes, esses componentes são analisados como aplicações independentes.

A plataforma de hospedagem considerada é apresentada na figura 1(a). Nesse ambiente, classes de aplicações, ao invés de utilizarem a infra-estrutura disponível diretamente, são hospedadas em máquinas virtuais (MV), que consistem de coleções de recursos virtuais. Por exemplo, uma máquina virtual pode conter uma CPU virtual, um disco virtual, uma rede virtual etc.. Dessa forma, são construídas diversas MVs cujas capacidades, além de isoladas, representam frações da capacidade física disponível. Uma vez que cada MV é dedicada para hospedar uma única classe, dizemos que as classes são isoladas. Finalmente, a configuração de alocação de capacidade é definida como a determinação das frações de capacidade que cada MV recebe de cada recurso do servidor físico correspondente.

Por último, em determinados momentos de suas operações, as classes podem receber um número de requisições superior à capacidade de processamento de suas respectivas máquinas virtuais, fazendo com que os serviços fiquem instáveis e que o tempo de resposta das transações cresça sem limites. Visando evitar essas situações indesejáveis, as MVs empregam um mecanismo de controle de admissão [31], através do qual requisições que possam causar qualquer tipo de instabilidade são rejeitadas.

3 Modelo de Gerenciamento de Capacidade

Os trabalhos relacionados ao gerenciamento de capacidade em centro de dados são descritos no capítulo 3. Ao longo da apresentação desses trabalhos, as propostas existentes na literatura são contrastadas com modelo aqui proposto, explicitando as principais contribuições da dissertação.

No capítulo 4, o modelo de gerenciamento autônomo de capacidade para centros de dados da Internet é apresentado. O capítulo se divide em quatro partes: um modelo de negócios baseado em penalidades e recompensas de nível de serviço, um modelo de desempenho, um modelo de otimização, que combina os dois modelos anteriores, e uma estrutura para gerenciamento autônomo de capacidade, onde o modelo de otimização é incorporado.

3.1 Modelo de Negócios Baseado em Penalidades e Recompensas de Níveis de Serviço

A primeira parte refere-se a um modelo de negócios baseado em contratos de nível de serviço que considera a contabilidade de serviço proporcional ao uso. O nível de serviço, por sua vez, é medido em relação à taxa de processamento alcançada, sujeita a um requisito de qualidade dos tempos de resposta das transações processadas. Esse requisito é expresso através do limite α da probabilidade das transações violarem um valor máximo R^{SLA} estabelecido para o tempo de resposta, ou seja, $P(R > R^{SLA}) \leq \alpha$, onde R é o tempo de resposta de uma única transação. Denominamos *taxa de processamento válida* aquela na qual as transações processadas satisfazem requisito de tempo de resposta supra-citado.

Para ilustrar o modelo de especificação de requisitos de múltiplos níveis, são estudados contratos dois níveis de requisitos, que definem dois modos distintos de operação: normal e pico. O modo de operação normal refere-se ao nível de serviço, demarcado pela taxa mínima de processamento válido X^{NSLA}, a qual os clientes contratam para satisfazer a demanda usual de suas aplicações, enquanto o modo de operação pico corresponde a um limite para essa métrica X^{SSLA} ($X^{SSLA} \geq X^{NSLA}$) estabelecido pelos clientes, até o qual o provedor tem incentivo para alocar capacidade extra para um serviço, a fim de evitar a degradação do desempenho, no caso de ocasionais picos de demanda. Uma escala arbitrária que ilustra o escopo dos dois modos de operação em relação à taxa de processamento é apresentada na figura 1(b).

Os modos de operação funcionam em conjunto com um modelo de custo baseado em penalidades, a que o provedor está sujeito caso viole o requisito de serviço do modo

de operação normal, e recompensas, recebidas pelo provedor ao operar a aplicação do cliente no modo pico. Por último, definimos o objetivo de negócios do provedor como a maximização do balanço entre penalidades e recompensas resultante da execução das aplicações hospedadas.

3.2 Modelo de Desempenho Baseado em Modelos de Filas

O modelo de desempenho é baseado em modelos de filas e mapeia os requisitos de nível de serviço, as características e cargas de trabalho das aplicações em necessidade de capacidade. Esse modelo deve determinar para cada MV, dado uma configuração da plataforma, a probabilidade das transações violarem o limite estabelecido para o tempo de resposta, a utilização e a taxa de processamento alcançadas. Além disso, essas métricas são estimadas sob duas políticas de escalonamento de serviço alternativas sob as quais os serviços transacionais estão freqüentemente submetidos: primeiro a chegar, primeiro a ser servido (PCPS) e compartilhamento de processador (CP).

A utilização e a taxa de processamento podem ser obtidas diretamente usando modelos de fila disponíveis. No entanto, uma expressão exata para calcular a probabilidade das transações violarem o valor limite para o tempo de resposta, que captura realisticamente as características de toda a classe de aplicações considerada, não é conhecida. Portanto, aproximações para esse fim são propostas nesse trabalho. Consideramos três aproximações alternativas para prever o comportamento das aplicações: baseadas na desigualdade de Markov [29], na desigualdade de Chebychev [29] e, como referência para fila M/M/1, na fórmula para calcular o percentil dos tempos de resposta [21]. Essas aproximações podem ser usadas alternativamente, resultando em diferentes graus de precisão, os quais são quantificados e avaliados nos resultados experimentais.

3.3 Modelo de Otimização

O principal componente da abordagem consiste em um modelo de otimização não-linear que combina o modelo de negócios com o modelo de desempenho e soluciona conflitos visando maximizar os objetivos de negócio do provedor.

A figura 2(a) ilustra a composição do modelo de otimização, incluindo suas entradas e saídas. As características das aplicações, os parâmetros de contrato de nível de serviço e a carga prevista para o próximo intervalo de controle constituem suas entradas; uma decisão de alocação de capacidade e os parâmetros do controle de admissão de cada MV, que maximizam os objetivos de negócio do provedor, são produzidos como saída. O modelo de otimização compõe o *Gerenciador de Capacidade*, que é incorporado à

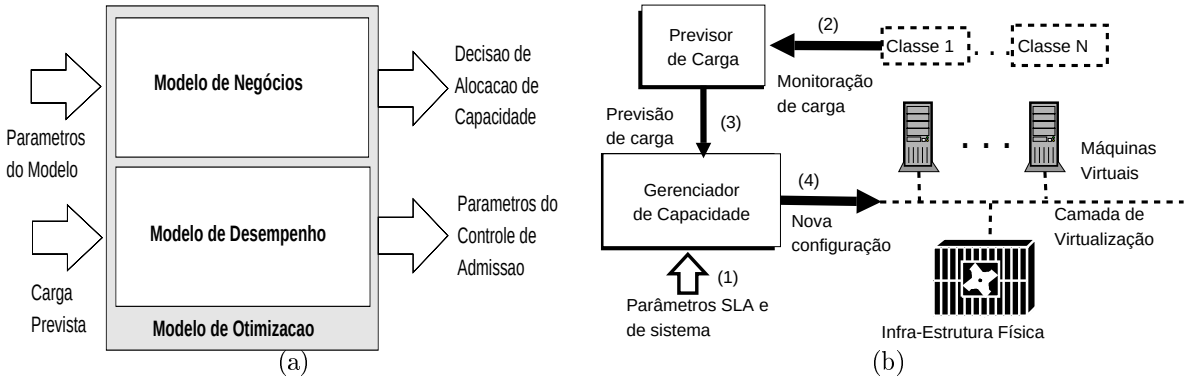


Figura 2: a) Composição do modelo de otimização. b) Laço de controle autônomo.

estrutura de gerenciamento autônomo, descrita a seguir.

3.4 Estrutura de Gerenciamento Autônomo

A estrutura para o gerenciamento autônomo de capacidade em centro de dados da Internet corresponde ao laço de controle mostrado na figura 2(b). O modelo de otimização é a parte principal do modelo e constitui o componente *Gerenciador de Capacidade*, o qual provê em tempo real as decisões de alocação de capacidade que visam maximizar o objetivo de negócios do provedor, dado o estado, as características e os requisitos das aplicações em algum momento da execução. Esse componente recebe como entrada os parâmetros de sistemas e de contratos de nível de serviço das aplicações (1) e interage periódicamente com o sistema entre intervalos denominados *intervalos de controle*.

Paralelamente, o componente *Previsor de Carga* monitora e armazena o nível de demanda recebido pelas diferentes classes (2). Quando o Gerenciador de Capacidade é invocado, previsões de carga para cada aplicação, referente ao próximo intervalo de controle, são geradas baseada no comportamento passado, para serem usadas como entrada do Gerenciador de Capacidade (3).

Após gerar uma nova configuração para o sistema, o Gerenciador de Capacidade envia comandos para a plataforma, a qual altera o mapeamento entre recursos virtuais e a infra-estrutura física na camada de virtualização e atualiza os parâmetros de controle de admissão nas MVs (4).

4 Análise Experimental e Conclusões

Através de simulações, avaliamos a eficácia da abordagem proposta e o grau de precisão produzido por cada uma das aproximações usadas no modelo de desempenho. Os experimentos consistem na execução concorrente das MVs, sujeitas a cargas sintéticas diferentes que seguem um processo Poisson não-homogêneo. As cargas variam no tempo, permitindo que os picos de requisições recebidos por uma MV complementem o período de baixa intensidade de carga de outra, sendo, dessa forma, possível avaliar a abilidade do modelo de gerenciamento de auto reconfiguração em resposta a mudança de prioridades de negócios. As simulações são instrumentadas para medir o desempenho das classes através de métricas como o tamanho da fila de espera, o tempo de resposta das transações e o resultado de penalidades e recompensas de cada classe.

Na análise de resultados numéricos, é realizada uma comparação entre a abordagem proposta e uma abordagem estática cuja configuração inicial é realizada visando a operação de longo prazo, baseado no valor médio das cargas de trabalho (conhecido a priori) durante todo o experimento. A abordagem proposta é combinada alternativamente com as três aproximações para a estimativa do comportamento probabilístico do tempo de resposta, considerando as políticas de escalonamento PCPS e CP.

Os resultados numéricos mostram primeiramente que, em comparação com a abordagem estática, a abordagem proposta, combinada com qualquer uma das aproximações do modelo de desempenho, apresenta vantagens financeiras para o provedor. Entretanto a aproximação baseada na desigualdade de Markov provê limites superiores relaxados e, dessa forma, superestima a necessidade de capacidade das aplicações. Conseqüentemente, essa aproximação produz resultados inferiores em relação à aqueles produzidos pelas aproximações baseadas na desigualdade de Chebyshev e a fórmula do percentil das filas M/M/1. Essa duas últimas produzem resultados com semelhante grau de precisão.

O sucesso da abordagem depende, além do aumento do potencial financeiro do provedor, da confiabilidade oferecida pelo sistema. Assim, observamos também, através dos valores de métricas de desempenho coletadas durante a simulação, que, em contraposição à abordagem estática, o modelo satisfaz essa condição, preservando a confiabilidade dos serviços hospedados, quando analisados sob a ótica do cumprimento dos requisitos de tempos de resposta e da estabilidade das filas das MVs.

Finalmente, o entendimento desse primeiro escopo do trabalho proporciona uma rica base de estudo para a exploração de novas idéias nos domínios de negócio e sistemas relacionados a esse ambiente. Assim, no capítulo 6 são oferecidas nossas conclusões e apresentadas direções para trabalhos futuros dentro do tópico aqui estudado.

Abstract

This work considers the problem of harnessing computing capacity in accordance to high-level business objectives for Internet Data Centers that host multiple third-party Internet services. For this purpose, we deal with issues at both business and systems domains. At the business domain, we propose a model that comprises i) a multi-level SLA specification that allows for per-use service accounting with respect to requirements of throughput and tail distribution response time, and ii) a penalty/reward service accounting model. At the systems domain, it is proposed alternative approximations for predicting the performance of the hosted services under two different scheduling disciplines, namely first come first served and processor sharing. With the objective of maximizing the provider's business potential, we propose an optimization model to combine business and systems strands, which is integrated into an autonomous capacity management framework, in an attempt to adapt the platform to changing needs in real time. Finally, through simulation, we assess the effectiveness of the proposed approach as well as the degree of accuracy resulting from each of the performance model approximations.

Agradecimentos

Dedico esse trabalho aos meus pais, Walter e Terezinha Abrahão, que sempre colocaram a minha educação como a maior prioridade, e à Cynthia, que, com todo o apoio, compreensão, paciência e carinho, me proporcionou muita alegria para conduzi-lo.

Agradeço, em primeiro lugar, ao Prof. Virgílio Almeida, orientador desse trabalho, que, com sua ampla experiência, foi fundamental para direcionar a pesquisa e prover idéias fecundas para seu desenvolvimento. À Profa. Jussara Almeida, por suas perspicazes observações e inúmeras contribuições, e ao Prof. Antônio Alfredo Loureiro, por seus diversos comentários na revisão do projeto como membro da banca avaliadora.

Aos pesquisadores da Hewllet-Packard Labs Palo Alto, Alex Zhang, mentor do meu programa de pesquisa nos Labs, Fereydoon Safai, Dirk Beyer e Cipriano Santos pela experiência de trabalho, pela colaboração para o desenvolvimento desse tópico de pesquisa e pela hospitalidade durante a minha estada em Palo Alto. Ao pessoal da Hewllet-Packard Brasil R&D, Cirano Silveira, Roque Scheer, Kátia Saikoski, Cláudia Pereira e João Jornada pelo suporte para a realização desse trabalho.

Agradeço ao pessoal do DCC-UFMG, que me ajudou com o esclarecimento de dúvidas e sugestões. Em especial, ao Prof. Wagner Meira Jr., Prof. Sabino José Ferreira Neto do Departamento de Estatística da UFMG, Fabrício Ceolin, Olga Goussevskaia e Bruno Diniz. Ao Dr. Pacheco, que foi um grande mentor durante toda essa fase, e aos amigos Eduardo Alves Valle Jr., Guilherme Rocha e Rodrigo Fonseca.

Devo agradecimentos também ao pessoal da secretaria do DCC-UFMG, que, com eficiência resolveu todos os problemas burocráticos durante a pós-graduação.

Finalmente, a dedicação a esse trabalho foi possível devido ao suporte financeiro da CAPES (PROF-2003) e da Hewllet-Packard Brasil R&D (CAMPS-UFMG-2004/2005). Agradeço ainda ao Prof. Virgílio Almeida e à Hewllet-Packard Brasil R&D pela oportunidade do estágio no HP Labs Palo Alto, que, além de ter ampliado meus horizontes, foi imprescindível para a realização desse trabalho.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the exponential growth in popularity over the past few years, Internet services, such as on-line banking, business to business applications, on-line shopping, etc., have become a popular solution for businesses to offer traditional services on-line and to deploy business operations in a distributed fashion. Due to the high cost and complexity of maintaining such services, businesses are increasingly relying on computing-based capacity outsourcing [34] as a financially attractive alternative to application hosting. In this scenario, SLA (Service Level Agreement) contracts [13] are established with a provider that supports a large diversity of services in shared Internet data centers (IDC).

IDCs are characterized as shared large-scale computing environments, consisting of centralized server farms with hundreds of processing and storage units, linked together through a high speed switched network fabric. In such environments, computer-based capacity is flexibly provided by mechanisms that can easily reconfigure the resource allocation, system parameters, and provide service differentiation in order to best attain a strategic goal, accommodate each customer unique needs and minimize costs. In contrast with businesses that independently host their services, which usually incurs high fixed costs with hardware and software purchase, requires operational efforts, management expertise and periodic upgrades, IDC providers derive benefit from economy of scale by sharing the infra-structure among multiple customers. In addition, as opposed to resource overprovision, which is often not cost-effective, IDC providers can take advantage of application multiplexing, where idle computing power previously assigned to an application can be flexibly re-assigned to an overloaded one. As a result, providers are able to attract customers by offering significantly smaller hosting costs.

In the SLA contracts, the parties agree upon the service rates, the minimum service expectations in terms of performance metrics (i.e., response time) and the respective penalties incurred by the provider for contract violations. Since the providers' revenue

results chiefly from the SLA business, their ability to generate value from the infrastructure depends heavily on the maturity of their management processes [34].

In this study, we work on the hypothesis that providers can maximize their business potential by optimally orchestrating the available computing power to its best business advantage, instead of simply accommodating applications satisfactorily. The problem investigated here falls into the general class of business-driven service management problems, where computing capacity is harnessed to best attain high-level business goals, and systems exhibit the ability of quickly reacting to business, user behavioral, and systems changes. However, since the effectiveness of the approach is heavily dependent on the reliability level offered by the system, providers cannot simply aim at achieving their business objectives at the expense of the stable operation of services. As a result, a capacity management approach should consider these two often conflicting goals: attaining business objectives while providing reliability to the hosted services. In the next sections, we further elaborate on the problem considered and explicit our research contributions.

1.1 Problem Statement

We deal with one of the critical (from many) issues towards the derivation of benefits from IDCs — the ability direct the available IDC capacity in such a way as to best attain business objectives. In order to carry out this study, we investigate a service hosting scenario where a provider hosts multiple third-party transactional Internet services. With the ultimate goal of maximizing the provider's revenue with the SLA contract business, the system must strive to optimally configure itself in an autonomous fashion. We work on the system's ability to handle conflicts of capacity needs and tradeoffs in light of the provider's financial objective, instead of simply attaining the performance targets.

While the study of high level business objectives for an IT environment [10] and self-managing (or autonomous) systems [24, 41] have been extensively explored in the literature, to the best of our knowledge, the connection between these two areas has been poorly addressed in previous work. Accordingly, these two factors are intimately intertwined [10, 13], and this work proposes means through which to bridge them. In addition, we address some characteristics of the problem that have not been dealt with in previous studies, as described next.

The problem considered involves both business (i.e., pricing, service accounting, customer priority, etc.) and technological challenges (i.e., management mechanisms, performance prediction, etc.). To achieve the proposed goal, an IDC provider must

be able to simultaneously keep track of all business variables, variations in level of the workloads received, particular system characteristics and requirements of different applications, and, ultimately, to consider the provider's financial objective. Needless to say, the complexity of this task is enormous. As a result of this high degree of complexity, costs of management increases exponentially, systems becomes more fragile, hard to diagnose, to monitor, to predict, and, more importantly, it overwhelms human ability to keep pace with all these changing variables.

Behavioral and business characteristics of the problem make the approach even more challenging. The emergence of new customer demands pose unprecedented operational challenges to providers. First, in order to stay competitive, instead of agreeing on satisfying an average response time requirement, providers have been offering contracts that promise the satisfaction of tail distribution requirements, meaning that an upper bound on the probability of a given performance metric exceeding a given threshold is required. Moreover, since the satisfaction of a single performance metric requirement alone does not guarantee the quality of user experience, customers have been also demanding multiple metric satisfaction. For example, it is important for an e-business service to place user orders quickly, and, at the same time, to serve as many users as possible. In this case, in addition to establishing end-to-end response time requirements, this class of applications also demand from providers a guarantee on the achieved throughput. Last but not least, there has been a recent outburst of interest in service contracts in which customers pay only for actual use [28, 32, 44], instead of paying a fixed service rate. As a result, this forces providers to review the traditional SLA contracts and to propose more flexible service accounting plans.

Systems issues of the problem also exacerbate its complexity. Internet services usually receive workloads that may present great fluctuations over time, which change the service capacity needs during their operation dramatically. As a consequence, in order to best explore the available resources, an autonomous approach to re-distribute the capacity allocated to services as a response to changing demands is needed. To make matters more complicated, these workloads exhibit bursty behavior [25], meaning that they present unexpected short-lived surges of requests that may incur severe financial losses if the system is not prepared to handle them. Last, downtimes and poor system performance are responsible for significant losses to many types of business. Consequently, in addition to allow for adaptability, these systems must strive to constantly sense the environment and take reactive measures in real time.

The above facts have direct impact on the capacity management for Internet services. In the next section, we describe our contributions in dealing with those challenges.

1.2 Research Contributions

We deal with the problem of business-driven capacity management in IDCs at both business and systems levels.

At the business level, we propose a multi-level SLA specification which allows customers to pay only for actual use. We illustrate this point through a two-level contract which defines two different operation modes, namely normal and surge. Through this model, customers pay for extra capacity (than that normally required) only when needed. Moreover, since high-level IT metrics (i.e., response time, processing rate, etc.) is closer to the customer experience than individual resource utilization, this model is designed so that customers pay proportionally to the performance achieved by their services. We also propose a cost model based on penalties, incurred by the provider as a result of service level requirement violations, and rewards, paid by customers when their expectations, expressed as throughput requirements, subject to a response time guarantee, is exceeded. Last, we define the provider's business objective as the maximization of the net result from penalties and rewards during the execution of the services.

At the systems level, we present a queuing-based performance model that predicts the performance of the hosted Internet services, given the system characteristics, performance requirements, and current workload conditions of the hosted applications. The queuing model is proposed here because the characteristics of applications and their relationship with high-level systems metrics can be naturally expressed by mathematical formulas, which can be further linked to the business model in a clear and meaningful optimization problem formulation. Moreover, queuing models also allow our problem to scale with the number of services being hosted.

Due to the customer demands discussed in Section 1.1, the performance modeling of Internet services requires more complex approaches. For instance, the estimate of the probability distribution of response times, required for predicting the service capacity needs, is challenging when dealing with queuing models. This is because i) the exact results for this metric are only available for special types of queues, most of which do not realistically capture the characteristics of the services considered, and ii) some of the available results make the optimization problem hard to solve in real time. As a consequence, approximations are often needed. Thus, we propose and evaluate different approximations to express the probabilistic response time requirement, comparing the level of accuracy resulting from each of them in the context of our problem formulation, under two different service scheduling discipline to which Internet services may be subjected, namely, first come first served (FCFS) and processor sharing (PS).

High-level business objectives are mapped into IT directives through an optimization model, which, aiming at maximizing the IDC provider's business objective, combines the business and the systems models described. Since changes in the workloads, in the requirements, and in the contracts over time may alter the way the system must behave to achieve its high-level goals, in order for IDCs to respond quickly and, consequently, to adapt to changing capacity needs, we integrate our management model into an autonomous real-time adaptive framework.

Finally, through a numerical analysis, we assess the effectiveness of the proposed approach, with respect to business advantages to providers and system reliability to customers, by demonstrating that it increases the provider's SLA business potential while preserving the stable operation of the services.

1.3 Organization

The rest of this dissertation is organized as follows. Chapter 2 gives an overview of the environment considered. Chapter 3 discusses related work on IDC capacity management and compares the proposed model to existing approaches. Chapter 4 describes the sub-models that comprise the proposed capacity management model, including the cost model, the autonomous framework, the optimization, and the performance models. The effectiveness of the proposed approach is presented in Chapter 5, where discrete event simulation is used to analyze the operation of a multi-service IDC along with the resulting cost implications on the financial affairs of the provider. Finally, Chapter 6 offers our conclusions and discusses future work.

Chapter 2

Background on Internet Data Centers

Internet data centers consist of centralized shared environments which provide means for flexible resource provisioning to Internet services with the potential to maximizing hardware and software utilization as well as to derive benefit from economies of scale (i.e., the operation cost increases sub-linearly with the number of Internet services supported). IDCs are usually composed of large computing resource farms which support many different software components executing concurrently. Figure 2.1 presents an architecture diagram for an IDC where hundreds (or thousands) of processing and storage elements are switched through a high speed network. The infra-structure is flexibly provisioned to Internet services (see Section 2.1) which are visible in the Internet to external users and applications.

Similarly to Internet data centers, geographically distributed systems, known as Grids [16], are related environments in which applications are served by a wide network of resources connected to the Grid. However, as opposed to centralized environments, Grids are composed by a number of server nodes from independent sites, which share their resources in a best-effort or peer-to-peer fashion.

Before discussing the characteristics of IDCs, let us define some terminology. The workload subject to an IDC environment can be categorized into two main strands — batch workloads and business workloads. Batch workloads are typically composed by scientific and engineering applications which require basically CPU cycles. On the other hand, business workloads can be further divided into interactive or transactional workloads. Interactive workloads are those produced by a limited (usually small) population of users which regularly interacts with the system (i.e., a help desk service) whereas transactional workloads come from an infinite population of users or other Internet services that make requests to retrieve or to update information in the system. As batch workloads have been largely studied in light of Grid computing [16], our work focuses on transactional business workloads, by considering a scenario where a provider

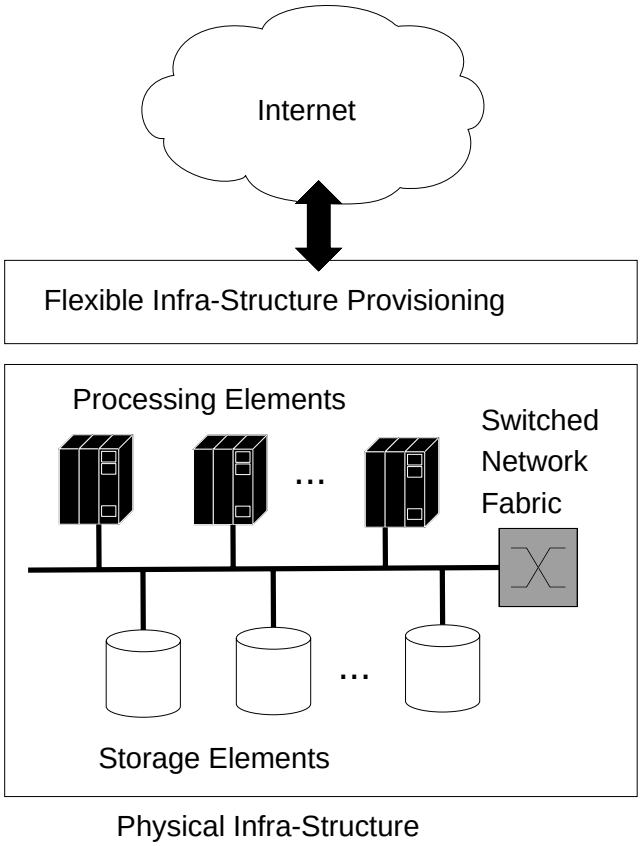


Figure 2.1: Internet data center architecture diagram.

hosts multiple third-party transactional Internet services in a shared IDC.

A resource is any shared system component which is multiplexed by the system among several Internet services being hosted by the IDC. As an example of resource, we can cite CPU time, memory space, storage, I/O, network bandwidth, or even a whole server computing power.

Typical Internet services are usually composed of different transaction types, subjected to different workloads, with different demands on the resources and executed by independent software components. We denote these components as *application classes*. Under the assumption that classes are independent, they are analyzed here as independent applications.

In the next section, virtualization, a core mechanism, which differs IDCs from traditional computing systems [44, 47], is described.

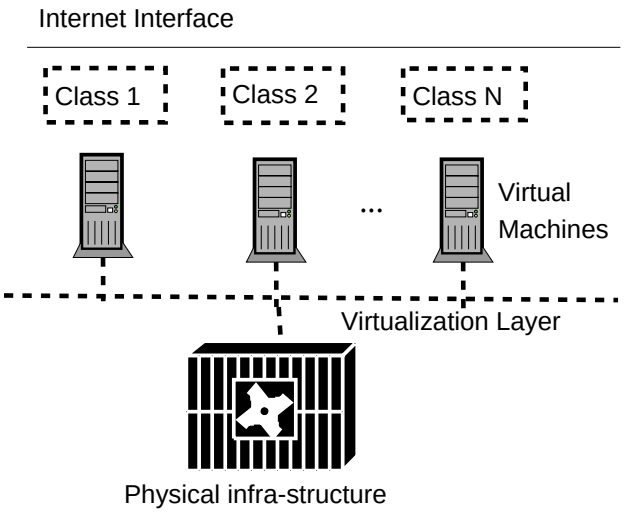


Figure 2.2: Internet data center hosting platform.

2.1 Virtualization

A key feature of IDCs is the ability to provide performance isolation by preventing direct contention for resources between different services. Accordingly, a virtualization scheme [9, 18] (or fair sharing policy) is employed for this purpose.

Virtualization is an abstraction of computing resources, which creates on top of the physical structure virtual (logical) resources which represent and conceal their physical "underlyings" [18, 43]. The term virtualization has different meanings in computer science and serves different purposes. For instance, the creation of a logical architecture from a collection of physical resources, thereby creating new functionalities [43] (i.e., virtual memory, virtual machines, etc). In this work, we concentrate on describing the use of virtualization in the context to which our model applies.

In IDCs, virtualization is responsible for partitioning the physical resources (i.e., processing, storage and communication resources) into multiple isolated virtual ones, each running at a fraction of its corresponding physical resource capacity. Hence, instead of using the physical resources directly, the hosted Internet services demand service from a pool of virtual resources, created and maintained by an intervening *virtualization layer*. On using this feature, systems are able to build pools of virtual servers, virtual operating systems, virtual I/O, virtual networks, and virtual applications.

The Internet service hosting platform through IDCs, considered in this work, is depicted in Figure 2.2. The virtualization layer creates N isolated virtual machines (VM) (also known as resource containers or application environments [9, 42]) on top of the physical infra-structure. Each VM is composed of a set of virtual instances of

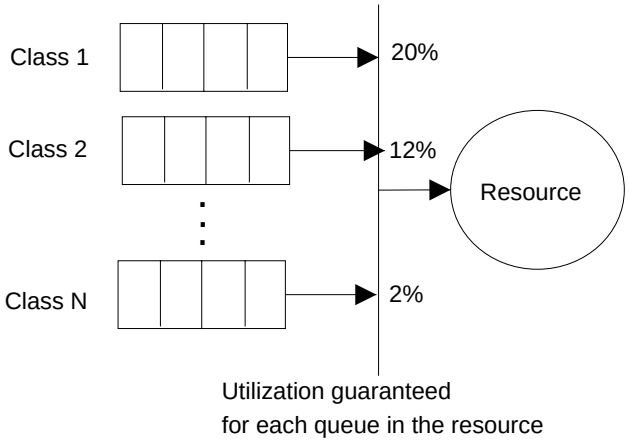


Figure 2.3: A resource serving multiple classes through virtualization.

each of the IDC's resources and is dedicated to serving a single application class only. This hosting model isolates classes one from another, each using the virtual machine as if it were a dedicated server, working at a fraction of the total (physical) capacity.

In this fashion, instead of simply relying on the operating system's multiprogramming feature (i.e., the process scheduler) for sharing computing resources among applications, IDCs employ a virtualization scheme that allows the physical resources to be proportioned to application classes, each receiving at least as much capacity as has been assigned to it, regardless of the load imposed by other classes. That is to say that Internet services do not share the resources through a common queue. There is actually one queue for each application class at each resource wherein each queue receives a fraction of the total resource time. For example, if application class 1 receives 20% capacity assignment and class 2 the remaining 80% for a given resource (i.e., a CPU), the resource guarantees 20% of the total resource utilization to serve application class 1, and 80% to 2. As a result, it enables IDCs to flexibly contract or expand the resource capacities assigned to application classes and to provide service differentiation. Figure 2.3 illustrates this concept by showing N different application classes sharing a resource, each using a different queue, associated to a utilization guarantee in the resource. Virtualization can be implemented at either operating system or hardware levels, and its module usually offers APIs (Application Programing Interface) that allows the operator to change its configuration at execution time.

Hence, we define the capacity allocation decision as the determination of the fractions of server capacity each VM i ($i = 1, .., N$) obtains from the corresponding physical server. Accordingly, the resource allocation decisions may be made based on capacity constraints [47], policies specified by operators [35] or business directives [34].

Last, we assume that the VMs employ an admission control scheme [31] that re-

jects requests for various purposes. For example, VMs may drop some requests to avoid service instability due to capacity limitations or to guarantee that performance requirements are met.

In order to provide an effective management of virtual resources, it is required a thorough understanding of performance issues, and business implications of allocation decisions. In Chapter 3, related work on capacity management of Internet data centers is presented. Moreover, the complexity of managing the infra-structure described, and the rate at which workload changes occur, make it impossible for humans to manually handle such environments. Thus, a model for providing configuration to IDCs autonomously in light of the provider's business objective is proposed in Chapter 4.

Chapter 3

Related Work on Capacity Management

In this Chapter, we discuss previous work upon which our model builds and related work that have some degree of intersection with ours. We start by discussing business proposals to the problem of computing-based capacity outsourcing. In sequence, we cite some references that discuss the impact of this new technological trend on the management of services. Moving on to capacity management issues, we describe the most popular approaches to the problem, namely, trace-based and optimization-based models. The complexity of modern systems, which overwhelms human capacity to keep pace with the rate of changes, resulted in several proposals for autonomous adaptation. In this work, we make use of this feature, and relevant references on this topic are presented in this Chapter. We also address performance modeling issues for general-purpose shared environments, which are categorized into two main strands, namely, model-based and model-free approaches. Finally, studies that propose an optimization model for satisfying performance requirements of applications are presented. In each topic, we discuss the relation between the cited works and the approach proposed in this dissertation.

The increasing interest in computer-based capacity outsourcing has brought it to the attention of the IT industry. This trend resulted in several proposals to maximize the capitalization on the strategic advantages that shared data centers may deliver. For instance, Ross and Westerman [34] study the impact of the maturity of both business and technological strands on the revenue of providers. In [13], Buco et al present evidences, showing that business-oriented management for shared environments, as proposed in this work, reduces the financial risks associated with service level violations. In the same direction, Appleby et al. [7] and Santos et al. [35] discuss the benefits

of managing resources with respect to pre-established goals instead of simply following application procedures. Rappa [32] proposes service contracts for shared data centers similar to the ones for public utilities (gas, electricity, phone, etc.), wherein customers pay only for actual use and shows the potential of these agreements for future computing services. These studies served as motivation for our working hypothesis, which states that orchestrating resources in light of business goals can be more profitable to providers than simply providing the smooth running of the hosted services.

In another aspect, the transition from traditional computing to business-oriented computing in shared environments, including the impact of IDC features in the application development, is discussed. Wilkes, Mogul, and Suermondt [44] present the main technological challenges and the roadmap to the maturity of these environments, aiming at adapting applications to this new paradigm. Graupner et al. [18] discuss the impact of virtualization on the management of shared environments. Appleby et al. [6] demonstrate a possible implementation of resource management for the computing power on demand in data centers. As described in Chapter 2, our model builds upon these features, specially virtualizaiton, which simplifies the problem, allowing us to completely abstract the physical topology of the infra-structure.

With the popularity of shared environments and computing power outsourcing, resource management became a essential capability to the success of operation. Among the first attempts to accomplish this task, trace models has been widely applied. Through a pre-production workload characterization, capacity needs and workload cycles can be detected, and, therefore, the infra-structure can be properly sized. One example of workload characterization on CPU utilization which serves as basis for such approaches was conducted by Abrahao and Zhang [4]. The Quartermaster [37], which consists of a set of tools and technologies wherein capacity needs are determined based on the application workload life-cycle and its influence on the system, is the best known example. Another application of the approach is given by Andrzejak, Arllit and Rolia [5] where CPU traces are used as basis for an off-line workload placement model that aims at better deploying applications in the available processors in a data center. However, unlike our approach, trace-driven capacity management depends heavily on the characteristics of the workload sample collected for analysis and does not consider the business issues involved.

Resource management optimization models have been the most popular approach for exploring cost-effectiveness in shared environments. These models usually differ in goal. For example, Zhu et al. [47] optimize the resource allocation in order to minimize network traffic and resource underutilization. Zhang et al. [46] propose an optimization model based on queuing-theoretical formulas to determine the minimum

number of servers that should be allocated at each tier of a single Internet Service in such a way that a response time requirement is satisfied. Urgaonkar, Shenoy and Roscoe [40] explore capacity overbooking for optimally placing a set of third-party applications in different servers of a shared environment. Although our model also uses an optimization model for capacity management, our problem formulation serves an additional purpose: to map high-level business objectives into system decisions.

The manual management of resources operated by humans has become increasingly unsuitable for modern computing systems due to the rapidly increasing size of the systems, the heterogeneity and the increased dependency of their components, the complexity of modern environments, and the rapid rate of state changes. In light of these facts, autonomic (or self-managing) approaches appeared as a solution for adequately administer the complexity of such systems. An introductory article about self-managing challenges is presented by Herrman, Muhl and Geihs [20]. In this work, the authors discuss to which extent automation brings a solution, or simply adds complexity to systems. In the automation direction, previous work considered autonomic closed control loops using different techniques for a number of purposes. The interested reader is encouraged to refer to Gosh and Herman's book [17] for a thorough treatment of the topic. In an attempt to provide the system with the ability to react quickly to changing business priorities, we incorporate our capacity management model into an autonomic framework.

The performance study of applications has been primarily divided into two main strands: model-based and model-free approaches. Model-based approaches are the ones in which a deeper knowledge of the application is required from the operator. However, since they are composed by analytic mathematical formulas, they are usually very efficient to compute. An example of a model-based approach consisting of a queuing model that periodically reconfigures the parameters of a web server so as to increase its performance is presented by Menascé and Bennani [27]. Claiming that these models are application-specific, demand more information about the applications than an operator has to be concerned, and sometimes fail at capturing the actual behavior of applications, the proponents of model-free approaches started to receive attention. These approaches aim at mapping the application states into observable performance implications and, then, to use this collected information as a reference for future courses of actions. They also claim that model-free approaches, as opposed to model-based ones, are able to capture anomalous behavior, such as application bugs, virus contamination interference, and to deal with complex applications, whose behavior is difficult to characterize. To illustrate this point, Walsh et al. [42] considered mapping the workload levels of a data center to their observed influence on

the system. Aiming at maximizing the sum of utility functions, they rely on a state-space based on the past behavior so as to provide information for future capacity allocation decisions. Menascé and Bennani [12] further revisited the latter framework, proposing a combinatorial search technique together with queuing models to provide scalability to the table-driven approach. Another important reference is Cohen et al., which propose a model-free approach based on Bayesian networks to study SLA violations. Besides these two main strands, some authors advocate the control theoretical approach [14, 23]. The reason to use a model-based in our capacity management approach is two-fold. First, it allows our model to scale with the number of applications being hosted, and, second, since it is based on mathematical expressions, it can be naturally combined with the expressions of a business model.

In the bridge between business goals and systems decision lies the main contribution of our work [2, 3]. While previous work have proposed the definition of business objectives [10] for IT management, and the performance management for shared environments [24, 41], the mapping between the two has been poorly addressed. Bartolini and Sallé [10], along with Buco et al. [13], propose a powerful framework for expressing business goals and to support IT decision takers. Through these models, it is possible to specify different objectives, from different perspectives (customer, provider, investor, etc.), and the IT metrics that would impact the achievement of such objectives. Through their model, it is also possible to predict the business impact of assigning different priorities to alternative courses of actions. However, unlike our focus in this dissertation, these studies completely abstract the actual system, capacity constraints, and system costs of attaining an objective.

Finally, Liu, Squillante, and Wolf [24] as well as Villela, Pradhan, and Rubenstein [41] deal with the problem of resource allocation for best accommodating e-commerce applications in a service provider. These studies are the closest to ours in that they also propose an optimization model and deal with performance metric and capacity constraints. However, they do not consider managing a platform with respect to business objectives. Also, they assume static workload, abstract the whole servers as a discrete unit of capacity allocation, and work with response time requirements alone.

In the next Chapter, we present our business-driven capacity model where its characteristics are further elaborated.

Chapter 4

Capacity Management Model Composition

This chapter describes the autonomous business-driven capacity management model proposed in this dissertation. It starts by defining the multi-level SLA contract that allows for per-use service accounting and the business model based on penalties and rewards in Section 4.1. In sequence, the autonomic framework that provides the system with the ability to adapt itself as a response to changes is presented in Section 4.2. The main model parameters and assumptions are described in Section 4.3.1, and the optimization model formulation is presented in Section 4.3.2. Finally, Section 4.4 describes the performance model alternative approximations.

4.1 SLA-Driven Business Model

This section describes the business model based on the multi-level SLA contract that allows for per-use service accounting through a penalty/reward scheme.

The service contracted by customers must meet certain performance expectations which the parties agree upon in the SLA contracts. This work focuses on the requirements of throughput and response time. In addition to agreeing on the requirements in the SLA contracts, the provider also establishes a service rate to charge customers proportional to the strictness of their service level requirements. For instance, customers who run critical businesses are willing to pay more so as to obtain high throughput and/or short response times whereas other customers sign up for service with loose requirements but lower costs.

The Internet offers plenty of examples of services which usually receive low to moderate load, but occasionally receive an exceptionally large surge of requests. Common

examples are on-line news services which expect a predictable number of users during normal operation but, whenever certain special events occur, a sudden surge of clients overload the site, changing the capacity needs dramatically [8]. Due to the highly dynamic nature of Internet workloads, we propose contracts with multiple levels of requirements. We illustrate this point with a two-level contract that allows for per-use service accounting and defines two different operation modes, namely normal and surge.

In the normal operation mode, customers contract the service level which satisfies their needs for the majority of operation time whereas in the surge operation mode, a higher service level limit is established, up to which the provider has an incentive to assign extra capacity to services so as to accommodate occasional workload peaks. Traditional contracts with a single performance target would require customers whose workload presents high peak-to-mean ratio to pay for the service level needed to satisfy both the average and the peak of demand during the entire operation, even though only part of the capacity the customer pays is actually utilized most of the time. Therefore, from the business standpoint, the two-level SLA approach can be advantageous both to customers who pay for extra capacity only when needed, and to providers who are able to offer more attractive service plans by operating with more flexibility.

There are several ways of measuring the service level provided to a customer. Instead of accounting the service by measuring the utilization of each individual computing resource (i.e., CPU), the proposed model works with a metric that has a direct meaning to customers: the performance level achieved by their services. Thus, the SLA performance requirements quantify the VM's ability to process transactions, provided the quality of the processed transaction response time satisfies a given requirement. Accordingly, it is considered the tail distribution response time requirement, which states that the response time of the transactions from class i must not exceed a given threshold R_i^{SLA} for more than $\alpha_i \times 100\%$ of the time, or equivalently, $P(R_i > R_i^{SLA}) \leq \alpha_i$, where R_i is the response time of a single transaction of class i.

The actual throughput of a class may be smaller than its arrival rate of requests. That is because, due to capacity limitations, some of the requests may be rejected. In addition, some of the processed transactions may violate the response time requirement in which case they are not counted (for SLA accounting purposes) as processed. Consequently, we refer to the the actual processing rate in which transactions are performed within the response time requirement as the *valid throughput*. In order to distinguish these metrics, we consider λ_i as the arrival rate, X_i as the actual throughput, and μ_i as the valid throughput of class i.

In order to determine the valid throughput, we define Q_i as the relative frequency of

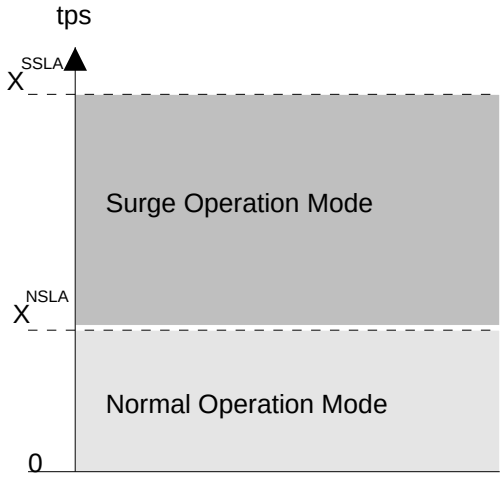


Figure 4.1: Example of an arbitrary valid throughput scale with the ranges of normal and surge operation modes.

transactions with response times below R_i^{SLA}, measured at the end of a time interval. Thus, if the requirement has been violated, that is $Q_i < (1 - \alpha_i)$, then $\mu_i < X_i$, and the valid throughput is given by the transactions processed within the time limit plus the tolerance allowed: $\mu_i = Q_i X_i + \alpha_i(Q_i X_i) = (1 + \alpha_i)(Q_i X_i)$. Otherwise, when $Q_i \geq (1 - \alpha_i)$, all transactions processed during that interval performed in accordance to the response time requirement and, thus, the valid throughput equals the actual throughput, $\mu_i = X_i$.

Figure 4.1 displays an example of an arbitrary scale that shows the range of the normal and the surge operation modes. In light of the above reasoning, the proposed SLA contracts contain performance targets for the normal operation mode, under NSLA requirements, and for the surge operation mode, under SSLA requirements. The NSLA requirement states that the valid throughput of application class i should be at least X_i^{NSLA} transactions per second, which is the upper limit to the normal operation mode. Nonetheless, violations of the NSLA requirements may occur, in which case the provider agrees to refund part of the service charge to customers, which is proportional to the difference between the NSLA throughput requirement and the saturated valid throughput, provided this difference has been caused by capacity limitations. Conversely, the SSLA target is established in order to prevent missing service or unavailability in case of occasional workload peaks. Thus, the parties establish a limit on the throughput, X_i^{SSLA} ($X_i^{SSLA} \geq X_i^{NSLA}$), which is the upper limit to the surge operation mode, up to which the owner of class i agrees to pay a reward to the provider whenever the valid throughput over the interval exceeds X_i^{NSLA}. Accordingly, rewards are proportional to the extra valid throughput achieved.

Table 4.1: Business Model Symbols.

Symbol	Description
R_i	measured response time (in seconds) of a single transaction of class i.
R_i^{SLA}	response time threshold (in seconds) for class i.
α_i	limit on probability of response time being greater than R_i^{SLA} for class i.
λ_i	arrival rate (in requests per second) of class i
X_i	actual throughput of class i (in tps)
μ_i	valid throughput of class i (in tps)
X_i^{NSLA}	valid throughput requirement (in tps) for class i in normal operation mode.
X_i^{SSLA}	valid throughput upper limit (in tps) for class i in surge operation mode.
Q_i	Relative frequency of transactions with response times below R_i^{SLA}.

The achieved performance of applications is computed at the end of periodic intervals (discussed in Section 4.2) so that the net result from penalties and rewards can be determined. Thus, given the framework described, we define the provider's business objective as the provision of capacity to VMs so as to maximize the net result from penalties and rewards during its operation.

It should be emphasized that this approach is not restricted to only two SLA targets. Thus, one may define multiple levels which specify multiple performance levels, with different costs, so as to account for different demands.

Finally, Table 4.1 summarizes the symbols introduced in this section and may serve as a reference to the reader.

4.1.1 Penalty and Reward Accounting

This section describes how penalties and rewards are computed and how the payoff for the execution of services is calculated at the end of each periodic interval.

Upon defining the response time tail distribution requirement as a QoS condition, the NSLA saturation is considered here as the VM's inability to achieve the valid throughput X_i^{NSLA} requirement of class i. Consider Y_i to be the penalty incurred by

Table 4.2: Values of penalty and reward.

	Symbol	Condition	Value
Penalty	Y_i	$\mu_i < X_i^{NSLA}$	$min\{X_i^{NSLA}, \lambda_i\} - \mu_i$
Reward	Z_i	$\mu_i \geq X_i^{NSLA}$	$min\{X_i^{SSLA}, \mu_i\} - X_i^{NSLA}$

the provider due to violating the performance requirement of class i. As mentioned in Section 4.1, penalties are proportional to the magnitude of the difference between the NSLA throughput requirement and the valid throughput. However, the throughput may have been smaller than the requirement due to capacity limitations or due to an arrival rate smaller than X_i^{NSLA}. Since the provider should be penalized only for the former case, a penalty is incurred whenever $\mu_i < X_i^{NSLA}$ and $\mu_i < \lambda_i$. In this case, the penalty is $Y_i = \lambda_i - \mu_i$. However, since the NSLA target requires the provider to process at least X_i^{NSLA} transactions over the time interval, if $\mu_i < X_i^{NSLA}$ and $\lambda_i \geq X_i^{NSLA}$, the incurred penalty should be $Y_i = X_i^{NSLA} - \mu_i$. Thus, the penalty cases just described can be expressed as $Y_i = min\{X_i^{NSLA}, \lambda_i\} - \mu_i$.

Conversely, if the valid throughput of an application class has exceeded the throughput NSLA requirement, $\mu_i \geq X_i^{NSLA}$, the provider is able to capitalize rewards proportionally to the magnitude of the difference between the extra valid throughput and the NSLA throughput requirement. However, the magnitude of the reward cannot exceed the limit X_i^{SSLA}. Let us denote Z_i to be the magnitude of the reward paid by class i owner to express the reward case as $Z_i = min\{X_i^{SSLA}, \mu_i\} - X_i^{NSLA}$.

The above conditions determine the values of the SLA penalties and rewards computed at the end of a time interval. The penalty or reward for any other possible condition is 0. All penalty and reward cases are summarized in Table 4.2. Last, penalties and rewards are weighted in the main model with constants c_i, which is the penalty cost for a unit of throughput NSLA saturation, and π_i, which is the reward price for a unit of extra valid throughput above X_i^{NSLA} for class i.

4.2 Autonomic Framework

In order to provide the system with the ability to adapt itself in response to changes, we propose the closed control loop presented in Figure 4.2. The heart of the model is the *capacity manager* component which, given the expected workload of each VM, the SLA requirements, and the system characteristics of the classes, reconfigures the IDC with the goal of maximizing the provider's business objective. The capacity manager

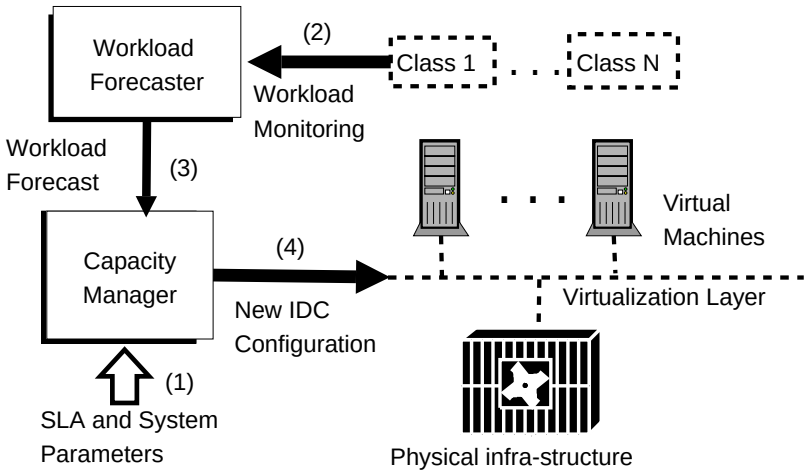


Figure 4.2: Closed control loop.

component is based on an optimization model (discussed in Section 4.3.2) which combines the cost and the system strands of the problem. Therefore, the capacity manager component is configured with SLA and system parameters of each application class (discussed in Section 4.1 and summarized in Table 4.3 of Section 4.3.1) (1). These values are updated whenever contract changes occur (i.e., a new application class is introduced or removed from the data center, or the requirements change).

The *workload forecaster* component is responsible for constantly monitoring the intensity of the workload of each class over time (2). It contains a local storage facility (i.e., a database) to store the past behavior of the workload so that the logged data can be used for periodic analysis. Accordingly, it uses one of the existing workload forecasting methods [1] to predict, based on past observed behavior, the workload that each class is likely to receive in the near future. Consequently, the forecast is also used as input by the capacity manager (3).

After determining a capacity allocation decision, the capacity manager component sends the new IDC configuration to the system (4), adjusting the virtual resource mappings in the virtualization layer into new values at the physical infra-structure and the admission control parameters for each VM.

We refer to the interval between consecutive controller interventions as the *controller interval*. The frequency of adaptations and the durations of these intervals may be fixed or variable depending on the characteristics of the system and its services. First, adaptation can be proactive, when the workloads of applications have been previously characterized and one can anticipate when significant changes are likely to occur. In this case, the capacity manager invocations can occur at periodic intervals of time whose limits correspond to times in which the workload of services is likely to change.

On the other hand, unexpected events may occur, but it is not always possible to predict them (i.e., a stock crash that results in an increase of network access). Thus, reactive measures are also needed. For example, services whose workload present a predicted workload behavior can be programmed to trigger extra adaptations whenever unexpected changes in demand occur. Second, limitations on the time taken for the system to adapt itself define the minimum length of the interval.

Finally, we consider that the net result from penalties and rewards for the execution of the services is computed for each controller interval, in the end of the corresponding time period.

4.3 Capacity Management Model

This section describes the capacity management model for Internet services, its parameters and assumptions (Section 4.3.1), and the optimization model (Section 4.3.2).

4.3.1 Main Model Parameters and Assumptions

The main model parameters, which are used as inputs to the Capacity Manager component, are presented in Table 4.3. Some of the parameters are extracted directly from the SLA contracts while others are system parameters and estimates made by the workload forecaster component.

The SLA parameters can be understood in light of the discussion presented in Section 4.1. The system parameters consist of: 1) the number N of VMs maintained by the virtualization layer, thereby the number of application classes hosted by the provider, 2) the maximum utilization υ_i the provider allows VM i to reach, and 3) the average service time of transactions from class i on the physical server (when executed with its total capacity).

Parameter υ_i is introduced because both the mean and the variance of response times can increase without limit when the utilization approaches 100%. Thus, in order to maintain a certain level of stability in the VMs, a fractional upper limit υ_i ($0 \leq \upsilon_i < 1$) is specified, up to which the utilization of VM i is planned.

We assume that transactions from the same application class are statistically indistinguishable and, thus have the same average service time on the physical server. Therefore, $E[S_i]$ is used here as a model parameter to indicate the mean service time of class i, which can be approximated by the arithmetic average of the observed values of S_i, measured in a pre-production execution of the applications on the physical server (with its total capacity).

Table 4.3: Model parameters.

SLA parameters

Symbol	Description
X_i^{NSLA}	valid throughput requirement (in tps) for class i in normal operation mode.
X_i^{SSLA}	valid throughput upper limit (in tps) for class i in surge operation mode.
R_i^{SLA}	response time threshold (in seconds) for class i.
α_i	limit on probability of response time being greater than R_i^{SLA} for class i.
c_i	penalty cost for a unit of throughput NSLA saturation for class i.
π_i	reward price for a unit of extra valid throughput above X_i^{NSLA} for class i.

System parameters

N	number of application classes, and thus number of VMs.
υ_i	maximum utilization planned for VM i.
$E[S_i]$	average service time (in seconds) of class i transactions on the physical server.

Workload Forecaster's estimates

λ_i^*	predicted arrival rate (in requests per second) of class i for next controller interval.
$P\{Z_i\}$	probability of capitalizing rewards from class i during next controller interval.

Since each VM receives a guaranteed fraction of time from the physical server, taking f_i as the fraction of the server assigned to the VM i, we approximate the average service time on the VM as $E[S_i]/f_i$. Accordingly, the capacity allocation decision is the determination of the capacity fractions (i.e., $f_i, i = 1..N$) to be assigned to each VM i. Therefore, f_i is the main decision variable of the proposed problem.

The last two rows of Table 4.3 present the outputs of the workload forecaster component. As mentioned in Section 4.2, the capacity manager component receives from the workload forecaster component an estimate λ_i^* of the arrival rate of requests during the next controller interval for each application class i.

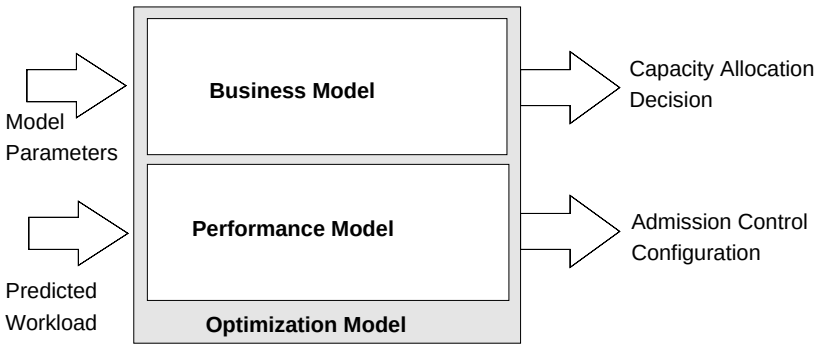


Figure 4.3: Capacity Manager model scheme, including its inputs and outputs.

When considering fixed controller intervals, events that are significantly shorter than the interval could mislead the manager. For example, a surge of requests with short duration can cause the capacity manager to seize the capacity needed by other classes to meet their NSLA requirements so as to provide extra capacity to a more profitable one. Clearly, this would incur penalties throughout the controller interval whereas the reward would be capitalized during a tiny fraction of the time. Thus, in order to minimize this undesired effect, the workload forecaster component may also produce the estimated probability of a class receiving a surge of requests, that is, an arrival rate greater than X_i^{NSLA}, during the next controller interval. Thus, parameter $P\{Z_i\}$ indicates the certainty level to bet on reward capitalization for VM i. If the system is allowed to adapt to workload changes, this parameter can be bypassed by setting it to 1.

4.3.2 Optimization Model Formulation

In this section, we present the optimization model which combines the business model with a performance model (discussed in Section 4.4) and determines, in light of application characteristics, SLA specifications and predicted workload of application classes, a capacity allocation decision and admission control parameters that maximize the provider's business objective. This model is the core of the Capacity Manager component, whose scheme, including its parameters and outputs, is presented in Figure 4.3. This component is further included in the autonomous framework (presented in Section 4.2).

The model objective function expresses the sum over all application classes of the expected payoff for their execution as follows:

$$Maximize \sum_{i=1}^{N} -c_i Y_i + \pi_i Z_i P\{Z_i\} \tag{4.1}$$

$$
\begin{array}{ll}
max & \sum_{i=1}^{N} -c_i Y_i + \pi_i Z_i P\{Z_i\} \\
s.t. & \\
& Y_i \geq min(X_i^{NSLA}, \lambda_i^*) - X_i \qquad (a) \\
& Y_i \geq 0 \qquad (b) \\
& Z_i = \delta_i (min(X_i^{SSLA}, X_i) - X_i^{NSLA}) \qquad (c) \\
& Z_i \geq 0, \quad \delta_i \in \{0,1\} \qquad (d) \\
& \lambda_i^* = \lambda_i^{acc} + \lambda_i^{rej} \qquad (e) \\
& X_i = \lambda_i^{acc} \qquad (f) \\
& P(R_i \geq R_i^{SLA}) \leq \alpha_i \qquad (g) \\
& \rho_i = \lambda_i^{acc} / \lambda_i^{sat} \leq \upsilon_i \qquad (h) \\
& \sum_i^N f_i \leq 1 \qquad (i) \\
& 0 \leq f_i \leq 1, \quad \rho_i \geq 0 \qquad (j) \\
& X_i, \lambda_i^{acc}, \lambda_i^{rej}, \lambda_i^{sat} \geq 0 \qquad (k)
\end{array}
$$

Figure 4.4: Optimization model.

The whole optimization model is depicted in Figure 4.4 and in order to refer to the model constraints, they are indexed using lowercase letters (a) to (k).

Constraints (a) to (d) express the cases shown in Table 4.2, replacing the arrival rate λ_i and the valid throughput μ_i of the past controller interval by the estimate λ_i^* and the actual throughput X_i (since we aim at making $\mu_i = X_i$) respectively. Notice that, since $Y_i \geq 0$ is stated in constraint (b), Y_i assumes 0 in cases of reward (i.e., when $X_i > X_i^{NSLA}$). In addition, in constraints (c) and (d), an artificial binary variable δ_i is introduced as part of the expression for Z_i. Thus, δ_i is forced to 0 when $X_i < X_i^{NSLA}$, making $Z_i = 0$ in case of penalties. Otherwise, δ_i assumes the value 1.

Due to capacity limitations, the VMs impose a limit on the number of requests that can be processed. However, the arrival rate can be greater than the maximum achievable processing rate in which case some of the requests must be dropped. Therefore, we assume that the VMs employ one of the existing admission control policies [31] which splits the arrival rate a VM receives into $\lambda_i = \lambda_i^{acc} + \lambda_i^{rej}$, where λ_i^{acc} is the rate at which VM i accepts requests, and λ_i^{rej} is the rate at which VM i rejects requests. This is expressed in the optimization model by constraint (e), using λ_i^* as the expected arrival rate.

Constraint (f) expresses the job flow balance condition [26] which states that all

accepted requests are actually processed by the VM, thereby assuring that no accepted request is lost.

If the reader will recall, the valid throughput of a VM is subject to the response time requirement satisfaction. Therefore, this is expressed as model constraint (g). Since the expected response time is a function of the VM throughput and the fraction of capacity it receives, $R_i = f(\lambda_i^{acc}, f_i)$, constraint (g) forces the VMs to accept only the transactions that would satisfy the response time requirement for any given value of f_i, with the objective of making $\mu_i = X_i$ at the end of the controller interval.

Notice that α_i expresses a tradeoff between the throughput and the quality of the processed transactions. Indeed, a smaller α_i results in lower throughput but guarantees a higher percentage of transactions with response times shorter than the threshold, whereas a large α_i increases the processing rate of transactions at the expense of their response times.

The utilization ρ_i of VM i is defined by constraint (h) as the ratio of λ_i^{acc} and the smallest arrival rate λ_i^{sat} at which VM i becomes saturated (see Section 4.4). It is also important to emphasize that, similarly to the expected response time, ρ_i is a function of both the acceptance rate of request and the fraction of capacity the VM receives, $\rho_i = g(\lambda_i^{acc}, f_i)$. Moreover, constraint (h) guarantees the stability condition by limiting ρ_i to $\upsilon_i \times 100\%$.

A direct consequence of the above constraints is that λ_i^{acc} is limited by constraints (e), (g) and (h), by the expected arrival rate, the response time satisfaction, and the maximum allowed utilization respectively.

The capacity allocation constraint is expressed in (i). This sum limits the capacity percentages assigned to the VMs to 100%.

Last, constraints (j) and (k) delimit the domain of the other variables.

It is important to emphasize how this model combines the business and systems models, by expressing their equations as its constraints. Accordingly, constraints (a) to (d) correspond to the business model whereas constraints (e) to (h) correspond to the systems model.

The performance model is introduced in this model, for predicting the values of the performance-related variables λ_i^{sat} and $P(R_i \geq R_i^{SLA})$, by substituting constraints (g) and (h) by different approximations based on queuing models, which result in different degrees of accuracy. With this objective, we discuss these methods in the next section.

This problem can be expressed as a non-linear optimization model, since constraint (g) results in a non-linear expression when we substitute it for the performance model approximations (see Section 4.4), and can be solved efficiently by one of the available non-linear solvers [38] (see Chapter 5). This characteristic is relevant in our discussion,

since it provides the model with the ability to produce adaptive decisions in real time.

4.4 Estimating the Performance Metrics

This section presents an analytical queuing model to calculate the values of the performance metrics used in the optimization model presented in Section 4.3.2. As discussed earlier, the queuing model is able to map the characteristics of applications, the SLA requirements, the current workload conditions, and an instance of IDC configuration into values of performance metrics specified in the SLA contracts.

The most challenging part of the performance prediction for the model considered is the estimation of the probability distribution of response times. This is because this exact probability distribution of response time can only be estimated for some types of queues [21] which do not capture all types of the services considered, and some of the available expressions are complex and, therefore, limit the optimization model real time computation [41]. For example, an expression for estimating this metric for queues that receive requests following a Poisson process and serve requested in a FCFS fashion with exponentially distributed service times is available, whereas an equivalent result for the queues with PS scheduling discipline is unknown [45]. Thus, we propose and compare different approximations for this purpose.

Since we are dealing with a general class of transactional services, they may present a large diversity of traffic patterns. However, for the sake of simplicity, we make the following simplifying [1] assumptions and leave the treatment of specific characteristics of different applications as future work:

- Since there is considerable evidence that the arrival process imposed by users follows Poisson [24, 41, 30], we assume Poisson request arrivals. We also assume that the service times of application classes are exponentially distributed.

- We consider two types of queues for modeling transactional service centers, namely M/M/1 and M/G/1 with processor sharing (PS). The former works under FCFS scheduling discipline, which may be more accurate for modeling, for example, transactional servers with write operations, whereas the latter serves customers in a round robbin fashion with a very small quantum, which closely mimics the behavior of the multiprogramming feature of modern operating sys-

[1] Again, these assumptions are made for the sake of simplicity of treatment. Therefore, they can be removed without invalidating the approach. However, whether similar results can be obtained when these assumptions are relaxed is out of the scope of this work.

tems [36]. Both queues have been frequently considered as reasonable abstractions for transactional service centers [41, 26].

Let us start by finding the saturating throughput of VM i. Applying the Utilization Law [26] results in the following expression for the utilization: $\rho_i = \frac{E[S_i]}{f_i} \times X_i$. Thus, the saturating throughput is determined to be:

$$X_i = \frac{\rho_i f_i}{E[S_i]} \leq \frac{f_i}{E[S_i]} = \lambda_i^{sat}. \tag{4.2}$$

Notice that by substituting λ_i^{sat} using Equation 4.2 in constraint (h) of Figure 4.4, it imposes the following limitation on the rate of acceptance of requests: $\lambda_i^{acc} \leq \upsilon_i f_i / E[S_i]$.

Let us now turn our attention to approximations to compute the tail probability distribution of the response time. We consider Markov's Inequality [29] as a first approximation, requiring only the expected response time of the classes, which is computed using the same expression for both M/M/1 and M/G/1 (PS) queues [21]:

$$E[R_i] = \frac{E[S_i]/f_i}{1 - \rho_i}. \tag{4.3}$$

Thus, using the mean response time and guaranteeing the stability condition (constraint (h) of Figure 4.4), it is possible to approximate the tail distribution response time requirement using Markov's Inequality as:

$$P(R_i \geq R_i^{SLA}) \leq \frac{E[R_i]}{R_i^{SLA}} = \frac{1}{R_i^{SLA}} \frac{E[S_i]}{f_i - \lambda_i^{acc} E[S_i]} \leq \alpha_i \tag{4.4}$$

The advantage of using Markov's Inequality is that it only requires the average response time and can be applied to both M/M/1 and M/G/1 (PS) interchangeably. Nevertheless, Equation 4.4 often provides a loose bound and, therefore, it could result in overly strict requirements of response time.

Often, it is possible to improve upon Markov's inequality to obtain a tighter bound by using Chebyshev's Inequality [29]. However, this result depends on both the average and the variance of the response time. The latter metric is queue dependent and, for the M/M/1 case, is given by [21]:

$$Var[R_i] = \frac{(E[S_i]/f_i)^2}{(1 - \rho_i)^2} \tag{4.5}$$

For M/G/1 (PS) queues, the computation of the response time variance uses an integration term which makes the optimization problem hard to solve. However, it can

be shown that a tight upper bound of the variance for a PS queue [45], especially for large jobs can be obtained by:

$$Var[R_i] \leq \frac{\rho_i(E[S_i^2]/f_i^2)}{(1-\rho_i)^3} \tag{4.6}$$

Notice that the Equation 4.6 requires the second moment of the service times, which we assume that can be also directly measured in a pre-production execution of the application on the physical server.

Given the mean and the variance of the response time of applications, Chebyshev's inequality limits the probability of a transaction's response time being greater than the SLA threshold as:

$$P(R_i \geq R_i^{SLA}) \leq \frac{Var[R_i]}{(R_i^{SLA} - E[R_i])^2} \leq \alpha_i \tag{4.7}$$

Although Equation 4.7 often give more precise bounds than Equation 4.4, it requires more information, such as the queue type and the second moment of service times.

The last proposed approximation uses the response time CDF exact solution for calculating the percentile of the response times for the M/M/1 queue [21] so as to express the tail distribution response time requirement as:

$$P(R_i \geq R_i^{SLA}) = e^{-R_i^{SLA}(f_i/E[S_i])(1-\rho_i)} \leq \alpha_i \tag{4.8}$$

Notice that Equation 4.8 is only exact when there is no rejection of requests. Thus, since we are assuming a general admission control policy, this expression is also an approximation. The equivalent result for M/G/1 (PS) queue is still an open problem [45].

In Chapter 5, we compare the level of accuracy provided by each of the approximations, Markov, Chebyshev and Percentile, presented in this section.

4.5 Summary

This chapter presents the sub-parts that comprise the business-driven capacity management model: the underlying business model, the autonomic framework, the optimization model, and, finally, the performance model. In the next chapter, we assess the effectiveness of the model through a numerical analysis.

Chapter 5

Experimental Analysis

In this chapter, we evaluate the proposed capacity management model as well as the approximations proposed to model the performance of application classes. Our objective is to demonstrate the effectiveness of the autonomous business-driven capacity management approach to increase the provider's business potential by reacting as a response to changing business priorities caused by workload changes. However, if the maximization of the provider's business objectives were achieved at the expense of the stability of the services during their operation, the reliability of IDCs would be seriously penalized. For example, customers who run critical businesses would be reluctant to host their applications in such environment. Thus, we evaluate the degree of stability presented by applications when using the proposed approach by analysing three performance metrics: i) the queue length of the services, ii) the response time requirement satisfaction, and iii) the throughput achieved by the classes.

Our results are derived from the discrete event simulation of different classes executing concurrently in the environment considered. Accordingly, each class runs in a separate VM and competes for the IDC capacity. The capacity manager component is called periodically in order to change the settings of each VM, that is, f_i and λ_i^{acc}, in response to changing business priorities. We have also instrumented the simulator in order to collect performance measurements, such as throughput, response time, queue length, and utilization of VMs.

Using the approximations presented in Section 4.4, the optimization model becomes a non-linear problem (due to the non-linear constraint of response time). Thus, we have conducted the experiments using the non-linear solver DONLP2 [38]. Experiments with up to 50 application classes resulted in solution times under one millisecond in a Athlon 2.2 GHz with 512 MB of main memory.

5.1 Experimental Setup

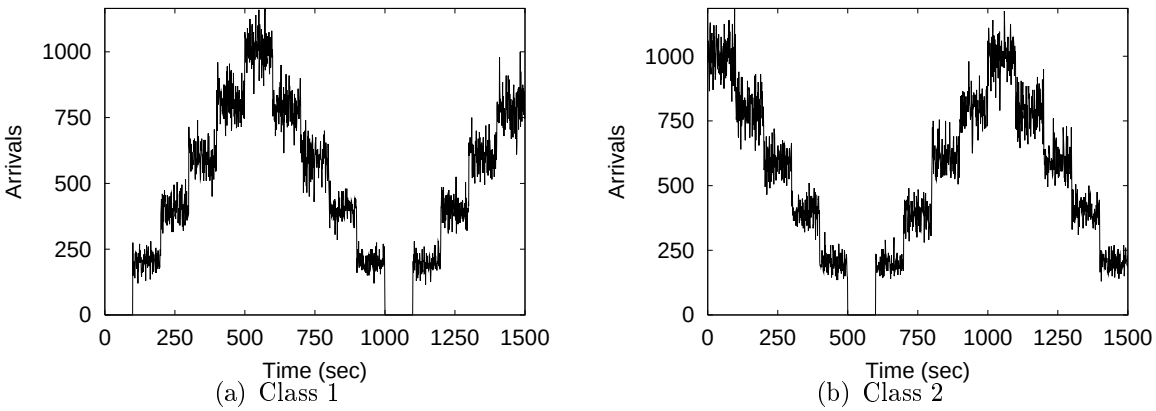
In the experiments described in this section, in order to provide a close observation of the behavior of classes during their operation, we present results for two competing VMs maintained on top of the shared physical infra-structure. Although it suffices to assess the quantitative results, we have run experiments varying the number of VMs up to 50, and the qualitative results presented similar conclusions.

It is known that different admission control policies have different effects on the performance of the system [31]. Since the main concentration in this study is on the resulting effect of employing the capacity management model, for the sake of simplicity, in the following set of experiments, a strict and conservative admission policy based on tokens is employed. That is to say that at each second a limited number of tokens is set up, which transactions need to acquire in order to enter the system. This limit is configured (dynamically) at each VM, using the value of variable λ_i^{acc} after solving the optimization model.

In the experiments involving adaptation, the selection and evaluation of the workload forecasting method is out of the scope of this discussion. In fact, one can use one of the existing time series forecast methods [1, 11, 19] and obtain results that vary with different degrees of accuracy. Instead, we assume an ideal forecasting, meaning that the future request arrival rate is known *a priori*. We also assume that there is no time limitation for adapting the system.

For the case in which the performance of the classes is approximated as a M/M/1 queue, we simulate a physical server in which both applications are served with exponentially distributed service time whose parameter is $E[S_i] = 10^{-3}$, using FCFS scheduling discipline. For the M/G/1 (PS) case, the applications experience the same service time distribution with the same parameter, but with processor sharing scheduling discipline. Last, since we aim at studying the performance of the virtual machines only, the virtualization layer allows us to completely abstract the actual physical topology of the data center.

The set of experiments were driven by synthetic workloads. Workload generators are used to submit requests following a periodic step-like non-homogeneous Poisson processes [29] whose shape is similar to the ones presented in Figure 5.1. This figure presents the arrivals submitted to each of the classes where the average arrival rate in the traces ranges from 0 to 1000 arrivals per second, each periodic cycle lasts for 1000 seconds, and each step lasts for 100 seconds. Notice that we introduce a shift of 500 seconds in the arrival process presented in Figure 5.1(b) with respect to that in Figure 5.1(a) in order to produce a displacement of their periods. This scenario



(a) Class 1 (b) Class 2

Figure 5.1: Synthetic periodic step-like Poisson arrival processes submitted to a) application class 1 b) application class 2.

makes for an interesting analysis of the ability of the system to re-deploy its capacity, by assigning the idle capacity of the VM experiencing low workload activity to the overloaded one.

Although the system has the ability to reconfigure itself to best serve applications, a preliminary capacity planning [25] study is required in order for the provider to determine how many and what types of applications can co-exist together in a data center. This initial planning involves an offline characterization of application performance data that can be previously collected in a pre-production isolated environment [4]. Accordingly, this study, known as statistical multiplexing [33, 40], aims at detecting the average, the peak, and the behavior of the workload of different applications so as to determine, for instance, whether there is enough capacity to serve the average workload of all applications simultaneously, whether the applications receive surges of requests at the same or different times, and whether the workload idle times of one application coincide with, and, therefore, can be complemented by, the peak times of the others. Since we can control the synthetic workload behavior, we omit this step here. However, notice that our setup is done in such a way that the server is underprovisioned to serve the two application experiencing values of workload higher than their average simultaneously. Accordingly, the maximum throughput of each application using the entire physical server capacity is 1000 tps (given by Equation 4.2), and the request arrival rates of both applications vary from 0 to 1000 requests per second.

Given the characteristics of the workload, the system is configured to adapt at the end of each workload step, that is 100 seconds, and, since the average request rate is stable over the steps, we bypass $P\{Z_i\}$ by setting it to 1. In addition, as the main focus

Table 5.1: SLA and system configurations used in the experiments.

υ_i	α_i	X_i^{NSLA}	X_i^{SSLA}	R_i^{SLA}	c_i	π_i
0.95	0.1	500	1000	0.1	1.0	0.5

is on the ability of the system to adapt in response to changing business priorities as a result of workload changes, equal requirements are set up for the classes, which are summarized in Table 5.1. Notice that the reward value is half of the penalty cost. This is done because it would induce the capacity optimizer to prioritize the avoidance of penalties, that is, to decide on reward capitalization only if all the NSLA requirements are met.

As a reference, we have run the classes without the capacity manager intervention, referred to here as the *Static* approach. In the latter, the system is started up with the configuration obtained from the solution of the optimization model, using the average value of the workload over the entire simulation period, and no further adaptation is triggered.

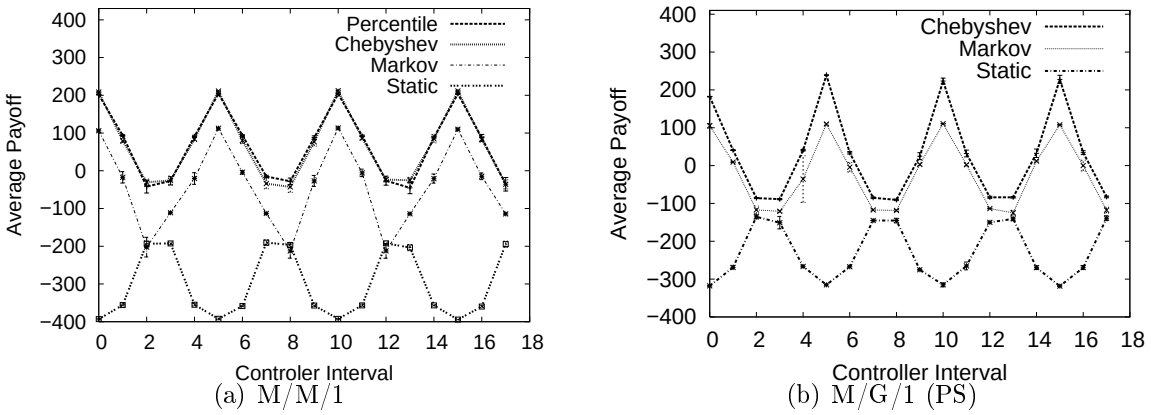
Finally, it is important to emphasize that synthetic workloads combined with the proposed capacity management approach allow us to assess the best case scenario. Conversely, the worst case is given by the Static approach. As we are dealing with the general class of transactional services, their workload traffics may differ significantly. As a consequence, the effectiveness of the approach would vary according to the characteristics of these workloads, what kinds of applications are being hosted together, etc., and these results are expected to fall between the two extreme cases.

5.2 Numerical Results

In sequence, the effect of the concurrent execution of the classes for the M/M/1 and M/G/1 (PS) modeling cases, using the three approximations, namely, Markov, Chebyshev and Percentile, is analyzed.

We present the average payoff (i.e., the net result from penalties and rewards) over 10 runs of the execution of the classes for each modeling case, throughout a period corresponding to half an hour, along with the 90% confidence interval of the averages in Figure 5.2.

The results presented in Figure 5.2 show that for the M/M/1 case, using the Static approach, the payoff for the execution varies approximately from -400 to -200 while using the Markov approximation this value varies between -240 and 100. The payoff



(a) M/M/1 (b) M/G/1 (PS)

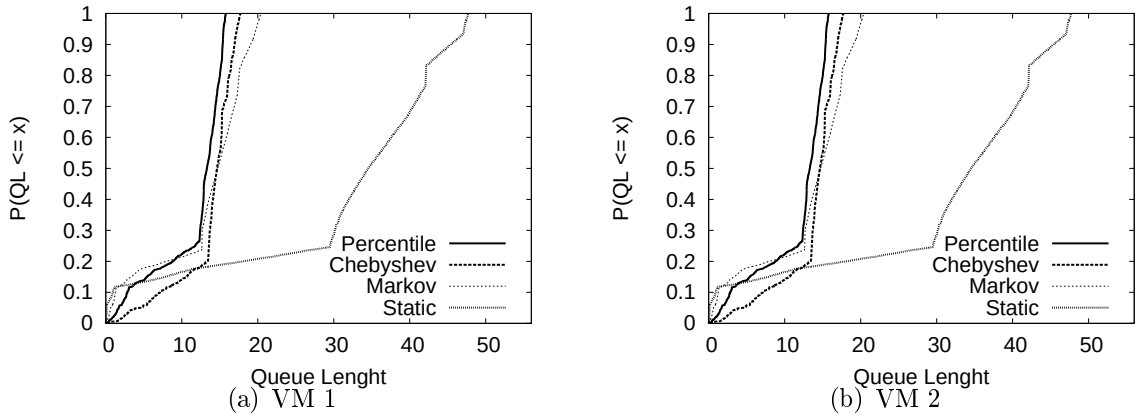
Figure 5.2: Average payoff for the execution of applications produced by the autonomous approach along with the 90% confidence interval using the Percentile, Chebyshev and Markov approximations, compared with the Static approach a) for the M/M/1 case b) for M/G/1 (PS) case.

obtained from the Chebyshev approximation is slightly smaller than the payoff obtained by the Percentile approach even though the two curves overlap most of the time. These latter two approximations produced values that range approximately from 0 to 200. In the M/G/1 (PS) modeling case, the Static approach produces payoffs varying approximately from -310 to -130, while using the Markov approximation these values range from -125 to 100, and using Chebyshev from -90 to 240.

Notice that the repeating pattern of the curves are produced by the periodic behavior of the workloads submitted to the VMs. In this case, the peaks of payoff values for the static case corresponds to periods of time in which the two workloads approach their average values together. Accordingly, these are periods where the autonomous approaches obtain their smaller payoff, as there is little opportunity to assign idle capacity of one VM to another. Indeed, periods where the peak of the workload submitted to one VM complements the valley of the other are the cases in which the autonomous approaches achieved the highest payoff values.

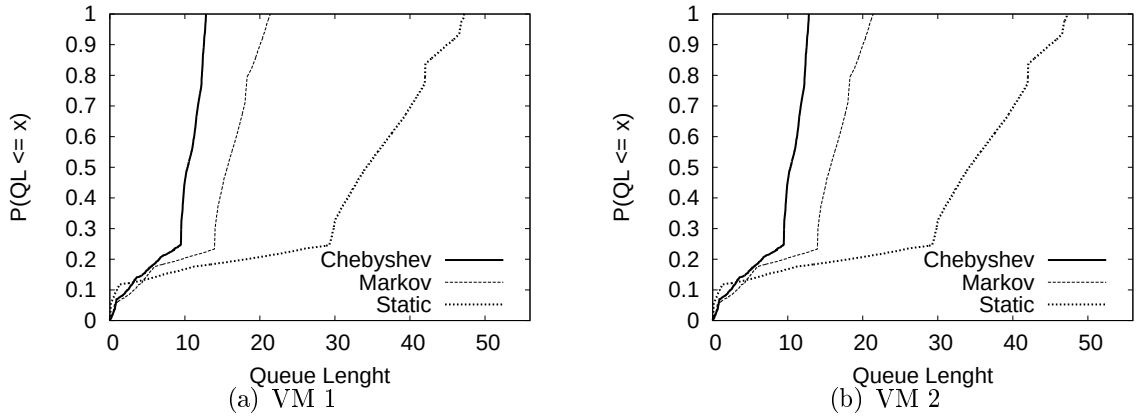
We now analyze the stability of the services through their performance behavior, by analyzing the queue length, the response time requirement satisfaction, and the processing rate throughout the runs.

Figures 5.3 and 5.4 show the CDFs of the queue lengths at both VMs, collected at each transaction departure. The two figures show that, for the M/M/1 case, when the system is adapted using any of the approximations, the queue length consistently remains under 15, whereas when there is no adaptation, the queue lengths vary from



(a) VM 1 (b) VM 2

Figure 5.3: CDF of the queue lengths over time of a) VM 1 and b) VM 2 for the M/M/1 case



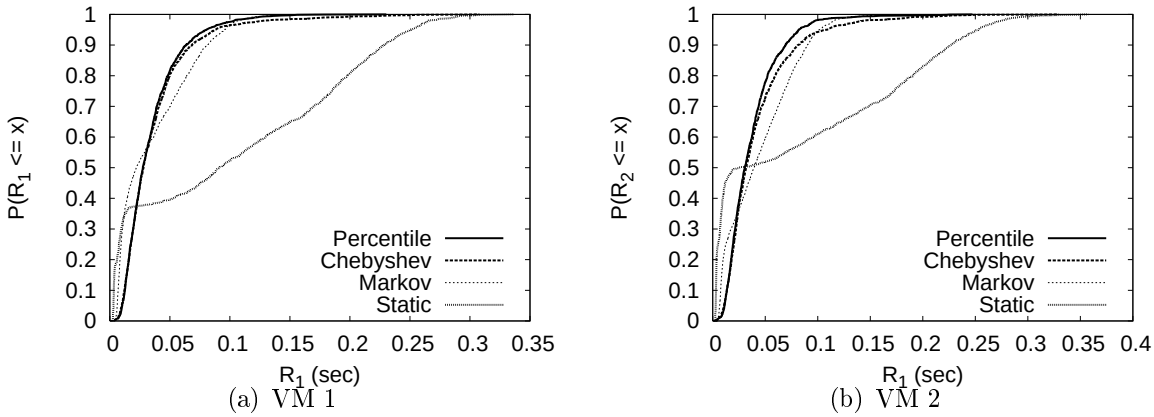
(a) VM 1 (b) VM 2

Figure 5.4: CDF of the queue lengths over time of a) VM 1 and b) VM 2 for the M/G/1 (PS) case.

30 to 48 in approximately 75% of the cases. For the M/G/1 (PS) case, using the Chebyshev approximation, the queue length stays at 10 most of the time; using the Markov approximation, this value is 15; and when no adaptation takes places, it varies between 30 and 48 in approximately 75% of the cases.

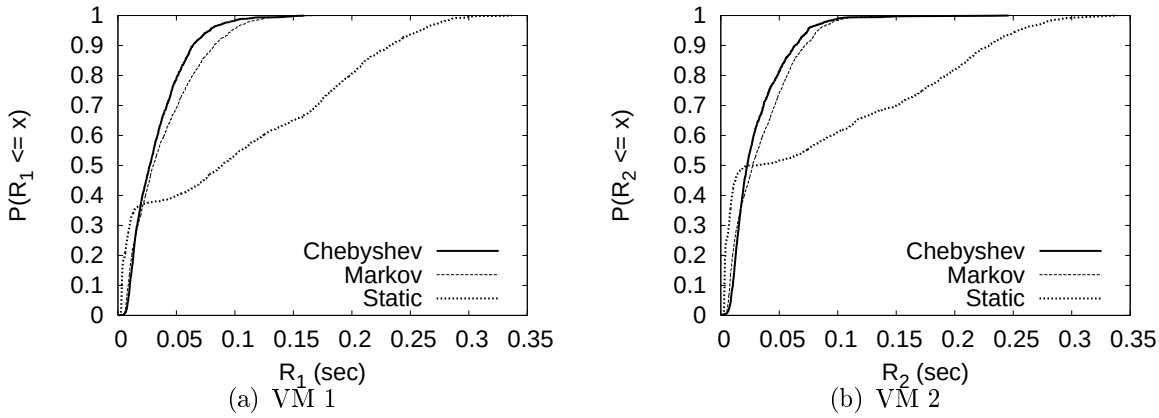
Notice that, in the cases where there is adaptation, the queue lengths stay within the theoretical value of the average number of customers for M/M/1 and M/G/1 (PS) queues [21], given by $\phi_i = \frac{\rho_i^2}{1-\rho_i}$. Replacing ρ_i by the maximum utilization values $\upsilon_i = 0.95$, at which we planned the VMs' operation, gives $\phi_i = 18.05$.

In order to analyze the satisfaction of the SLA tail distribution response time requirement during the runs, we have plotted the CDFs of the response times for the



(a) VM 1 (b) VM 2

Figure 5.5: CDF of the response times of a) VM 1 and b) VM 2 for the M/M/1 case.

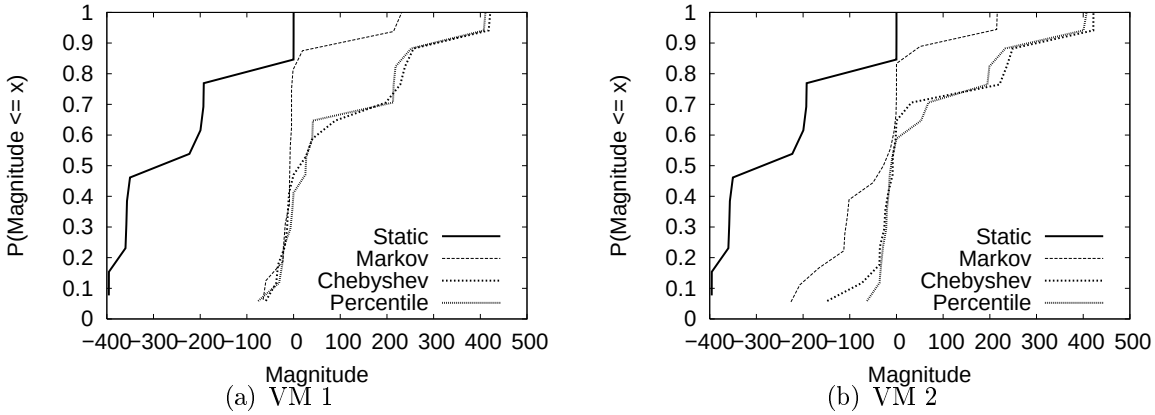


(a) VM 1 (b) VM 2

Figure 5.6: CDF of the response times of a) VM 1 and b) VM2 for the M/G/1 (PS) case

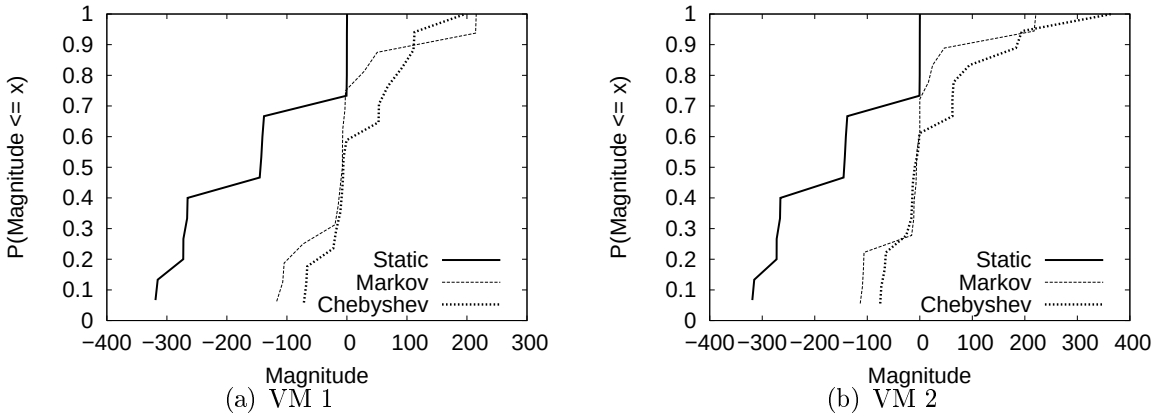
VMs in Figure 5.5 and 5.6 for both modeling cases. Notice that the response time requirement is met for both VMs, in the two modeling cases, when using the autonomous approaches with any approximation. Accordingly, the response time of the transactions were shorter than 0.1 seconds more than 90% of the time. By contrast, using the static approach the response time threshold is respected approximately 55% and 60% of the time for the VMs 1 and 2 respectively in the M/M/1 case, and approximately 57% for both VMs in the M/G/1 (PS) cases.

Up to this point, we have shown that the response time requirements of the classes are attained by using any of the approximations for the two modeling cases and, therefore, they are equally effective for guaranteeing the response time requirement satisfaction. In contrast, the resulting throughput explains why one approximation results in



(a) VM 1 (b) VM 2

Figure 5.7: CDFs of the magnitude of penalties and rewards produced by a) VM 1 and b) VM 2, for the M/M/1 case.



(a) VM 1 (b) VM 2

Figure 5.8: CDFs of the magnitude of penalties and rewards produced by a) VM 1 and b) VM 2, for the M/G/1 (PS) case.

higher payoffs than the others. In order to observe this effect, let us analyze the magnitude of the penalties and rewards achieved by each class separately for both modeling cases. We present the CDFs of the magnitude of the penalties and rewards over the controller intervals for each VM, for the M/M/1 modeling case in Figure 5.7 and for the M/G/1 (PS) case in Figure 5.8.

Clearly, in all the plots, the static approach resulted in significantly higher penalties and smaller rewards in comparison to the results provided when the system is adapted. In addition, the values resulting from the Markov approximation are also smaller than the ones achieved when using Chebyshev and Percentile. These latter two approximations produced curves that almost coincide. For the M/M/1 case, this is clear in

the values corresponding to rewards for both Figures 5.7(a) and 5.7(b), and is also observed in the values of penalties in the Figure 5.7(b) only. This is because, in this experiment, the classes have the same requirements, and, thus, the solver favored VM 1 in these cases, since this decision produces a equally optimal solution as distributing the capacity evenly between the classes. For the M/G/1 (PS) queue, this difference is also observable, especially in reward values. This result conforms to Figure 5.2(b) where the payoff difference produced by the Markov and Chebyshev approximations is greater in the higher values while they nearly coincide in the smaller values.

To sum up, we conclude that the use of the autonomous business-driven capacity management approach proposed is able to react to changing business priorities, thereby significantly increasing the net result from penalties and rewards in comparison to the Static approach. We can also observe that the Markov approximation is more conservative than the others, because it over-estimates capacity needs due to the response time requirements. On the other hand, the Chebyshev approximation produces average payoffs slightly smaller than the Percentile solution for the M/M/1 case. Moreover, Chebyshev can also be applied to the M/G/1 (PS) modeling case, producing improvements in comparison with the Markov approximation. Although a closed formula to estimate the response time percentile is unknown for several types of queues (in the queuing theory), due to the satisfactory performance of the Chebyshev approximation for M/M/1 queues, which closely matched the results derived from the exact formula for finding the response time percentile, we conjecture that it may also present effective performance for other types of queues, such as M/G/1, for which a reference (results using the exact expression) cannot be established. Since the reliability of service operation is fundamental to prove the effectiveness of the approach, we also demonstrated that, when the approach is applied, the classes present stable performance by maintaining their queue lengths controlled and attaining the response time requirements.

The results presented in this Chapter provide a solid base upon which future developments can be built. In Chapter 6 we offer our conclusions and point out future work directions.

Chapter 6

Conclusions and Future Work

The problem of mapping systems decisions into implications on high-level business objectives presents a enormous challenge. To make matters more complicated, providers aim at attaining such business objectives in the most cost-effective manner while preserving the reliability of services. To tackle this problem, we have considered an autonomous business-driven capacity management approach for Internet data centers that allow for service differentiation to applications. With the goal of best exploring Internet data center resources to the provider's best SLA business advantage, a model that encompasses both business and system issues has been proposed.

In the business domain, in order to provide the IDC with the ability to accounting for actual use, in an environment with great fluctuation of system states due to the characteristics of workloads, we propose a multi-level SLA specification for different operation modes. We illustrated this point through a two-level contract which defines two different operation modes, namely normal, which corresponds to the service level customers contract to satisfy their usual demands, and surge, which is a service level upper limit, up to which customers are willing to pay for receiving extra capacity in order to avoid performance problems with their applications during workload peaks. In addition, the multi-level SLA contract works together with a cost model based on penalties, incurred by the provider when the minimum expectations are not met, and rewards, paid by customers for occasional extra capacity provisioning (more than that contracted for the normal operation mode). In light of this SLA business model, we defined the provider business objective as the maximization of the net result from penalties and rewards during its operation.

In the systems domain, we developed a queuing model for predicting the Internet service performance, thereby determining capacity needs, with respect to the SLA requirements of throughput, subjected to a tail distribution response time guarantee. Since the expressions that capture realistically the characteristics of all the service

types considered are not available in the queuing theory, we evaluate approximations, namely Markov, Chebyshev and Percentile, for predicting the performance of the hosted services under two different scheduling disciplines that services may be subject to, first come first served (FCFS) and processor sharing (PS).

The business and system models are woven together through a non-linear optimization model, which receives as inputs system parameters and data extracted from SLA contracts so as to allow the capacity manager to adapt the IDC to changing capacity needs in real time in such a way that a provider's business objective is maximized.

Through a numerical analysis derived from the simulation of an IDC hosting concurrent applications, we were able to assess the effectiveness of the approach. Our observations allow us to conclude that, in contrast with a static approach in which the system is not adapted over time, when the capacity management model proposed in this work takes place, changing business priorities can be detected, and corrective measures can be taken in order to increase the financial benefit derived from IDCs. This result is observable when using any of the performance model approximations proposed. Even so, the difference in the level of accuracy resulting from the alternative approximations to express the tail distribution requirement of response time is considered. Accordingly, we found that under both scheduling discipline, even though Markov approximation can improve the net result from penalties and rewards while requiring little information, it provides loose upper bounds, thereby being unnecessarily conservative in comparison with the other approximations, and, as a consequence, overestimating capacity needs. In contrast, Chebyshev approximation presents similar level of accuracy in comparison to the Percentile approach and can be applied for modeling the performance of applications under both FCFS and PS scheduling disciplines.

In another aspect, the queuing model approximations allow the capacity management model to scale with the number of applications being hosted, since they are based on simple equations. Accordingly, the optimization model presents low computational cost, thereby providing the capacity management approach with the ability to interact with the system in real time.

Last, application owners are usually concerned about reliability. That is to say that providers cannot overlook the stability of services when trying to maximize their business objective. Indeed, in order to guarantee that their applications would be always available and stable, customers who run critical business may even decide to incur high costs by overprovisioning resources. In contrast, we demonstrated, by analyzing the performance data obtained through the instrumentation of the simulator, such as queuing length and response time, that our approach preserve the stability of the services, thereby making overprovision unnecessary.

To conclude, the results obtained serve as basis to future developments and advances in this research topic. In the following section, we indicate future research directions and describe our ongoing work.

6.1 Future Work

The understanding of this first scope of the problem makes for interesting future analysis, which can be derived by incrementing the model presented here. In this fashion, we have established directions for future work on business and system strands of the problem.

The first obvious task is to validate the model by building the real system and exploring the approach using different traffic patterns, extracted from real Internet service traces. Second, in this analysis, we made use of a high level of abstraction, by considering each VM as a single resource. However, the performance of a system is usually limited only by a single bottleneck resource. However, the single resource abstraction forces the whole VM's capacity to change by a given factor, in which case all its resources' capacity change. In contrast, we believe that, if it were possible to manage the devices' capacities independently, this would produce a higher level of accuracy in cases that only a bottleneck capacity adjust would suffice. As a consequence, we expect that this improvement would also result in better utilization of the available resources. Although this modeling approach has already been applied for deterministic performance metric requirement (i.e., maximum average response time) [39], this problem becomes extremely complex when tail distribution requirements (as we consider in this work) are involved. This is because the queuing theory lacks results for estimating the probability distribution of performance metrics (i.e., response time) in models composed by queues in tandem for different traffic patterns. Lin et al. propose an approximation for this purpose in a theoretical work [22]. However, an experimental analysis which presents the degree of accuracy of this model experimentally has not been conducted. In light of this, we have been working on an extension for the model that allows for the capacity management of each VM device independently (i.e., CPU, disk etc). In addition, we also aim at considering for future work relevant system features in Internet data centers, such as service replication and multi-tiered applications. Last, we intend to explore the influences of the employing different workload prediction techniques and admission control schemes.

In this work we dealt with the effects of changing workloads in the business priority of services within a data center. However, in our ongoing work, we identified that business priorities are influenced by at least three levels of abstraction, thereby

requiring further investigation. The first one is workload changes, which is considered in this work. The second is pricing schemes. Accordingly, phone companies explore this factor to change the workload received by their users. For example, if they establish a higher cost to make phone calls during business hours, the direct consequence is that users may wait to place some calls at night or in the weekends. Last, the third level relates to the system load. To illustrate this point we can use the example of car traffic. Unless absolutely necessary, drivers may avoid commuting during a traffic jam. Transferring this example to systems, in order to have a better interaction experience, users may prefer to place transactions during idle or low utilization times. Thus, we believe that these three levels of abstraction influence each other and we have been using game-theoretical approaches to study the equilibrium situation of all these factors acting simultaneously. We believe that this study will result in business models which mimic business and behavioral dynamics more realistic for Internet data centers.

Appendix A

Generating Step-Like Non-Homogeneous Poisson Processes

In this appendix, we describe how to generate a step-like non-homogeneous Poisson process whose shape can be seem in Figure 5.1.

A non-homogeneous Poisson process [29] has similar characteristic with the Poisson process, but it relaxes the assumption of stationary increments. Thus, it allows for arrival rates which can vary with time, according to a intensity function $\lambda(t)$ $(t > 0)$.

In order to generate a non-homogeneous Poisson process with a step-like shape, we use a periodic intensity function with the parameters presented in Table A.1. The first parameter, λ^{MAX}, represents the maximum Poisson arrival rate; s is the number of steps to generate for each periodic cycle; the periodic cycle duration is denoted by c; last, o is the offset of time, which specifies the point at which to start the process.

Table A.1: Intensity function parameters.

Symbol	**D**escription
λ^{MAX}	maximum arrival rate.
s	number of steps at each cycle.
c	periodic cycle duration.
o	time offset to start the process.

Let us illustrate the use of parameters by presenting at Figure A.1 a process whose configuration is shown in Table A.2.

From the parameters, the height of steps can be derived dividing the maximum arrival rate by the number of steps at each half cycle, $\frac{\lambda^{MAX}}{s/2}$, and the duration of

Table A.2: Process configuration.

λ^{MAX}	s	c	o
1000	10	1000	0

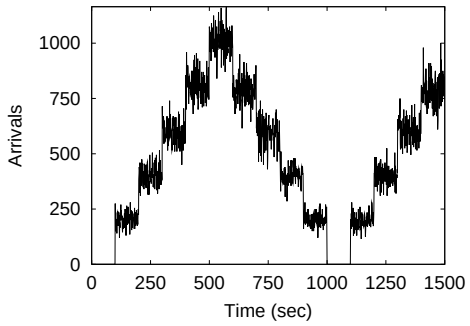


Figure A.1: Step-like non-homogeneous Poisson process.

each step is simply the cycle duration divided by the number of steps, $\frac{c}{s}$. Thus, the expression for the intensity function is:

$$\lambda(t) = \frac{\lambda^{MAX}}{s/2}\left[\left((t+o)\frac{s}{c}\right)(mod(s/2))\right] \tag{A.1}$$

However, Equation A.1 generate traces that have half of the desired period with ascending steps only. In order to correct this behavior, it is required to invert the second half of each periodic cycle, by inverting the sign of $\lambda(t)$ whenever $(t+o)mod\ c < \frac{c}{2}$.

Bibliography

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

[2] Bruno Abrahao, Virgilio Almeida, and Jussara Almeida. A self-managing approach to enterprise application performance. In *First International Workshop on Adaptive and Self-Managing Enterprise Applications (ASMEA'05)*, 2005.

[3] Bruno Abrahao, Virgilio Almeida, Jussara Almeida, Alex Zhang, Dirk Beyer, and Fereydoon Safai. Self-adaptive sla-driven capacity management for Internet services. In *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, April 2006.

[4] Bruno Abrahao and Alex Zhang. Characterizing application workloads on CPU utilization for utility computing. Technical Report HPL-2004-157, Hewllet-Packard Labs, 2004.

[5] Arthur Andrzejak, Martin Arlitt, and Jerry Rolia. Bounding the resource savings of utility computing models. Technical Report HPL-2002-339, Hewllet-Packard Labs, 2002.

[6] K. Appleby, S. Fakhouri, L. Fong, G. Goldzmidt, and M. Kalantar. Oceano - SLA based management of a computing utility. In *IFIP/IEEE Int'l Symposium on Integrated Network Management*, May 2001.

[7] K. C. Appleby, S. B. Calo, J. R. Giles, and K.-W. Lee. Policy-based automated provisioning. *IBM Systems Journal*, 1(43):120–135, 2004.

[8] Martin Arlitt and Tal Jin. Workload characterization of the 1998 world cup web site. Technical Report HPL-1999-35R1, Hewllet-Packard Labs, 1999.

[9] G. Banga, P. Druschel, and J.C. Mogul. Resource containers: A new facility for resource management in server systems. In *3rd USENIX Symposium on Operating Systems Design and Implementation*, 1999.

[10] Claudio Bartolini and Mathias Sallé. Business driven prioritization of service incidents. In *15th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, DSOM*, 2004.

[11] Y. Baryshnikov, E. Coffman, D. Rubenstein, and B. Yimwadsana. Traffic prediction on the Internet. Technical report, Columbia University, New York, 2002.

[12] Mohamed Bennani and Daniel Menascé. Resource allocation for autonomic data centers using analytic performance models. In *IEEE Int'l Conference on Autonomic Computing*, 2005.

[13] M. J. Buco, R. N. Chang, L. Z. Luan, C. Ward, J. L. Wolf, and P. S. Yu. Utility computing SLA management based upon business objectives. *IBM Systems Journal*, 1(43):159–178, 2004.

[14] Y. Diao, N. Gandhi, J.L. Hellerstein, S.Parekh, and D. M. Tilbury. Using MIMO feedback control to enforce policies for interrelated metrics with application to the apache web server. In *IEEE/IFIP Network Operations and Management Symp.*, 2002.

[15] T. Eilam, K. Appleby, J. Breh, G. Breiter, H. Daur, S. A. Fakhouri, G. D. H. Hunt, T. Lu, S. D. Miller, L. B. Mummert, J. A. Pershing, and H. Wagner. Using a utility computing framework to develop utility systems. *IBM Systems Journal*, 1(43):97–120, 2004.

[16] Ian Foster and Carl Kesselman. *The Grid 2.* Morgan Kaufmann, 2nd edition edition, 2004.

[17] Sukumar Ghosh and T. Herman. *Self-Stabilizing Systems.* Carleton University Press, 1997.

[18] Sven Graupner, Ralf König, Vijay Machiraju, Jim Pruyne, Akhil Sahai, and Aad Van Moorsel. Impact of virtualization on management systems. Technical Report HPL-2003-125, Hewllet-Packard, 2003.

[19] J. Hellerstein, F. Zhang, and P. Shahabuddin. A statistical approach to predictive detection. *Computer Networks*, January 2000.

[20] Klaus Herrmann, Gero Muhl, and Kurt Geihs. Self-management: The solution to complexity or just another problem? *IEEE Distributed Systems*, 6(1), 2005.

[21] Leonard Kleinrock. *Queueing Systems Volume I: Theory.* Wiley Interscience, 1975.

[22] Wuqin Lin, Zhen Liu, Cathy H. Xia, and Li Zhang. Optimal capacity allocation for multi-tiered IT systems with end-to-end delay guarantees. In *IFIP WG 7.3 International Symposium on Computer Performance, Modeling, Measurements and Evaluation (to Appear)*, 2005.

[23] Xue Liu, Xiaoyun Zhu, Sharad Singhal, and Martin Arlitt. Adaptive entitlement control of resource containers on shared servers. In *9th IFIP/IEEE Int'l Symposium on Integrated Network Management*, 2005.

[24] Zhen Liu, Mark Squillante, and Joel L. Wolf. On maximizing service-level-agreement profits. In *ACM Eletronic Commerce Conference*, October 2001.

[25] Daniel Menascé and Virgilio Almeida. *Capacity Planning for Web Services: metrics. models and methods.* Prentice Hall, 2000.

[26] Daniel Menascé, Virgilio Almeida, and Lawrence Dowdy. *Performance by Design.* Prentice Hall, 2003.

[27] Daniel A. Menascé and Mohamed N. Bennani. On the use of performance models to design self-managing computer systems. *IEEE Distributed Systems*, 6(1), 2005.

[28] G. A. Paleologo. Price-at-risk: A methodology for pricing utility computing services. *IBM Systems Journal*, 1(43):20–31, 2004.

[29] A. Papoulis and S. U. Pillai. *Probability, Random Variables, and Stochastic Processes.* McGraw-Hill, 4th edition, 2002.

[30] V. Paxson and S. Floyd. Wide-area traffic: The failure of poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, 1995.

[31] H. G. Perros and K. H Elsayed. Call admission control schemes : A review. *IEEE Magazine on Communications*, 34(11):82–91, 1996.

[32] M. A. Rappa. The utility business model and the future of computing services. *IBM Systems Journal*, 1(43):32–41, 2004.

[33] J. Rolia, X. Zhu, M. Arlitt, and A. Andrzejak. Statistical service assurance for applications in utility grid environments. Technical report, Hewlett-Packard Laboratories, Palo Alto, CA, USA, 2002. HPL-2002-155.

[34] J. W. Ross and G. Westerman. Preparing for utility computing: The role of IT architecture and relationship management. *IBM Systems Journal*, 1(43):5–19, 2004.

[35] Cipriano A. Santos, Akhil Sahai, Xiaoyun Zhu, Dirk Beyer, Vijay Machiraju, and Sharad Singhal. Policy-based resource assignment in utility computing environments. Technical Report HPL-2004-142, Hewllet-Packard Labs, 2004.

[36] Avi Silberschatz, Peter Baer Galvin, and Greg Gagne. *Operating Systems Concepts, Seventh Edition*. John Wiley & Sons, 2004.

[37] Sharad Singhal, Sven Graupner, Akhil Sahai, Vijay Machiraju, Xiaoyun Zhu Jim Pruyne, Jerry Rolia, Martin Arlitt, and Pano Santos. Quartermaster - a resource utility system. Technical Report HPL-2004-152, Hewllet-Packard Labs, 2004.

[38] P. Spellucci. An SQP method for general nonlinear programs using only equality constrained subproblems. *Mathematical Programming*, 3(82):413–448, 1998.

[39] Bhuvan Urgaonkar, Giovanni Pacifici, Prashant Shenoy, Mike Spreitzer, and Asser Tantawi. An analytical model for multi-tier internet services and its applications. In *ACM SIGMETRICS*, 2005.

[40] Bhuvan Urgaonkar, Prashant Shenoy, and Timoty Roscoe. Resource overbooking and application profiling in shared hosting platforms. In *Proceedings of the USENIX Symp. on Operating Systems Design and Implementation*, Dec. 2002.

[41] Daniel Villela, Prashant Pradhan, and Dan Rubenstein. Provisioning servers in the application tier for e-commerce systems. In *IEEE Int'l Workshop on Quality of Service*, June 2004.

[42] W. E. Walsh, G. Tesauro, J.O. Kephart, and R. Das. Utility functions in autonomic computing. In *IEEE Internaltional Conf. Autonomic Computing (ICAC'04)*, 2004.

[43] Andrew Whitaker, Richard S. Cox, Marianne Shaw, and Steven D. Gribble. Constructing services with interposable virtual hardware. In *First Symposium on Networked Systems Design and Implementation (NSDI '04)*, 2004.

[44] John Wilkes, Jeffrey Mogul, and Jaap Suermondt. Utilification. Technical report, Hewlett-Packard Laboratories, Palo Alto, CA, USA, 2004. HPL-2004-124.

[45] S. F. Yashkov. Processor-sharing queues: Some progress in analysis. *Queuing Systems*, 2:1–17, 1987.

[46] Alex Zhang, Pano Santos, Dirk Beyer, and Hsiu-Khuern Tang. Optimal server resource allocation using an open queuing network model of response time. Technical Report HPL-2002-301, Hewllet-Packard Labs, 2002.

[47] Xiaoyun Zhu, Cipriano Santos, Julie Ward, Dirk Beyer, and Sharad Singhal. Resource assignment for large-scale computing utilities using mathematical programming. Technical Report HPL-2003-243R1, Hewllet-Packard Labs, 2003.