Gerenciamento de recursos computacionais em plataformas de Computação em Nuvem

Cássio Alexandre Paixão
Silva Alkmin

Dissertação apresentada
ao
Instituto de Matemática e Estatística
da
Universidade de São Paulo
para
obtenção do título
de
Mestre em Ciências

Programa de Pós-Graduação em Ciência da Computação
Orientador: Dr. Daniel de Angelis Cordeiro

Durante parte do desenvolvimento deste trabalho
o autor recebeu auxílio financeiro da CAPES

São Paulo, dezembro de 2014

Gerenciamento de recursos computacionais em plataformas de Computação em Nuvem

Esta é a versão original da dissertação elaborada pelo candidato Cássio Alexandre Paixão Silva Alkmin, tal como submetida à Comissão Julgadora.

Resumo

ALKMIN, C. P. **Gerenciamento de recursos computacionais em plataformas de Computação em Nuvem**. 2014. 53 f. Dissertação (Mestrado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2014.

A indústria de tecnologia da informação vivencia grandes avanços com o desenvolvimento de tecnologias de Computação em Nuvem, cujas plataformas permitem a locação sob demanda de grandes quantidades de recursos computacionais. Os desafios de infraestrutura para prover tais recursos de modo mais eficiente, sem comprometer a qualidade do serviço contratado, ainda são vários. O gerenciamento de recursos computacionais se destaca entre eles, frente à diversidade de aplicações que são executadas nesses ambientes e à demanda volátil dos recursos, resultante da liberdade que os usuários, humanos ou mesmo outras aplicações, têm para utilizar os recursos contratados. A simulação de plataformas de Computação em Nuvem permite analisar, com baixo custo, efeitos sobre o uso dos recursos computacionais ao empregar diferentes estratégias para a gestão de recursos nessas plataformas, como consolidação de servidores, predição de desempenho, migração de máquinas virtuais e estratégias para suspensão de servidores físicos. Este trabalho investiga estratégias para gerenciar esses recursos, com ênfase na consolidação de servidores, no seu mapeamento para o problema de *bin packing* vetorial com cestos de tamanho variável, e na predição de desempenho, com análise do histórico da demanda de recursos computacionais como uma série temporal. Com tal embasamento teórico, este estudo analisa os efeitos que diferentes estratégias de gestão de recursos computacionais têm sobre a eficiência no uso dos recursos computacionais e no cumprimento de acordos de nível de serviço. Também se apresenta o SimMyCloud, arcabouço desenvolvido para simulação de plataformas de Computação em Nuvem. Um traço de execução real de uma plataforma de Computação em Nuvem, disponibilizado pelo Google, foi utilizado como base para as simulações, cujas execuções evidenciaram forte influência da escolha de uma estratégia de predição de desempenho para um uso eficiente dos recursos computacionais.

Palavras-chave: computação em nuvem, consolidação de servidores, predição de desempenho.

Abstract

ALKMIN, C. P. **Computational resources management at cloud computing platforms**. 2014. 53 f. Dissertação (Mestrado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2014.

The technology information industry is facing a grown up moment due to the development of Cloud Computing technologies, whose platforms permit an on demand rent of big amounts of computational resources. The infrastructural challenges to provide those resources efficiently, without compromising the quality of service, are vast. The computational resources management is one of the main ones, due to the diversity of applications running in these environments and the volatile resources demand. This volatility is a consequence from the freedom the users, humans or other applications, have to use their rented resources. Cloud computing platforms simulations permit low costs analysis to effects about the resources management in these environments. This work investigates some computational resources management strategies, with focus on server consolidation (and its relation with variable size vector bin packing) and load prediction (which analyses the lasts demand measurements as a time series). It then analyses the effects in computational resources use and service level agreements fulfilling while different server consolidation heuristics and load prediction algoritms are used. A framework developed to simulate a Cloud Computing platform is then presented. The simulations ran based on a real-world workload publicly avaliable by Google, and showned up a huge influence of the load prediction method to the active servers, SLA violations, and other characteristcs.

Keywords: cloud computing, server consolidation, load prediction.

Sumário

Lista de figuras

Lista de tabelas

Capítulo 1

Introdução

A Computação em Nuvem agrega uma diversidade de tecnologias desenvolvidas ao longo das últimas décadas a fim de disponibilizar ao usuário doméstico um meio de alugar, no sentido mais amplo da palavra, computação. Muitos desafios impediram que centrais de processamento de dados, ou *datacenters*, fossem para um usuário doméstico o que o próprio nome propunha, de modo que outras empresas se tornaram seus principais clientes. Para ter acesso à computação, os usuários domésticos (e ainda muitas empresas) necessitavam adiquirir todos os recursos de que desejava usufruir: *hardware* capaz de processar dados, manipulá-los e persistí-los para uso posterior. Com o advento de tecnologias para acesso rápido à internet, gestão de recursos computacionais de modo automatizado e, principalmente, virtualização de recursos, o usuário doméstico passa a ter liberdade para alugar apenas a quantidade de computação de que necessita e quando necessita, com o custo inicial de ter algum dispositivo capaz de conectar à internet.

Assim, a Computação em Nuvem disponibiliza aos usuários uma liberdade sem precedentes no uso de recursos computacionais. Tal liberdade acompanha, porém, diversos desafios a serem enfrentados por parte dos provedores desses serviços [ZCB10]. Dentre eles, pode-se destacar o provisionamento automatizado de recursos, a consolidação de servidores e o gerenciamento de energia, intimamente relacionados à pesquisa conduzida nesta dissertação.

A importância de uma boa gestão dos recursos computacionais transcende os limites do escopo computacional: além de prover melhor qualidade do serviço, uma boa gestão desses recursos pode diminuir o custo operacional para manter a infraestrutura. No contexto de Computação em Nuvem, os serviços e tarefas são executados em ambientes com recursos virtualizados, ou seja, sobre uma camada de abstração para acesso aos recursos. Este trabalho lida com situações em que se virtualiza o acesso a recursos computacionais, como processamento, memória e rede, contexto no qual a consolidação de servidores recebe especial atenção para lidar com a gestão desses recursos, visto que é um dos componentes responsáveis por definir que servidores físicos irão prover os recursos demandados por cada máquina virtual.

Os usuários, os quais podem ser humanos ou mesmo serviços e aplicações automatizados, são responsáveis por informar a quantidade de cada recurso computacional que se deseja provisionar para seus serviços ou aplicações. Tais dados informados pelos usuários costumam ser estimativas, e comumente levam em consideração uma "folga" para que seus serviços não sejam prejudicados em um possível momento em que necessitem de mais recursos. Esse superprovisionamento aplicado a cada um dos ambientes virtualizados acaba por aumentar consideravelmente a quantidade de recursos alocados, porém ociosos. Esta situação pode ser observada em análise sobre um traço de execução disponibilizado pelo Google, o qual é melhor explorado na Seção 4.2: em geral, mais de 80% da capacidade de memória das máquinas físicas foi alocada à execução de aplicações, e, em alguns momentos, mais de 100% da capacidade de CPU estava alocada, em relação à quantidade de recursos requisitados. O nível de utilização desses recursos, porém, não excediam 50% ou 60% da capacidade de memória e CPU [RTG+12], respectivamente.

A fim de diminuir a quantidade de servidores ligados e aumentar a utilização dos recursos ativos, pode ser tentador optar por alocar mais ambientes virtualizados em cada servidor. Porém os efeitos

colaterais dessa decisão trazem novos desafios à discussão:

Imprevisibilidade da demanda de recursos Plataformas de Computação em Nuvem são ambientes multi-inquilinos[1], e cada um dos usuários tem a liberdade para definir como os recursos alugados serão utilizados. Como exemplo, enquanto alguns usuários optam por disponibilizar um serviço *web* com utilização esporádica, outros podem optar por executar aplicações com uso intensivo de processamento. Essa característica dificulta a previsibilidade da quantidade de recursos que serão utilizados por cada ambiente virtualizado em um momento futuro.

Respeito a acordos de nível de serviço A disponibilização dos recursos computacionais aos usuários está sujeita a acordos de nível de serviço (*SLA – Service Level Agreement*), cujos termos podem conter garantias sobre a disponibilidade dos recursos, sob pena de multas em caso de descumprimento. A decisão de alocar mais recursos do que os disponíveis nos servidores pode levar a provedora do serviço a infringir diversos acordos de nível de serviço, caso, hipoteticamente, todas as aplicações nesse servidor necessitem usar, ao mesmo tempo, 100% dos recursos alugados.

A alocação de recursos nesses ambientes não é realizada de modo estático, a qualquer momento usuários podem requerer recursos para mais ambientes virtualizados, ou mesmo liberar os recursos de ambientes virtualizados que não lhes serão mais úteis. Nesse ambiente dinâmico, a migração em tempo real de máquinas virtuais torna-se uma importante aliada na gestão dos recursos computacionais. Com tal tecnologia, uma máquina virtual pode passar a ser executada em um outro servidor, com um intervalo de indisponibilidade que pode chegar a apenas 60 ms [CFH+05].

Dadas as tecnologias de consolidação de servidores e migração de máquinas virtuais em tempo real, em conjunto com técnicas de predição de desempenho, este trabalho pretende explorar como a predição de desempenho pode auxiliar a estratégia de consolidação de servidores a utilizar menos servidores para prover os recursos demandados pelos ambientes virtualizados, sabendo que se houver predição de aumento no uso de recursos, alguns ambientes virtualizados poderão ser migrados para servidores mais ociosos.

1.1 Contribuições

O desenvolvimento do presente trabalho contribui ao disponibilizar à comunidade uma ferramenta de simulação de plataformas de Computação em Nuvem, uma heurística para alocação de máquinas virtuais, e a análise de vantagens e desvantagens ao empregar diferentes estratégias para gerenciar recursos computacionais em plataformas de Computação em Nuvem:

- Arcabouço para simulação de plataformas de Computação em Nuvem. Foi desenvolvido o SimMyCloud [AC14], um arcabouço no qual podem ser implementadas novas estratégias para gestão dos recursos computacionais em uma plataforma de Computação em Nuvem, e os efeitos ao empregá-las em um traço de execução fornecido, real ou artificial, podem ser comparados.

- Heurística *Grouping VSVBP* para lidar com o problema de alocação de itens em cestos de tamanho variável – *Variable Size Vector Bin Packing*. Foi realizada adaptação da heurística de alocação de itens *Fast Vector Packing algorithm* [DKJ11] para lidar com cestos de diferentes tamanhos em cada dimensão.

- Comparação dos efeitos sobre eficiência no uso dos recursos computacionais e violações de acordos de nível de serviço ao empregar diferentes estratégias de gerenciamento de recursos em plataformas de Computação em Nuvem. Foram analisados os efeitos ao utilizar as estratégias

[1] Ambientes multi-inquilinos (ou ambientes *multitenant*) são caracterizados por ter uma aplicação, plataforma, sistema operacional, ou outro tipo de recurso, compartilhado entre diversos usuários, de modo que possíveis falhas em um usuário não interfiram na disponibilidade de recursos aos demais.

de consolidação de servidores *FirstFit*, *Grouping VSVBP* e *Slim BFD*, além das estratégias de predição de desempenho *LastFiveMeasurements* e *RBFPrediction*, sob um traço de execução real disponibilizado pelo Google.

1.2 Organização do trabalho

O Capítulo 2 apresenta alguns conceitos relacionados à gestão de recursos em plataformas de Computação em Nuvem, com ênfase naqueles mais relacionados ao problema e soluções propostas neste trabalho. Heurísticas e estratégias para gerenciamento de recursos computacionais em plataformas de Computação em Nuvem são apresentadas no Capítulo 3, enquanto o Capítulo 4 apresenta o arcabouço de simulação desenvolvido e um traço de execução de uma plataforma real de Computação em Nuvem. As descrições dos experimentos executados e análises sobre os mesmos estão presentes no Capítulo 5. O Capítulo 6 lista as conclusões sobre os resultados alcançados.

Capítulo 2

Conceitos

A gestão de recursos computacionais em plataformas de Computação em Nuvem enfrenta desafios em diversos pontos em que diferentes estratégias podem ser utilizadas para se tomar uma decisão. Ao ligar um servidor físico, por exemplo, diferentes estratégias podem levar em consideração a posição do servidor no ambiente físico, a quantidade de recursos disponíveis, ou mesmo a arquitetura desse servidor, e cada estratégia utilizar critérios diferentes para definir qual servidor será ligado.

Nos experimentos deste trabalho, busca-se realizar predições de desempenho das máquinas virtuais em uma plataforma de Computação em Nuvem, de modo a selecionar, com base nos valores preditos, em quais servidores físicos tais máquinas serão alocadas. Espera-se que na alocação resultante os servidores consigam prover os recursos necessários para execução das máquinas virtuais nele alocadas, sem que servidores sejam sobrecarregados.

Assim, explora-se neste capítulo o conceito de Computação em Nuvem, contexto no qual a pesquisa foi realizada; acordos de nível de serviço, contratos que regulamentam o aluguel de recursos nessas plataformas; virtualização de recursos, tecnologia empregada para se concretizar parte das ideias do conceito de Computação em Nuvem; consolidação de servidores, que lida com o problema de alocar as máquinas virtuais em servidores físicos; e a predição de desempenho, que busca prever quanto de recursos cada máquina virtual deve utilizar em um período de tempo futuro.

2.1 Computação em Nuvem

Computação em Nuvem (ou *Cloud Computing*) é o termo utilizado para referenciar tanto aplicações disponibilizadas como serviços pela rede quanto o *hardware* e sistemas nos *datacenters* que provêm tais serviços [AFG+09]. A infraestrutura para provê-los permite acesso sob demanda a um conjunto compartilhado de recursos (servidores, armazenamento e aplicações, por exemplo) configuráveis, que podem ser rapidamente alocados e liberados com mínimo esforço de gerenciamento ou interação com o provedor de serviço [MG09].

A definição de Computação em Nuvem pelo NIST – *National Institute of Standards and Technology* – aponta cinco características intrínsecas desse modelo:

Utilização sob demanda O usuário pode provisionar capacidades de computação, tais como tempo para execução e uso de memória, da maneira que julgar necessário. Tais recursos serão disponibilizados automaticamente, sem intervenção humana em cada provedor de serviço.

Acesso pela rede Os recursos estão disponíveis pela rede e acessíveis por mecanismos padrões, permitindo uso a partir de diferentes plataformas (como celulares, *laptops* e estações de trabalho).

Compartilhamento de recursos Os recursos computacionais do provedor são compartilhados entre múltiplos usuários, com recursos físicos e virtuais atribuídos de acordo com a demanda de cada usuário.

Elasticidade A alocação de recursos ocorre sob demanda em tempo de execução, de modo que a capacidade disponível a cada usuário pode aumentar ou diminuir rapidamente em tempo de execução, o que dá ao usuário a impressão de que os recursos disponíveis para utilização são infinitos.

Serviço mensurável O controle e otimização dos recursos ocorre de modo automatizado, com base em métricas adequadas ao tipo de serviço oferecido (como armazenamento, processamento e quantidade de usuários ativos).

Com tais características básicas [MG09], os provedores optam por disponibilizar o serviço em três modelos básicos, não excludentes: *Software as a Service (SaaS)*, onde o provedor permite ao usuário o uso de aplicações que executam em sua infraestrutura; *Platform as a Service (PaaS)*, onde o provedor permite que o usuário crie e disponibilize suas próprias aplicações, utilizando como base linguagens de programação, bibliotecas, serviços e ferramentas definidas pelo provedor; e *Infrastructure as a Service (IaaS)*, onde o provedor disponibiliza ao usuário um ambiente em que o mesmo pode alugar recursos computacionais e disponibilizar e executar sistemas arbitrários, desde sistemas operacionais a aplicações.

A contratação de um serviço de Computação em Nuvem implica em responsabilidades tanto por parte do provedor do serviço quanto por parte do contratante. Tais responsabilidades são descritas acordos de nível de serviço, firmados no momento da contratação do mesmo.

2.2 Acordos de nível de serviço

Ao contratar um serviço de Computação em Nuvem, ficam acordadas em contrato responsabilidades tanto sobre o uso quanto sobre a disponibilização dos recursos. Dentre as restrições para os usuários, podem ser definidos limites sobre o teor dos conteúdos disponibilizados, de modo a evitar, por exemplo, que se infrinja leis aplicáveis no país onde os servidores se encontram. Em relação a restrições aplicadas aos provedores, pode-se citar a definição de um tempo máximo de resposta dos servidores; a garantia de que, a qualquer momento, o usuário poderá utilizar a quantidade máxima dos recursos alugados; a garantia de um tempo máximo para resolução de problemas; ou ainda a disponibilidade mínima do serviço, onde o provedor deve garantir que o serviço estará disponível em, por exemplo, 99,9% do tempo em um período de 30 dias. Tais restrições compõem um acordo de nível de serviço, ou *SLA – Service Level Agreement –*, cujo objetivo é definir as responsabilidades de cada uma das partes envolvidas no contrato. Ainda que a maioria dos exemplos tenham sido qualitativos, recomenda-se quantificar as restrições ao elaborar tais acordos.

A infraestrutura para prover Computação em Nuvem é o conjunto do *hardware* e *software* que viabiliza as cinco características básicas desse modelo. A nuvem pode ser vista como composição de duas camadas: uma física e uma de abstração. A física é composta pelos recursos de *hardware* necessários para comportar os serviços providos, e geralmente inclui componentes de servidores, armazenamento e rede. A de abstração consiste no sistema implantado na camada física, de modo a prover as características básicas do modelo. Essa estratégia para viabilizar uma plataforma que cumpra com os requisitos básicos de plataformas de Computação em Nuvem utiliza a técnica de virtualização de recursos, explorada na próxima seção.

2.3 Virtualização de recursos

A virtualização de recursos é uma técnica utilizada para criar uma camada de abstração entre a aplicação e os recursos que ela utiliza, podendo este ser físico (o próprio *hardware*), um sistema operacional, um dispositivo, ou mesmo outras aplicações. As figuras 2.1 e 2.2 ilustram os elementos envolvidos no acesso aos recursos sem e com o uso de uma camada de virtualização, respectivamente. Essa camada de acesso indireto aos recursos tem sido bastante utilizada para solucionar problemas de segurança, otimização de desempenho, administração de sistemas, entre outros desafios na área

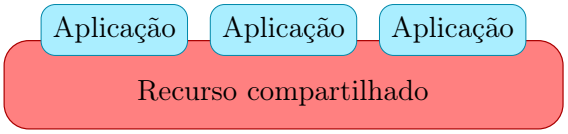


Figura 2.1: *Elementos em um ambiente com acesso direto aos recursos*

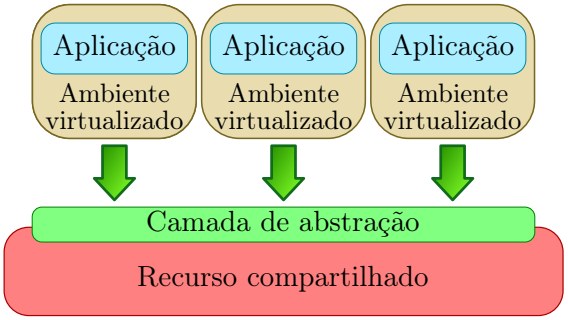


Figura 2.2: *Elementos em um ambiente de virtualização*

de computação [FDF05]. Tal abstração é o componente principal para a elasticidade do ambiente de Computação em Nuvem, visto que os recursos disponibilizados para cada ambiente podem ser alterados durante a execução dos mesmos.

Em plataformas *IaaS*, onde se compartilha recursos de servidores físicos, máquinas virtuais são comumente empregadas como estratégia para virtualização, as quais simulam os recursos computacionais de uma máquina real e tornam o ambiente virtualizado criado independente de outras máquinas virtuais e do *hardware*. Apesar de ambientes virtualizados poderem ser concretizados em entidades diferentes de acordo com o tipo de recurso que se está virtualizando (como, por exemplo, um ambiente criado para uma aplicação executar em modo de compatibilidade[1]), a presente pesquisa lida com a virtualização no contexto de compartilhamento de recursos computacionais em servidores físicos, de modo que os termos "ambiente virtualizado" e "máquina virtual" são utilizados com a mesma semântica ao longo desta dissertação.

Um componente chave da Computação em Nuvem é o gerenciamento da infraestrutura de virtualização [SMLF09], cujo sistema, chamado de hipervisor (ou *hypervisor*), deve gerenciar o ciclo de vida das máquinas virtuais e permitir uma alocação de recursos configurável. Alguns desses sistemas [euc14, ope14, ovi14] tem como funcionalidades a alocação dinâmica de máquinas virtuais, o balanceamento de carga automático, a consolidação de servidores e o redimensionamento dinâmico da infraestrutura, de modo a garantir que os objetivos supracitados sejam alcançados. Para realizar balanceamento de carga automático, faz-se uso da técnica de migração de máquinas virtuais em tempo real [CFH+05], a qual permite migrar uma máquina virtual de um servidor físico para outro, com curto período de indisponibilidade da aplicação.

Ao isolar cada aplicação em um ambiente virtualizado, permite-se que dois ou mais ambientes virtualizados compartilhem recursos de um mesmo servidor físico. A principal técnica utilizada para definir quais servidores físicos irão prover recursos para cada máquina virtual é a consolidação de servidores. A consolidação de servidores quebra a relação $1:1$ entre máquinas virtuais e servidores físicos, de forma a diminuir quantidade de servidores ativos necessários e, indiretamente, diminuir custos com energia, refrigeração, aquisição de equipamentos e outros.

[1] Alguns sistemas operacionais utilizam este recurso para permitir que um aplicativo planejado para ser executado em uma versão anterior do sistema operacional possa ser executada na versão mais recente, caso a API do sistema tenha sido alterada, por exemplo.

2.4 Consolidação de servidores

A tecnologia de virtualização tem sido utilizada há décadas [Vog08], e permite que ambientes isolados compartilhem os mesmos recursos computacionais de modo que cada um os perceba como se fosse "dono" do *hardware* sobre o qual é executado. Nesse contexto, tem-se o desafio de definir que servidor físico irá prover os recursos para cada ambiente virtualizado. Tal desafio incentivou o desenvolvimento de pesquisas na temática de consolidação de servidores, que consiste na multiplexação de recursos de *hardware* para diversos ambientes virtualizados, ou seja, permite a alocação de vários desses ambientes em um único servidor físico, que irá prover seus recursos à execução de tais ambientes. Em plataformas de Computação em Nuvem, a decisão de que máquinas físicas proverão os recursos para cada ambiente virtualizado fica a cargo dos *hypervisors*, gerenciadores de ambientes virtualizados cujos algoritmos buscam utilizar a menor quantidade de máquinas físicas possível, paralelamente a outros objetivos, como manter a qualidade do serviço ou evitar violações de acordos de nível de serviço.

Diversos pesquisadores exploraram esse problema da alocação de ambientes virtualizados em máquinas físicas. Doddavula, Kaushik e Jain [DKJ11] criaram uma heurística que aplica algumas generalizações sobre a demanda de recursos, com o objetivo de ser rapidamente executada. A heurística, que é abordada com mais detalhes na Seção 3.1, agrupa os ambientes virtualizados até que os recursos demandados por cada grupo seja similar à quantidade de recursos disponíveis nas máquinas físicas. Panigrahy e outros pesquisadores [PTUW11] avaliaram heurísticas para o problema de *bin packing* vetorial, variações propostas para o algoritmo *First Fit Decreasing* (FFD) ao lidar com tamanhos em múltiplas dimensões. Esses trabalhos lidaram com o problema da alocação de máquinas virtuais com a premissa de que os servidores físicos possuíam a mesma quantidade de recursos.

Gabay e Zaourar [GZ13] exploraram o problema da realocação de máquinas virtuais, com foco em melhorar a distribuição do uso dos recursos computacionais ao longo da execução das aplicações. No percurso, apresentaram o problema de *bin packing* vetorial com cestos de tamanho variável em conjunto com duas versões da heurística *Best Fit Decreasing* (BFD) – *Bin Centric* e *Item Centric* –, além de sugerir meios de linearizar os tamanhos de itens e cestos. Tais linearizações transformam o tamanho de itens e cestos, que originalmente são representados em vetores multidimensionais, em tamanhos $x_i \in \mathbb{R}$, de modo a simplificar a comparação de tamanhos de itens e cestos. A Seção 3.3 apresenta detalhes da abordagem de Gabay e Zaourar.

A alocação de ambientes virtualizados em servidores físicos pode ocorrer de forma estática, situação na qual se conhece todos os ambientes virtualizados e os servidores estão vazios, e então aloca-se todos os ambientes apenas uma vez, ou de forma dinâmica, quando permite-se alocar máquinas virtuais em algum momento após a alocação inicial, ou seja, os servidores podem já estar fornecendo recursos a outros ambientes virtualizados.

Ferreto e outros pesquisadores [FNCDR11] lidaram com a alocação dinâmica de máquinas virtuais em conjunto com a migração de servidores, contexto no qual introduzem o conceito de controle de migração, ao adicionar no processo uma etapa para definir se, e quais, máquinas virtuais devem ser migradas. Na pesquisa, compararam a quantidade de migrações ocorridas em dois cenários: o primeiro permitia a realocação de todas as máquinas virtuais a cada 300 segundos (cinco minutos), considerando o uso de recursos de cada máquina virtual nesse intervalo subsequente; o segundo empregava um algoritmo de controle de migração, o qual mantinha nos mesmos servidores físicos as máquinas virtuais cujo uso de recursos não havia se alterado. O traço de execução que utilizaram foi disponibilizado pelo Google, e contém dados de uso de recursos de diversas tarefas em um período de sete horas [Hel10].

Definir quais servidores físicos devem fornecer recursos para cada máquina virtual é um desafio que pode ser analisado como o problema de *bin packing* vetorial com cestos de tamanho variável, conforme será tratado na próxima seção.

2.5 Variable Size Vector Bin Packing – VSVBP

O problema de alocar máquinas virtuais na menor quantidade de servidores físicos possível é computacionalmente equivalente a uma variação do clássico problema de *bin packing*. Na versão clássica do problema [MT90], tem-se n cestos de tamanho $c \in \mathbb{R}^+$, e a mesma quantidade de itens de tamanhos $s_i \in \mathbb{R}^+, i \in N = \{1, \ldots, n\}$, de modo que deseja-se

$$\begin{aligned}
&\text{minimizar} \quad z = \sum_{j=1}^{n} y_j \\
&\text{sujeito a} \quad \sum_{i=1}^{n} s_i x_{ij} \leq c y_j \quad ,\forall j \in N, \qquad (2.1)\\
&\qquad\qquad \sum_{j=1}^{n} x_{ij} = 1 \quad ,\forall i \in N, \qquad (2.2)\\
&\qquad\qquad y_j \in \{0,1\} \quad ,\forall j \in N,\\
&\qquad\qquad x_{ij} \in \{0,1\} \quad ,\forall i, \forall j \in N,
\end{aligned}$$

onde

$$y_j = \begin{cases} 1, \text{ se o cesto } j \text{ for usado, ou} \\ 0, \text{ caso contrário} \end{cases}$$

$$x_{ij} = \begin{cases} 1, \text{ se o item } i \text{ estiver alocado no cesto } j, \text{ ou} \\ 0, \text{ caso contrário} \end{cases}$$

Logo, deseja-se saber qual a quantidade mínima de cestos que devem ser utilizados para acomodar todos os itens apenas uma vez (Equação 2.2), sem que a soma dos tamanhos dos itens em um cesto seja superior a c (Equação 2.1). Esse problema é sabidamente NP-difícil [GJ79], indicando que não existe algoritmo polinomial capaz de encontrar a solução ótima do problema, a não ser que $P = NP$. Em termos práticos, isso implica que é difícil encontrar a solução ótima em tempo razoável quando se possui muitos itens. Essa dificuldade incentiva o uso de heurísticas para encontrar soluções relativamente boas, com a vantagem de necessitar de tempo de execução menor.

Ainda que o problema clássico de *bin packing* possa ser explorado quanto a heurísticas, propriedades e outros fatores, são necessárias algumas generalizações para que se possa considerar o problema da alocação de ambientes virtualizados na menor quantidade de servidores físicos possível como uma variação do problema de *bin packing*. Para alcançar a variação de *bin packing* vetorial com cestos de tamanho variável (tradução livre do original *Variable Size Vector Bin Packing* [GZ13], ou simplesmente VSVBP), o problema foi adaptado com as seguintes variações:

1. o número de cestos é finito;
2. os cestos podem possuir tamanhos variados; e
3. os itens e cestos são considerados vetores com a mesma quantidade de dimensões, sendo os valores em cada dimensão independentes dos demais.

O problema de *bin packing* vetorial com cestos de tamanho variável é então definido por: Dados os índices $\mathcal{I} = \{1, \ldots, N\}$ para os itens, $\mathcal{B} = \{1, \ldots, M\}$ para os cestos (*bins*) e $\mathcal{D} = \{1, \ldots, d\}$ para as dimensões, sejam S uma matriz $(\mathbb{R}^+)^{N \times d}$ de tamanhos de itens e C uma matriz $(\mathbb{R}^+)^{M \times d}$ de capacidades de cestos, com $s_i^j \in S$ sendo o tamanho do item $i \in \mathcal{I}$ na dimensão $j \in \mathcal{D}$, e $c_k^j \in C$ sendo a capacidade do cesto $k \in \mathcal{B}$ na dimensão $j \in \mathcal{D}$. O problema consiste em encontrar uma alocação viável $x \in \{0,1\}^{N \times M}$ de itens em cestos tal que:

$$\sum_{i \in I} s_i^j x_{i,k} \leq c_k^j \qquad \forall j \in \mathcal{D}, \forall k \in \mathcal{B} \tag{2.3}$$

$$\sum_{k \in B} x_{i,k} = 1 \qquad \forall i \in \mathcal{I} \tag{2.4}$$

$$x_{i,k} \in \{0,1\} \qquad \forall i \in \mathcal{I}, \forall k \in \mathcal{B}$$

sendo que $x_{i,k} = 1$, se o item i está no cesto k, ou $x_{i,k} = 0$, caso contrário.

A inequação 2.3 impõe restrição quanto à capacidade de cada cesto, cuja soma dos tamanhos dos itens nele alocados não deve ultrapassar a capacidade do cesto em qualquer dimensão, enquanto que a equação 2.4 garante que todos os itens serão alocados, e em apenas um cesto. A versão de decidibilidade desse problema define um número $Q \in \mathbb{N}$ e busca responder se há uma alocação viável x que use no máximo Q cestos, enquanto a versão de otimização busca a quantidade mínima de cestos que podem ser utilizados.

Para tornar o problema da alocação de ambientes virtualizados em máquinas físicas computacionalmente equivalente ao problema *Variable Size Vector Bin Packing*, considera-se cada ambiente virtualizado um item, com cada requisito de recurso computacional representado em uma dimensão, e cada máquina física um cesto, com a capacidade de cada recurso computacional representada na dimensão correspondente.

2.6 Predição de desempenho

Ao lidar com a alocação dos recursos computacionais em um provedor de serviços de Computação em Nuvem, precisa-se definir a quantidade de recursos que será alocada a cada ambiente virtualizado. Os recursos devem ser suficientes para a execução das aplicações em cada máquina virtual, e a alocação deve ocorrer de modo a evitar violações em acordos de nível de serviço, pois, como melhor explorado na Seção 2.2, tais violações podem incorrer em prejuízos para o provedor de serviços de Computação em Nuvem. Ao requisitar um ambiente virtualizado, os usuários necessitam informar a quantidade de recursos desejada para suas máquinas virtuais, a qual é, em geral, suficiente para executar as aplicações que deseja. Assim, de forma a garantir ao usuário que a quantidade de recursos requisitada estará disponível sempre que necessitar, uma primeira opção é alocar para cada cada ambiente a exata quantidade de recursos informada pelo usuário.

Liu [Liu11], ao medir a utilização de recursos em máquinas físicas de provedores públicos de serviços de Computação em Nuvem, mais especificamente Amazon EC2 [Ser] e GoGrid [GoG], identificou que apenas uma pequena quantidade de recursos são realmente utilizados. No período de uma semana, por exemplo, a média de uso de recursos em 20 servidores do serviço Amazon EC2 foi de 7,3%, sendo que durante toda a semana, a utilização de recursos em uma máquina individual esteve no intervalo de 3,6% a 16,9%. Em análise a um traço de execução disponibilizado pelo Google, o ClusterData2011_1 [RWH11], Reiss e outros [RTG+12] apontam que grande parte dos recursos requisitados pelos usuários ficam ociosos: no caso de recursos de memória, são utilizados aproximadamente 53% da quantidade de memória requisitada, e apenas 40% dos recursos de CPU requisitados. Com tais informações disponíveis, tem-se que a opção de alocar a cada ambiente a quantidade de recursos requisitada pelo respectivo usuário pode acarretar em uma grande quantidade de recursos ociosos em uma plataforma de Computação em Nuvem.

A fim de diminuir a porcentagem dos recursos computacionais ociosos, podemos alocar para as aplicações uma quantidade de recursos mais próxima da que realmente será utilizada. Para tal, busca-se predizer quanto de recursos cada aplicação utilizará em um intervalo de tempo subsequente ao momento da alocação, ou seja, busca-se realizar uma predição de desempenho das mesmas. Predições podem ser realizadas com base em modelos que descrevem o comportamento da aplicação, dados históricos, execuções anteriores, ou mesmo um misto dessas abordagens, além de poderem utilizar diversas variáveis para se computar o valor predito. Este trabalho atém-se ao caso em

que se utiliza das observações anteriores para prever o próximo valor nessa série, de modo que as mesmas são modeladas como uma série temporal. Uma série temporal é um conjunto de observações realizadas sequencialmente ao longo do tempo. Como considera-se que essas observações sejam dependentes, técnicas para lidar com esses dados permitem identificação de padrões e análises de sazonalidade, tendências, predições, entre outros. Em linhas gerais, o problema da predição de séries temporais segue a formulação:

dado	$X = \{x_1, x_2, \ldots, x_n\}$
encontrar	x'_{n+1}
que minimize	$\lvert x'_{n+1} - x_{n+1}\rvert$
onde	$x_i, x'_i \in \mathbb{R}, \forall i \in \mathbb{N}$
	x_i é a quantidade de recursos utilizada pelo ambiente no tempo i
	x'_i é a quantidade prevista de recursos que o ambiente utilizará no tempo i

No cenário ideal, tem-se um oráculo que prediz a quantidade exata de recursos que este ambiente irá utilizar no instante de tempo subsequente. Ainda que este cenário seja utópico, há na literatura diversas estratégias para lidar com este problema [CG14, HJLW14, Ish10], algumas delas requerem execução anterior das aplicações, a fim de definir um modelo de uso de recursos, outras utilizam apenas dados históricos.

Ishii [Ish10] aborda ambos os casos em sua tese. Com o objetivo de otimizar operações de leitura e escrita sobre dados distribuídos, o trabalho modelou o comportamento dessas operações em séries temporais, para posterior avaliação e classificação segundo seu processo gerador. Segundo Ishii, ao reconhecer o processo gerador de uma série é possível inferir a melhor técnica para sua modelagem, e, para tal, três aspectos importantes devem ser considerados nessa investigação: a estocasticidade, a linearidade e a estacionariedade da série temporal. A estocasticidade avalia se uma série possui comportamento determinístico, ou seja, recorrente e periódico ao longo do tempo, com possíveis variações em escala, ou comportamento estocástico, onde as observações dependem do estado anterior e de componentes aleatórios. A linearidade avalia se as observações são organizadas por uma combinação linear entre ocorrências passadas e ruídos. Séries não lineares são obtidas por combinações não lineares entre os mesmos aspectos, e são mais difíceis de estudar e modelar [Ish10]. A estacionariedade avalia se a série temporal é formada por um estado particular de equilíbrio estatístico, ao evoluir aleatoriamente no tempo em torno de uma média e variância constantes.

Na abordagem que requer uma execução prévia das aplicações, o momento atual da execução dos processos das aplicações é avaliado, e as similaridades com o histórico de informações monitoradas são verificadas. Por fim, define-se uma janela de possíveis operações futuras a serem executadas. Na abordagem que utiliza predição automática e *online* de operações em arquivos, foram empregadas técnicas de extração de conhecimento, predição e janela deslizante adaptativa. A etapa de extração de conhecimento visa obter um modelo que melhor representa o processo gerador da série, o qual será utilizado como base para selecionar que técnica de predição será empregada. A Tabela 2.1 apresenta a relação entre a classificação dos processos geradores e as técnicas de predição para uso em cada caso. A janela deslizante define a quantidade de eventos futuros que a heurística deve levar em consideração para tormar decisões de otimização.

Para diminuir a porcentagem de recursos ociosos em infraestruturas de Computação em Nuvem, Caglar e Gokhale [CG14] propuseram um método de sobrealocação de recursos (*resource-overbooking*), o qual prediz, com base no histórico de uso dos recursos em cada máquina física, coeficientes de sobrealocação para cada máquina física na infraestrutura. Esse processo se divide em três etapas: predição de uso de recursos, predição de coeficientes de sobrealocação e uso de um assessor de desempenho, sendo este último o responsável por ajustes na predição, de modo a evitar violações de acordos de nível de serviço.

Como estratégia para a predição de uso dos recursos computacionais, Caglar e Gokhale utilizaram uma rede neural denominada perceptron de múltiplas camadas (ou MLP – *Multilayer Per-*

Classificação	Técnicas de predição
Determinística	FNN, AMI, Reconstrução Espaço Fase
Estocástica → Não linear	RBF, Polynom, Filtros
Estocástica → Linear → Estacionária	AR, ARMA
Estocástica → Linear → Não estacionária	ARIMA

Tabela 2.1: *Técnicas de predição que podem ser adotadas com base na classificação do processo gerador de uma série temporal, segundo Ishii [Ish10]*

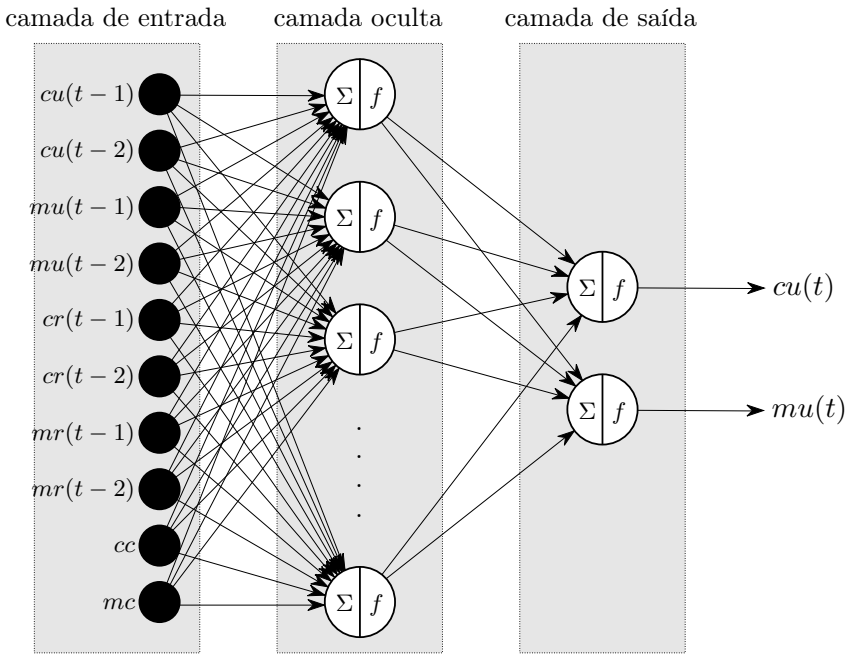


Figura 2.3: *Estrutura da rede neural artificial para predição do uso de recursos (adaptado de [CG14])*

ceptron), uma rede neural artificial sem realimentação com uma camada de entrada, uma camada de saída, e uma ou mais camadas ocultas. No trabalho, os pesquisadores optaram por utilizar uma única camada oculta, com 22 neurônios nessa camada[2], de forma que a arquitetura resultante pode ser visualizada na Figura 2.3 (a semântica das funções pode ser conferida na Tabela 2.2). As saídas da rede neural são os usos de CPU e memória preditos para o próximo intervalo de tempo, enquanto CPU e memória requisitados e a média de uso de recursos de CPU e memória nos dois intervalos de tempo imediatamente anteriores, em conjunto com a capacidade desses recursos nas máquinas físicas são os parâmetros da camada de entrada da rede neural.

Na pequisa, utilizaram as primeiras 695 horas do traço de execução do Google [RWH11] para treinamento da rede neural, de forma a utilizar a 696ª hora do traço para verificar a eficácia do modelo resultante. A função de ativação f dos neurônios na camada oculta é tangente sigmoidal, e o algoritmo de Levenberg-Marquardt foi empregado para aprendizagem por retropropagação, na etapa de treinamento. A estratégia da dupla alcançou os resultados esperados, de modo que identificaram a factibilidade de usar coeficientes de sobrealocação maiores que as identificadas no traço de execução do Google, sem, para isso, violar acordos de nível de serviço.

[2] essa quantidade foi definida pelos autores empiricamente, por apresentar melhores resultados em seus experimentos

Função/Variável	Semântica
t	hora da predição
$cu(t)$	uso médio de CPU na hora t
$mu(t)$	uso médio de memória na hora t
$cr(t)$	recurso de CPU requisitado na hora t
$mr(t)$	recurso de memória requisitado na hora t
cc	capacidade de CPU da máquina física
mc	capacidade de memória da máquina física

Tabela 2.2: *Semântica das funções e variáveis da rede neural artificial presentes na Figura 2.3*

2.6.1 Predição com função de base radial

A pesquisa de Ishii [Ish10], apresentada na Seção 2.6, explora alguns métodos para predição de valores em séries temporais, recomendando-os de acordo com a classificação do processo gerador de uma série. Dentre estes, o método de predição que usa a técnica de função de base radial (ou RBF – *Radial Basis Function*) é um dos indicados como opção caso o processo gerador da série temporal seja estocástico e não-linear.

Essa estratégia é utilizada para predição em sistemas dinâmicos. Um sistema dinâmico é composto por um conjunto de estados e por uma regra que determina o estado atual do mesmo, com base nos estados passados. Séries temporais unidimensionais podem ser possíveis saídas de sistemas dinâmicos. Algumas técnicas são utilizadas para reconstruir ou transformar a saída desses sistemas de modo que a série temporal $x_1, x_2, \ldots, x_n$ possa ser reconstruída em espaço multidimensional $x_n(m, \tau) = (x_n, x_{n+\tau}, \ldots, x_{n+(m-1)\tau})$, no qual m é a dimensão embutida e τ representa a dimensão de separação (atraso, ou *time delay*) [Ish10]. O espaço fase resultante considera somente relações entre estados (observações ou conjuntos de observações de uma série temporal), e, sendo o componente tempo removido, regressões podem ser realizadas a fim de compreender tendências de uma série temporal reconstruída nesse espaço fase.

Função de base radial é uma abordagem matemática para aproximar funções em espaço multidimensional [Ish10], utilizada neste trabalho com o objetivo de encontrar uma superfície que forneça ajuste adequado para as observações de uma série. São necessárias duas etapas para utilizar o modelo RBF: treinamento, no qual as observações passam a ser representadas na forma de funções radiais φ_i, com pesos w_i; e execução, na qual a função de base radial é utilizada para predizer novas observações com base no comportamento modelado da série de entrada.

O capítulo seguinte apresenta métodos de consolidação de servidores, que serão utilizados em conjunto com a predição de desempenho.

Capítulo 3

Heurísticas para consolidação de servidores

O problema da consolidação de servidores, descrito com mais detalhes na Seção 2.4, consiste em alocar diversos ambientes virtualizados em servidores físicos. Essa alocação pode ocorrer de variadas formas, e levar em consideração fatores como consumo de energia, arquitetura das máquinas, perfil de execução dos ambientes virtualizados, topologia de redes, entre outros. O objetivo das heurísticas exploradas neste trabalho é minimizar a quantidade de servidores ativos necessários para execução dos ambientes virtualizados.

Esse problema, como descrito na Seção 2.5, é computacionalmente equivalente ao problema de *Variable Size Vector Bin Packing* na versão de otimização, em que se busca alocar todos os itens (ambientes virtualizados), com uso da menor quantidade de cestos (servidores físicos) possível. Como o problema é NP-difícil e as plataformas contam com grande quantidade de máquinas virtuais, o uso de heurísticas torna-se um forte atrativo para encontrar alocações viáveis em tempo hábil.

Este capítulo explora uma heurística para o problema da alocação de itens multidimensionais em cestos de tamanho fixo, a *Fast Vector Packing algorithm*, a qual foi adaptada para uso no problema de alocação de itens com cestos de tamanho variável – VSVBP. Essa adaptação resultou na heurística *Grouping VSVBP*, que, em conjunto com as heurísticas *FirstFit*, *Best Fit Decreasing* e *Slim BFD*, é utilizada para resolver o problema de *bin packing* com cestos de tamanho variável.

3.1 Fast Vector Packing algorithm

Doddavula e outros [DKJ11] investigaram o problema de consolidação de servidores como o problema de *Vector Packing*, mas a estratégia proposta permite seu uso para o problema de *Vector Bin Packing*[1], que consiste na busca pela alocação de itens na menor quantidade de cestos, sendo que os tamanhos de itens e cestos são vetores multidimensionais e todos os cestos têm, em cada dimensão, tamanho 1. Os autores utilizaram uma estratégia de agrupamento de máquinas virtuais com base na demanda de recursos, a fim de alcançar um rápido tempo de execução. Doddavula e os demais pesquisadores envolvidos no projeto dessa heurística identificaram que tal decisão poderia acarretar em soluções que necessitassem de mais servidores físicos, e deixaram a cargo do utilizador definir a granularidade da demanda de recursos para realizar os agrupamentos: quanto menor a granularidade, maior o tempo de execução, mas provavelmente menor será a quantidade de servidores físicos necessários.

O algoritmo consiste em quatro etapas:

- Criação de um modelo de classificação de magnitude – Nessa etapa são definidas as classificações da magnitude do uso de recursos. Tais classes podem seguir um intervalo uniforme de

[1] O problema de *Vector Packing* lida com a alocação de n itens de m tipos diferentes [Bra]. Porém, o algoritmo proposto pelos autores não leva essa restrição em consideração, de modo que pode-se considerar $m = n$ sem causar efeitos colaterais. Assim, o algoritmo pode ser utilizado para o problema de *Vector Bin Packing*.

distribuição (como 0 a 10, 10 a 20, 20 a 30, etc) ou não (como 0 a 5, 5 a 15, 15 a 20, etc). Um exemplo de classificação pode ser verificado na Tabela 3.1.

- Mapeamento de estados – Definido o modelo, nessa etapa mapeia-se o uso de recursos de cada máquina virtual para uma das classes. Tal mapeamento é realizado independentemente para cada dimensão, de modo que ao final dessa etapa cada máquina virtual é representada como um *workload*, que possui uma combinação de classes (por exemplo, FF, FL, etc) associada.

- Formação da matriz de estados – Uma matriz é criada para agrupar os *workloads* de uma mesma combinação de classes. O tamanho da matriz é definido pelo número de combinações possíveis, ou seja, C^D, onde $C \in \mathbb{N}$ é a quantidade de classes e $D \in \mathbb{N}$ é a quantidade de dimensões, caso haja a mesma quantidade de classes em todas as dimensões. Ao considerar, por exemplo, duas dimensões no exemplo de modelo de classificação da Tabela 3.1, temos a seguinte matriz de estados, de ordem 4:

FF	FL	FM	FS
LF	LL	LM	LS
MF	ML	MM	MS
SF	SL	SM	SS

  Criada a matriz, adicionamos os *workloads* nos grupos correspondentes.

- Consolidação de servidores – A alocação dos *workloads*, e respectivas máquinas virtuais, nos servidores se dá com base nas seguintes regras:

  1. Aloque cada *workload* com a maior quantidade de recursos (F, no caso do exemplo da Tabela 3.1) em um único servidor, visto que, por utilizar grande quantidade de pelo menos um recurso, não poderão ser combinados com nenhum outro *workload*;
  2. Combine os *workloads* restantes da seguinte forma: Comece com *workloads* na posição (i, j) da matriz, combinando-os com *workloads* em $(n - i, n - j)$, com i e j variando de 0 a n independentemente. Caso haja mais *workloads* em (i, j) que em $(n - i, n - j)$, combine-os com aqueles em $(n - i + a, n - j + b)$, com a e b variando de 0 a i e de 0 a j, respectivamente.
  3. Atualize a classe dos *workloads* resultantes das combinações realizadas. No caso do exemplo da Tabela 3.1, as combinações possíveis e as classes resultantes podem ser vistas na Tabela 3.2; e
  4. Repita as regras de alocação, combinação e atualização das classes resultantes até que todos os *workloads* tenham a maior classe possível. Quando não for mais possível realizar combinações, aloque cada *workload* restante em um servidor.

Estados	Classes	Intervalos (%)
Full	F	75 a 100
Large	L	50 a 75
Medium	M	25 a 50
Small	S	0 a 25

Tabela 3.1: *Exemplo de modelo de classificação de magnitude para uso na heurística* Fast Vector Packing algorithm

Os autores explicitam que o balanço entre soluções que utilizem poucas máquinas físicas e soluções com rápida execução é realizado pela definição do modelo de classificação. Quanto mais classificações, maior a probabilidade de obter soluções com menor quantidade de servidores, porém, maior o tempo necessário para executar o algoritmo.

Classes	Possibilidades de combinações	Classes resultantes
F	Não é possível combinar	
L	ao combinar com S	L + S = F
M	ao combinar com M	M + M = F
	ao combinar com S	M + S = L
S	ao combinar com L	S + L = F
	ao combinar com M	S + M = L
	ao combinar com S	S + S = M

Tabela 3.2: *Possíveis combinações de* workloads *e classes resultantes dessas combinações, com base no modelo de classificação de magnitude da Tabela 3.1*

3.2 First Fit

A heurística *First Fit* para o problema de *bin packing* vetorial com cestos de tamanho variável segue o mesmo princípio da heurística básica para o problema clássico de *bin packing*. Dados os itens a serem alocados, e os cestos nos quais os itens devem ser alocados, a heurística opta por alocar cada item no primeiro cesto com capacidade suficiente para alocá-lo.

Com o uso dessa heurística, os resultados obtidos podem ser bastante variados dependendo da ordem com que os itens são processados. Uma variação que diminui essa suscetiblidade a variações pela ordem de entrada dos dados é a heurística *First Fit Decreasing*, que realiza um pré-processamento na lista de itens a serem alocados, ordenando-os de forma decrescente.

3.3 Best Fit Decreasing

A heurística *Best Fit Decreasing* busca alocar os itens nos cestos em que melhor "se encaixam", ou seja, em cada alocação realizada, seleciona o item e cesto de modo que o tamanho restante no cesto seja o menor possível.

Na versão clássica do problema de *bin packing*, lida-se com tamanhos em apenas uma dimensão, o que torna o problema da comparação de tamanhos trivial. Ao lidar com mais dimensões, precisa-se definir uma regra que ajude a identificar, por exemplo, que tamanho é maior: $(2, 1)$ ou $(1, 2)$, sendo o valor em cada posição das tuplas referente a uma dimensão.

Gabay e Zaourar [GZ13] propõem o uso de coeficientes vetoriais e multiplicação escalar de vetores a fim de realizar essa linearização. Sendo α e β os vetores coeficientes para tamanhos de itens e cestos, respectivamente, e D a quantidade de dimensões, o tamanho de um item i dado por $s(i) = (s_1^i, s_2^i, \ldots, s_D^i)$ passa a ser $s'(i) = \alpha \cdot s(i) = \sum_d^{1 \to D} \alpha_d \cdot s_d^i$. Situação similar ocorre com um cesto j de tamanho $c(j) = (c_1^j, \ldots, c_D^j)$, substituindo α por β, ou seja, $c'(j) = \beta \cdot c(j) = \sum_d^{1 \to D} \alpha_d \cdot c_d^j$.

Ao considerar uma lista de itens $\mathcal{I}$, uma lista de cestos $\mathcal{B}$ e $r(j) \in (\mathbb{R}^+)^D$ a capacidade ainda não utilizada em cada cesto $j \in \mathcal{B}$, Gabay e Zaourar sugerem o uso de um dos três coeficientes abaixo para α e/ou β:

- $\frac{1}{R}$, onde $R = \sum_{i \in \mathcal{I}} s(i)$ é o vetor soma dos tamanhos dos itens;
- $\frac{1}{C}$, onde $C = \sum_{j \in \mathcal{B}} r(j)$ é o vetor soma das capacidades ainda disponíveis nos cestos; ou
- $\frac{R}{C}$.

A escolha de $\frac{1}{R}$ como coeficiente aumenta a relevância[2] dos recursos com maior demanda. Optar pelo uso de $\frac{1}{C}$ como o coeficiente aumenta a relevância dos recursos mais escassos. Já o uso do coeficiente $\frac{R}{C}$ define a relevância com base na raridade dos recursos, onde o recurso mais raro (com

[2]Os recursos com maior relevância terão maior peso ao definir o tamanho linearizado de um item ou cesto. Por exemplo: sejam $x_1 = (1, 0)$ e $x_2 = (0, 1)$ os tamanhos de dois itens. Se o segundo recurso tem maior relevância, então sobre os tamanhos linearizados $x'_1 \in \mathbb{R}$ e $x'_2 \in \mathbb{R}$ dos itens 1 e 2, podemos afirmar que $x'_2 > x'_1$.

maior proporção demanda/capacidade) tem maior relevância no tamanho linearizado do item ou cesto. Vale frisar que a cada item alocado, ou a cada cesto ignorado (que não será mais considerado para alocar um item), a quantidade de recursos em oferta ou demanda é alterada, o que leva à necessidade de recalcular os coeficientes afetados.

Definido o modo como os tamanhos serão linearizados, há duas estratégias para uso do *Best Fit Decreasing (BFD)*:

BFD Item Centric estratégia com foco na alocação de itens, em que se busca alocar um item por vez. Enquanto houver itens a serem alocados, seleciona-se o maior deles, e então busca-se alocá-lo no cesto em que melhor se encaixa.

BFD Bin Centric estratégia com foco no preenchimento dos cestos. Enquanto a lista de cestos não estiver vazia, o menor deles é selecionado, e então busca-se alocar nele os itens que melhor se encaixam. Quando nenhum outro item puder ser alocado neste cesto, o mesmo é retirado da lista e o ciclo de seleção de cesto e alocação se repete. Vale notar que nessa estratégia deve ser definida a quantidade máxima de cestos a serem utilizados.

3.4 Slim Best Fit Decreasing

A heurística *Slim Best Fit Decreasing*, ou *Slim BFD*, foi concebida durante a execução do presente trabalho, resultante de uma adaptação do *Best Fit Decreasing*, estratégia *BFD Item Centric*, apresentado na Seção 3.3. O objetivo era aproximá-lo da possibilidade de uso em um ambiente que requer baixo tempo de execução, visto que na heurística original muitos cálculos eram realizados após cada alocação.

A fim de alcançar os objetivos propostos, o cálculo dos tamanhos de itens e cestos ocorre apenas uma vez, antes de iniciar as alocações. No cálculo dos coeficientes $\frac{1}{C}$ e $\frac{R}{C}$, considera-se como recursos disponíveis os recursos disponíveis em todos os cestos, tanto aqueles em uso quanto os demais disponíveis, ainda que não estejam em uso. Logo, para uso desta estratégia torna-se necessário definir quantidade de cestos disponíveis.

O Algoritmo 1 apresenta o funcionamento dessa heurística.

Algoritmo 1 *Heurística Slim BFD*

```
1: função SlimBFD(𝓘, 𝓑)
2:     calcula os coeficientes α e β
3:     S ← tamanhos dos itens
4:     C ← tamanhos dos cestos
5:     enquanto há itens em 𝓘 faça
6:         x ← maior item em 𝓘
7:         aloca x no menor cesto viável em 𝓑
8:         remove x da lista de itens 𝓘
9:     fim enquanto
10: fim função
```

3.5 Grouping VSVBP

A heurística *Fast Vector Packing algorithm*, apresentada na Seção 3.1, foi adaptada neste trabalho para lidar com cestos de tamanho variável. A heurística resultante, nomeada *Grouping VSVBP*, permite utilizar a estratégia tanto para alocação inicial com cestos de tamanho variado quanto para alocação dinâmica, onde, no início da execução da heurística, já há itens alocados em servidores. Das adaptações realizadas, pode-se destacar:

- Os servidores passam a ser agrupados de acordo com a quantidade de recursos disponíveis; e

- Quando em uma iteração não se faz nenhuma alocação usando uma classe de recursos (por exemplo, F), passa-se a tentar alocar grupos de itens em cestos da classe de recursos imediatamente inferior (por exemplo, L).

Na etapa de criação do modelo de classificação de magnitude, define-se uma quantidade N de classes, referentes a intervalos uniformemente distribuídos entre 0% e 100%, onde 100% é a maior quantidade disponível do recurso em questão em um dos servidores disponíveis.

A etapa de mapeamento de estados inclui um novo componente: o mapeamento de estados dos servidores. Os servidores são agrupados de acordo com a maior classe dentre as classes em cada dimensão.

A maior modificação ocorre na etapa de consolidação de servidores:

1. Tenta-se alocar todos os *workloads* da maior classe definida nos servidores com essa classe. No exemplo da classificação de magnitude expressa na Tabela 3.1, tenta-se alocar os *workloads* FF, FL, FM, FS, LF, MF e SF em servidores F. Usa-se a heurística *First Fit* para realizar essa alocação.

2. Realiza-se uma iteração de agrupamento de *workloads*.

3. Repete-se a tentativa de alocação dos *workloads* da maior classe definida.

Caso uma tentativa de alocação de *workloads* em servidores da mesma classe não realize qualquer alocação, passa-se a considerar como "maior classe" a classe imediatamente inferior, e então o ciclo de tentativas e agrupamentos recomeça. Ainda neste exemplo, seriam então realizadas tentativas de alocar os *workloads* LL, LM, LS, ML e SL em servidores L.

Finalizados os ciclos para alocação em todas as classes, utiliza-se a estratégia First Fit para alocação das máquinas virtuais restantes.

As heurísticas *FirstFit*, *Best Fit Decreasing* (estratégias *BFD Item Centric* e *BFD Bin Centric*), *Slim BFD* e *Grouping VSVBP* foram avaliadas experimentalmente e implementadas no simulador descrito no próximo capítulo.

Capítulo 4

Simulação de plataformas de Computação em Nuvem

Plataformas de Computação em Nuvem empregam várias estratégias de gerenciamento de recursos computacionais para prover seus serviços de modo eficaz, ao garantir em contrato uma qualidade mínima do serviço prestado, e eficiente, ao buscar minimizar os custos envolvidos. As estratégias empregadas podem variar de acordo com especificidades do negócio, como o perfil de uso dos recursos das aplicações em execução, ou questões legislativas do local onde a plataforma está instalada.

A fim de analisar vantagens e desvantagens de novas estratégias para gerenciamento dos recursos computacionais, plataformas, em geral reduzidas, de Computação em Nuvem podem ser configuradas e ambientes virtualizados serem executados nessas plataformas. Ao seguir essa abordagem, os pesquisadores mantêm controle sobre arquitetura de rede, *softwares*, configurações dos servidores e perfis de uso de recursos dos ambientes virtualizados. Ainda que tal abordagem apresente benefícios ao considerar, direta ou indiretamente, a maioria dos fatores que influenciam a execução das estratégias em uma plataforma real de Computação em Nuvem, há muitos desafios para a implantação desses ambientes para testes em larga escala, que considerem centenas a milhares de servidores físicos. Para experimentação e análise do comportamento de estratégias em plataformas maiores, utiliza-se técnicas de simulação.

Uma simulação tem como objetivo executar um modelo que representa os elementos e interações que ocorrem em um ambiente real. Quando o modelo representa satisfatoriamente o comportamento do ambiente que busca representar, alterações são realizadas no modelo de modo a obter *insights* sobre como os elementos se comportariam caso as novas regras fossem aplicadas no ambiente real. Dentre as vantagens dessa abordagem, destaca-se o custo reduzido para experimentar novas regras de interação, bem como a diminuição dos riscos que porventura estejam envolvidos ao fazer testes em um ambiente real, como, por exemplo, no lançamento de foguetes. Para simular o emprego de diferentes estratégias para gestão de recursos computacionais em uma plataforma de Computação em Nuvem, foi desenvolvido o SimMyCloud [AC14].

O SimMyCloud foi projetado para que novos algoritmos de gestão de recursos sejam avaliados utilizando-se traços de execução de plataformas de Computação em Nuvem reais. São extraídos eventos lógicos e físicos presentes em tais traços e fornecida uma API que permite a criação de novas estratégias para gerenciamento de recursos computacionais, além de acesso a estatísticas de desempenho em tempo de execução e métricas consolidadas de desempenho ao final da simulação. O fluxo da simulação permite avaliar efeitos de diferentes estratégias para alocação dinâmica de ambientes virtualizados, predição de desempenho dos mesmos, migração de ambientes virtualizados e critérios para desligar máquinas físicas ou colocá-las em estado de baixo consumo de energia. Vale notar que durante o desenvolvimento do simulador foi considerado sua aplicação em gestão de plataformas para execução de máquinas virtuais, porém seu uso pode ser generalizado para execução de ambientes virtualizados.

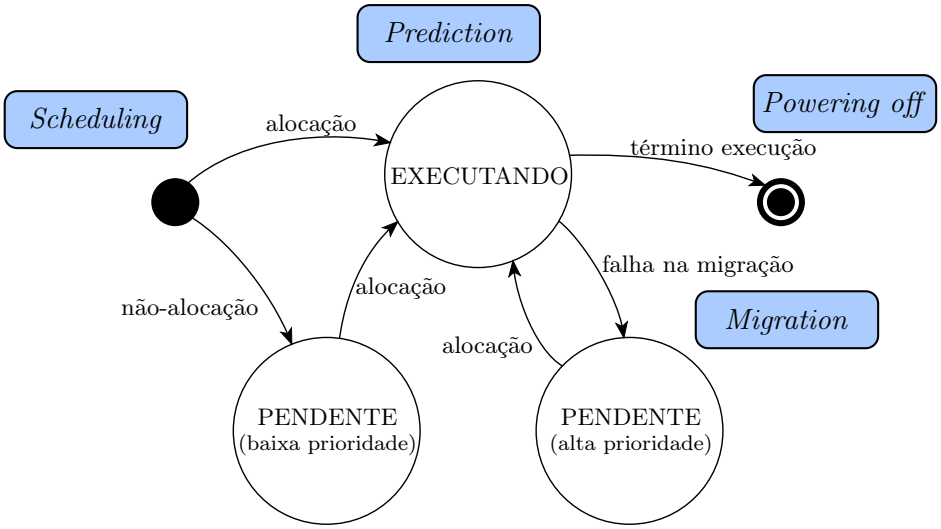


Figura 4.1: *Ciclo de vida de uma máquina virtual no* SimMyCloud *e momentos em que estratégias para gerenciamento de recursos computacionais mais atuam*

4.1 SimMyCloud

O SimMyCloud provê um arcabouço para a simulação e análise de diferentes estratégias de gerenciamento de recursos computacionais em ambientes com recursos virtualizados. O simulador foi modelado como um *sistema a eventos discretos*, onde a simulação ocorre com o processamento de uma sequência de eventos ordenados de acordo com um relógio lógico [MC10]. Nesse modelo de sistema, o estado do mesmo só pode ser alterado durante o processamento de um evento.

As máquinas virtuais são executadas sob um modelo de reservas, onde cada requisição para execução de uma máquina virtual contém a quantidade de recursos computacionais desejada e o tempo pelo qual ela deve se manter em execução. Tais requisições devem estar disponíveis em arquivos, os quais podem ser gerados artificialmente ou reproduzir os eventos ocorridos em uma plataforma real de Computação em Nuvem.

Realizada a requisição para execução de uma máquina virtual, a mesma segue o ciclo de vida ilustrado na Figura 4.1. Assim que um evento de requisição de execução ocorre, o arcabouço utiliza a estratégia de alocação de máquinas virtuais (*Scheduling strategy*) para tentar alocá-la em um servidor físico que supra sua demanda de recursos computacionais. Caso seja alocada, a máquina virtual passa ao estado EXECUTANDO. Caso contrário, ela passa para o estdo PENDENTE, sendo adicionada, com baixa prioridade, a uma fila de máquinas virtuais pendentes, a qual é verificada periodicamente para possível alocação de máquinas virtuais nesse estado.

A estratégia de predição de desempenho (*Prediction strategy*) prediz a demanda futura de recursos computacionais das máquinas virtuais no estado EXECUTANDO. Nesse momento, os recursos computacionais alocados às máquinas virtuais são atualizados pelos valores preditos. A estratégia de controle de migração (*Migration strategy*) é responsável, então, por identificar as máquinas virtuais que devem ser migradas, de modo a evitar sobrecarga de alguma máquina física. Tais máquinas virtuais têm seus estados gravados e são desalocadas das máquinas físicas que as hospedavam. A mesma estratégia é utilizada, então, para definir as máquinas físicas de destino dessas máquinas virtuais. As máquinas virtuais que não forem realocadas passam para o estado PENDENTE, sendo adicionadas, com alta prioridade, à fila de máquinas virtuais pendentes.

Quando a execução de uma máquina virtual termina, ou seja, a mesma ficou alocada pelo tempo de execução requisitado, a estratégia para desligamento ou suspensão de servidores físicos (*PoweringOff strategy*) é posta em uso para desligar, ou colocar em modo de baixo consumo de energia, os servidores que julgar capazes de serem colocadas nessa situação. A simulação termina quando todas as máquinas virtuais são executadas.

Durante a simulação, módulos de estatísticas são os responsáveis por computar dados relevantes

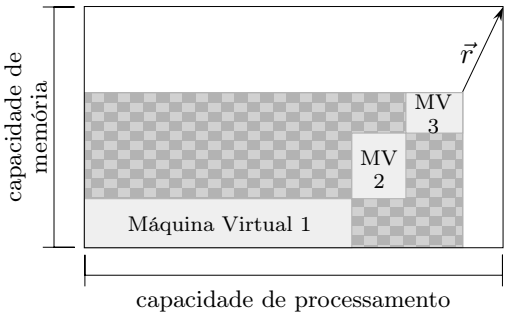


Figura 4.2: *Vetor residual de uma máquina física [AC14]*

para a comparação do emprego de diferentes estratégias, persistindo-os em arquivos que consolidam esses dados ou valores calculados. Novos módulos de estatísticas podem ser adicionados para medir e analisar diferentes métricas de desempenho. O SimMyCloud atualmente permite a análise de uma série de métricas, dentre as quais:

\# **violações de *SLA*** número de máquinas virtuais que executaram em uma máquina física sobrecarregada.

\# **máquinas físicas ativas** quantidade de máquinas físicas ativas, ou seja, não estão em um estado de baixo consumo de energia.

\# **migrações** quantidade de migrações de máquinas virtuais que ocorreram no último intervalo de tempo.

\# **máquinas virtuais em espera** quantidade de máquinas virtuais que não foram executadas por falta de recursos disponíveis[1].

Capacidade residual total soma da capacidade residual das máquinas físicas ativas.

% capacidade residual em cada máquina física porcentagem da capacidade residual de cada máquina física.

***Stretch* de cada máquina virtual** razão entre o tempo que a máquina virtual demora para ser executada (contado a partir de sua submissão no sistema) pelo tempo realmente utilizado para a sua execução.

Cabe elucidar o que é considerado uma violação de *SLA* e a capacidade residual de uma máquina física. Uma violação de *SLA* (*Service Level Agreement*) ocorre quando uma máquina virtual está alocada em uma máquina física sobrecarregada, ou seja, a quantidade de recursos disponíveis na máquina física é inferior à soma das demandas das máquinas virtuais alocadas nessa máquina física. A *capacidade residual* [PTUW11] de uma máquina física é definida como a norma do vetor residual $\vec{r}$, cujas componentes são determinadas pela quantidade ainda disponível de cada recurso na máquina física, como ilustrado na Figura 4.2.

Foram implementadas algumas estratégias para gestão de recursos computacionais sobre esse arcabouço. As heurísticas *FirstFit*, *Best Fit Decreasing* (versões *Bin Centric* e *Item Centric*), *Slim BFD* e *Grouping VSVBP*, respectivamente apresentadas nas seções 3.2, 3.3, 3.4 e 3.5, estão disponíveis como estratégias alocação de máquinas virtuais. A estratégia de controle de migrações implementada foi baseada em duas estratégias, uma apresentada por Ferreto e outros [FNCDR11] e outra por Khanna e outros [KBKK06]. Na estratégia resultante, chamada *MigrateIfOverload*, seleciona-se para migração os menores ambientes virtualizados que estão alocados em máquinas físicas sobrecarregadas, até que elas tenham menos que 100% de seus recursos alocados. Por "menores", considera-se os ambientes virtualizados com os menores tamanhos linearizados, os quais são

[1] Uma máquina virtual em espera não é considerada uma violação de *SLA*.

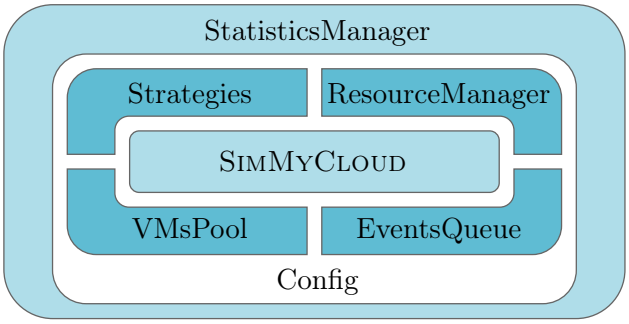


Figura 4.3: *Arquitetura do* SimMyCloud *[AC14]*

calculados pela soma absoluta da quantidade de recursos alocados para o ambiente virtualizado em questão.

Para suspensão ou desligamento de máquinas físicas, foi implementada a estratégia *PowerOffIfEmpty*, que indica para suspensão (ou desligamento) as máquinas físicas que não estão provendo recursos a nenhum ambiente virtualizado. Para a predição de desempenho foram implementadas duas estratégias. A primeira delas, *LastFiveMeasurements*, prevê que o uso de recursos no próximo intervalo de tempo será a média das últimas cinco medições. A segunda, *RBFPrediction*, considera as últimas medições de uso de recursos como uma série temporal, transforma-a para reconstrução em um espaço multidimensional e as representa na forma de funções de base radial, com o objetivo de realizar a predição de desempenho, que, no caso, seria o próximo valor da série.

O desenvolvimento do SimMyCloud seguiu alguns princípios de desenvolvimento de *software*, os quais foram úteis para alcançar uma arquitetura enxuta e simplificar a adição e configuração de novos módulos e estratégias. Foram quatro os princípios que guiaram as decisões arquiteturais do simulador, a saber:

Extensibilidade As regras e fluxos que podem variar entre diferentes simulações devem ser componentizados, de forma a permitir fácil adição ou substituição de regras de negócio.

Softcoding Variáveis e seleção de estratégias e módulos não devem ser *hard-coded*, de modo a permitir que simulações diferentes possam ser executadas sem necessidade de alterações no código fonte.

Duck typing Assume-se que se os objetos manipulados possuem certos métodos e propriedades, então os mesmos têm o comportamento e a semântica esperadas, o que evita o uso de computação para verificar constantemente se estratégias e módulos são de tipos específicos.

Agnosticismo perante componentes Cada componente tem sua responsabilidade e sabe de que dados ou outros componentes da simulação precisa, assim, todos os componentes da simulação podem ser acessados pelos demais de uma forma padronizada, o que permite minimizar a injeção de dependencias.

Com base nesses princípios, foi concebida a arquitetura do SimMyCloud, ilustrada na Figura 4.3. O componente SimMyCloud é o responsável por executar o fluxo principal do simulador, desde a configuração do ambiente até a execução de rotinas para finalizar a simulação. Além desses pontos extremos, coordena o acesso aos demais componentes de acordo com os eventos a serem processados. As estratégias consolidam as regras de negócio da simulação: se há diversas maneiras para se resolver uma questão, elas são as responsáveis por tomar as decisões e executá-las. Uma estratégia de consolidação de servidores, por exemplo, deve decidir em quais servidores um conjunto de ambientes virtualizados devem ser alocados, e alocá-los.

A tarefa de abstrair o acesso aos recursos computacionais é do componente *ResourceManager*. Para se iniciar a execução de ambientes virtualizados em um servidor, ter acesso aos servidores ativos e suspensos ou mesmo atualizar a quantidade de recursos alocados a um ambiente virtualizado,

entre outras tarefas relacionadas, é necessário que o componente que deseja executar essas tarefas interaja com o *ResourceManager*. O *VMsPool* gerencia os ambientes virtualizados que não estão em execução, caso que ocorre quando não há servidores com recursos computacionais disponíveis que supram a demanda desses ambientes. O componente *EventsQueue* lida com os eventos a serem processados no simulador, desde a recepção de novos eventos, que podem ser inseridos a qualquer momento, até a priorização dos mesmos para processamento.

Os componentes *StatisticsManager* e *Config* atuam de modo indireto na simulação. O *Config* gerencia os componentes configurados para a simulação atual, e fornece uma interface padronizada para que cada componente possa acessar ou interagir com os demais. O *StatisticsManager*, por sua vez, é responsável por receber notificações dos eventos de simulação que ocorrem (por exemplo: alocação de ambiente virtualizado, servidor ativado, ou término da execução de um ambiente virtualizado), redirecionando-os aos módulos de estatísticas que os consumirão para persistir as estatísticas desejadas.

4.2 Traço de execução ClusterData2011_1

Dados sobre requisições e alocação e uso de recursos em plataformas de Computação em Nuvem raramente são disponibilizados pelas administradoras dessas plataformas. Porém, a fim de incentivar pesquisas que levassem em consideração situações mais próximas das realmente enfrentadas pelas empresas, a Google disponibilizou em 2010 um traço de execução com o uso de recursos de diversas máquinas virtuais que executaram em seu ambiente, durante o período de sete horas [Hel10]. Em 2011, a Google disponibilizou outro traço de execução com mais dados, o ClusterData2011_1 [RWH11].

O ClusterData2011_1, parcialmente explorado por Reiss e outros pesquisadores [RTG+12], possui dados referentes a eventos e monitoração ocorridos em um período de 29 dias, em uma célula[2] dos *datacenters* da Google, a qual era composta por aproximadamente 12.500 máquinas físicas. Os dados, que somam cerca de 40Gb compactados, possuem informações sobre eventos de *jobs* e tarefas, eventos e atributos de máquinas físicas, restrições de tarefas e uso de recursos por cada tarefa.

Para fins de confidencialidade, os dados foram anonimizados, de modo a impossibilitar a associação de dados divulgados a usuários, aplicações ou mesmo detalhes da infraestrutura. Com isso, atributos de máquinas físicas e requisição e uso de recursos foram normalizados com base na máquina física com maior capacidade, o que força os valores finais a ficarem no intervalo $[0.0, 1.0]$. A Tabela 4.1 apresenta as máquinas físicas disponíveis no início do período monitorado, com alguns respectivos atributos.

Número de máquinas	Plataforma	CPUs	Memória
6732	B	0.50	0.50
3863	B	0.50	0.25
1001	B	0.50	0.75
795	C	1.00	1.00
126	A	0.25	0.25
52	B	0.50	0.12
5	B	0.50	0.03
5	B	0.50	0.97
3	C	1.00	0.50
1	B	0.50	0.06

Tabela 4.1: *Configuração das máquinas na célula. Capacidades de CPU e memória foram linearmente transformadas, de modo que a maior capacidade de cada recurso é de `1.0`. As plataformas indicam o uso de três diferentes combinações de microarquiteturas e tecnologias de memória. Adaptado de [RTG+12]*

[2] Uma célula é uma coleção de máquinas físicas que são operadas como uma unidade de gerenciamento única

Capítulo 5

Análise experimental

Com o objetivo de identificar quais estratégias de gestão de recursos computacionais em plataformas de Computação em Nuvem são mais indicadas para reduzir a quantidade de servidores físicos ligados, e ao mesmo tempo respeitar acordos de nível de serviço, foram realizados experimentos sobre essas plataformas.

O arcabouço SimMyCloud, apresentado na Seção 4.1, foi utilizado como base para a execução de simulações com diferentes combinações de estratégias de predição de desempenho e de consolidação de servidores. Optou-se por utilizar nas simulações um subconjunto dos dados de um traço de execução de uma plataforma real de Computação em Nuvem, de modo a considerar, nas simulações, características de requisições e uso de recursos de aplicações reais executadas nessa plataforma. O traço de execução utilizado é o ClusterData2011_1 [RWH11], divulgado pelo Google e apresentado na Seção 4.2.

O traço de execução original indica a quantidade de recursos utilizados por cada tarefa de cada *job*[1], de modo que cada tarefa foi considerada, nas simulações, uma máquina virtual independente. A quantidade e a capacidade de recursos dos servidores físicos na plataforma simulada são as mesmas presentes no início do traço de execução, de tal modo que podem ser conferidas na Tabela 4.1.

Uma análise preliminar do traço de execução mostrou que, no período dos 29 dias com dados monitorados e disponibilizados, foram executadas 24.595.382 tarefas com características bem diversas: desde grandes serviços *web* a programas com uso intensivo de CPU, tarefas requisitadas tanto por aplicativos quanto por usuários humanos, além de possuírem diferentes prioridades [RTG+12].

A fim de realizar uma análise mais fina dos efeitos sobre o uso de recursos ao empregar diferentes estratégias de consolidação de servidores e predição de desempenho, foram consideradas as requisições de execução de máquinas virtuais ocorridas nas primeiras 6h do traço de execução. Além disso, suas execuções foram truncadas para que todas terminassem nesse período. Uma requisição às 5h que requeria a execução de um ambiente virtualizado pelo período de dois dias, por exemplo, foi truncada para ser executada por um período de 1h. As informações sobre o uso de recursos por cada tarefa foram utilizadas tais como disponibilizadas no traço original.

Os experimentos foram projetados para analisar as seguintes métricas de desempenho: quantidades de servidores ativos, servidores sobrecarregados, violações de acordos de nível de serviço, ambientes virtualizados no estado PENDENTE (ver Seção 4.1), máquinas virtuais migradas, a capacidade residual dos servidores ativos, e o *stretch* de cada máquina virtual. Todos os dados foram coletados durante as simulações, e podem representar ou o montante acumulado entre períodos subsequentes, como, por exemplo, quantidade de ambientes virtualizados migrados, ou um valor calculado no tempo de simulação em que os dados foram gerados, como, por exemplo, a capacidade residual dos servidores ativos. A exceção a esses casos é o cálculo do *stretch* de cada máquina virtual, que é realizado no momento em que ela termina sua execução com sucesso. O estado do sistema em um tempo de simulação reflete qual seria o estado do sistema nesse mesmo instante em uma plataforma real de Computação em Nuvem com as mesmas características e demandas.

[1] No traço de execução, toda tarefa pertence a um único *job*, e um *job* pode ter uma ou mais tarefas, as quais podem ser executadas paralelamente em diferentes servidores.

Uma das métricas mede quantas violações de acordos de nível de serviço houve em cada intervalo de tempo em uma simulação. Como informado na Seção 4.1, este trabalho considera que uma violação de acordo de nível de serviço ocorre para uma máquina virtual quando o servidor físico que provia recursos para a mesma está sobrecarregado, ou seja, a soma das demandas das máquinas virtuais (no traço de execução original) é superior à capacidade de recursos do servidor em questão. Apesar de essa quantidade de violações evidenciar que "faltou recursos" para executar algumas máquinas virtuais, não há qualquer compensação sobre essa situação. Possíveis compensações poderiam variar de cancelamento e reexecução dessas máquinas virtuais, a expandir o tempo necessário para finalizar a execução dos ambientes virtualizados presentes nesse servidor.

Os servidores físicos podem estar, durante a simulação, no estado desligado (ou suspenso), quando não podem prover recursos a qualquer máquina virtual, ou ativo, quando estão disponíveis para prover recursos a máquinas virtuais. Caso a estratégia de gestão de recursos opte por alocar uma máquina virtual em um servidor desligado, a mesma pode requerer sua ativação e então alocar a máquina virtual neste servidor. Por hora, o modelo assume que esta operação ocorre instantaneamente, ignorando o fato de que servidores necessitam de um tempo para inicializar (ou sair do modo suspenso). Consideração similar foi realizada no processo de migração de máquinas virtuais: apesar de haver custos envolvidos na migração das mesmas, uma estratégia que opte por fazer essa migração irá considerar que ela ocorre instantaneamente, e sem custo para a infraestrutura.

Todas as simulações foram executadas com uma estratégia de migração de máquinas virtuais que seleciona para migração os menores ambientes virtualizados[2] das máquinas sobrecarregadas, até que saiam deste estado. A estratégia para desligar (ou suspender) máquinas físicas periodicamente verifica que máquinas físicas estão ativas, mas sem prover recursos a qualquer ambiente virtualizado, e então as desliga (ou as coloca em estado suspenso).

As variações nas estratégias de consolidação de servidores e de predição de desempenho foram realizadas de modo independente, resultando em nove experimentos, visto que foram utilizadas três variações para cada uma dessas estratégias. As heurísticas de alocação de servidores utilizadas foram a *FirstFit* (*vide* Seção 3.2), a *Grouping VSVBP* (Seção 3.5) a o *Slim BFD* (Seção 3.4), enquanto os algoritmos *LastFiveMeasurements* e *RBFPrediction* foram utilizados como estratégias para predição de desempenho. Para fins de comparação, foram executadas simulações sem o uso de uma estratégia de predição de desempenho, de forma que mantém alocados aos ambientes virtualizados os recursos inicialmente requisitados pelos respectivos usuários.

O algoritmo *LastFiveMeasurements* prevê que, durante o próximo intervalo de tempo, cada ambiente virtualizado irá utilizar a média dos recursos que utilizou nas cinco medições anteriores. No início da execução de um ambiente virtualizado, onde as cinco mensurações podem ainda não ter sido realizadas, a quantidade de recursos requeridos pelo usuário é utilizado no cálculo da média.

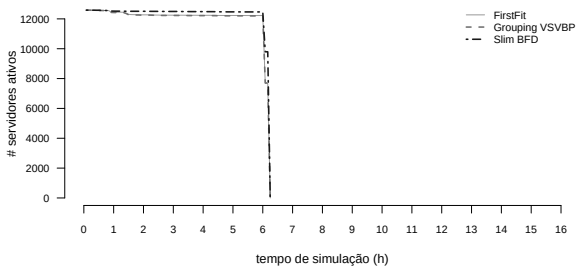
A partir da classificação do processo gerador de uma série temporal, sugere-se o uso de algumas técnicas para predição de valores na mesma [Ish10]. Ao analisar o uso de recursos de CPU e memória no subconjunto utilizado do traço de execução, e modelando as medições de cada ambiente virtualizado em séries temporais, percebeu-se que as séries temporais seguiam processos de geração estocásticos e não lineares. O método de predição com função de base radial, apresentado na Seção 2.6.1, é uma das indicações para tal classificação, tendo sido escolhido para uso na presente pesquisa. A estratégia de predição de desempenho *RBFPrediction* tem como base a implementação disponível no pacote `TISEAN` [HKS99], a qual avalia previamente a série de entrada para fixar os centros das funções radiais.

Os cálculos das previsões de uso de recursos de CPU e memória são realizados de forma independente. Optou-se por considerar um histórico de 50 observações no uso de recursos para se realizar a predição com a estratégia *RBFPrediction*. Desse modo, até que as 50 medições estejam disponíveis, não são realizadas alterações na quantidade de recursos computacionais alocados para a máquina virtual, ou seja, são mantidos alocados os recursos requisitados pelo usuário.

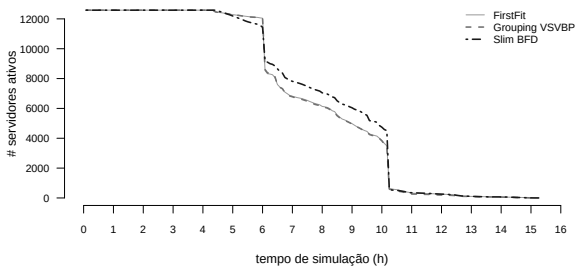
[2] Para se comparar o tamanho de ambientes virtualizados, a estratégia de migração realiza uma transformação nos vetores que indicam a demanda de recursos. Uma máquina virtual com demanda $s_i = (cpu_i, mem_i)$ é considerada de tamanho $s'_i = cpu_i + mem_i$.

5.1 Análise das simulações

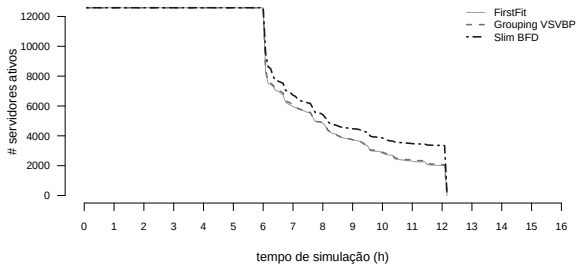
As figuras 5.1a, 5.1b e 5.1c exibem a quantidade de servidores ativos em cada tempo de simulação. É possível perceber nos mesmos uma variação no comportamento ao longo da execução dos ambientes virtualizados, bem como que o tempo de simulação necessário para executar todos os ambientes virtualizados requisitados foi diferente. Foram necessários aproximadamente 6h15min, 15h15min e 12h10min do tempo de simulação para executar todos os ambientes virtualizados no período do traço de execução definido anteriormente, respectivamente ao empregar os algoritmos *LastFiveMeasurements*, *RBFPrediction* e sem empregar um algoritmo de predição de desempenho. Dado que foi utilizado um subconjunto referente a 6h do traço de execução, esse foi o tempo necessário para executar todos esses ambientes virtualizados no ambiente do Google. A relação entre os tempos necessários em cada situação pode ser vista na Figura 5.2.



(a) *algoritmo* LastFiveMeasurements



(b) *algoritmo* RBFPrediction



(c) *sem algoritmo de predição de desempenho*

Figura 5.1: *Quantidade de máquinas físicas ativas por tempo de simulação, ao utilizar os algoritmos de predição de desempenho* LastFiveMeasurements *(5.1a),* RBFPrediction *(5.1b) e sem uso de um algoritmo de predição de desempenho (5.1c).*

Quando cada máquina virtual possui à disposição a exata quantidade de recursos requisitados, percebe-se que máquinas físicas ainda ficam sobrecarregadas, o que acarreta em violações de acordos de nível de serviço, como pode ser verificado nas figuras 5.3a e 5.3b. Apesar da quantidade de máquinas sobrecarregadas ser menor nesse cenário, quando comparado com cenários em que utiliza-se um algoritmo de predição de desempenho (ver figuras 5.4a e 5.5a), percebe-se que uma alocação dinâmica de recursos se faz necessária: uma máquina virtual pode, em algum momento, necessitar de mais recursos que os inicialmente requisitados. Lidar com essa elasticidade é uma das características essenciais da Computação em Nuvem [MG09]. A utilização de algoritmos de predição de desempenho pode reduzir a necessidade de os usuários terem que acompanhar o desempenho de suas aplicações, para decidirem sobre quando e como a quantidade de recursos disponibilizada a seus ambientes virtualizados deve ser alterada.

Ao adicionar à analise as figuras 5.4a, 5.4b, 5.5a e 5.5b, que apresentam a quantidade de máquinas físicas sobrecarregadas e o número de violações de acordos de nível de serviço ao utilizar um algoritmo de predição de desempenho, tem-se que o uso do algoritmo *RBFPrediction* levou a

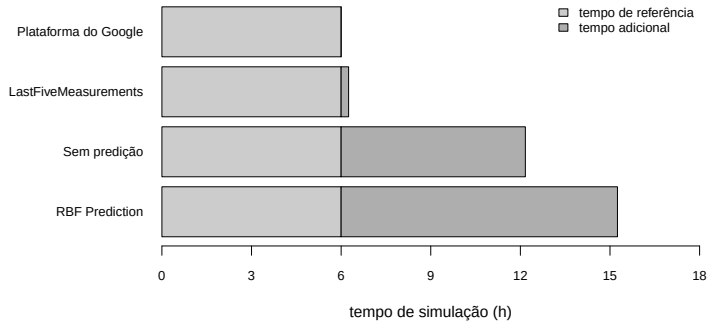
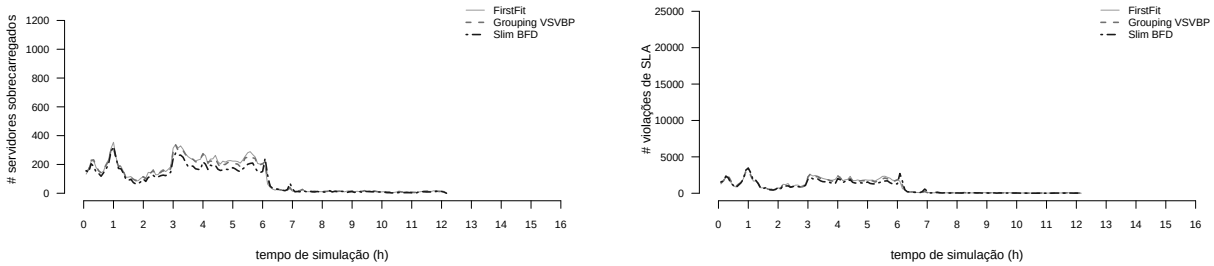


Figura 5.2: *Tempo de simulação necessário para executar todas as máquinas virtuais requisitadas ao utilizar diferentes estratégias de predição de desempenho.*

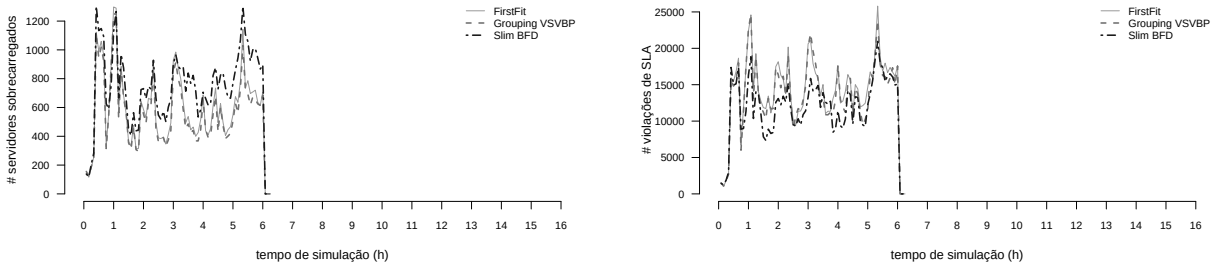


(a) *servidores sobrecarregados* **(b)** *violações de acordos de nível de serviço*

Figura 5.3: *Quantidades de servidores sobrecarregados (5.3a) e violações de acordos de nível de serviço (5.3b) por tempo de simulação, sem uso de um algoritmo de predição de desempenho.*

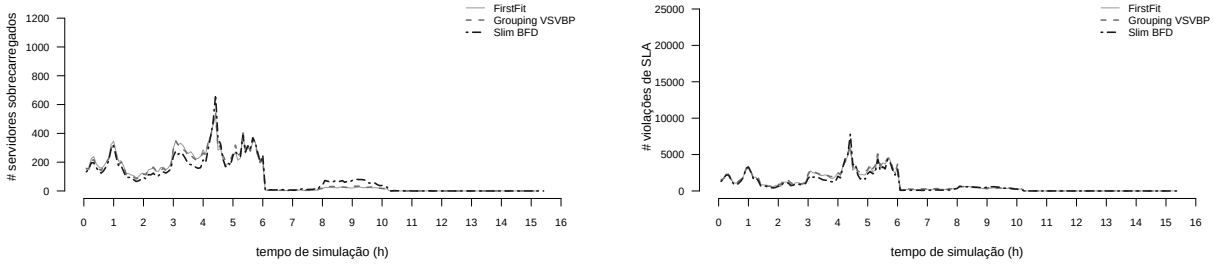
menos sobrecargas e violações de acordos de nível de serviço que o uso do algoritmo *LastFiveMeasurements*. Ao empregar o algoritmo *RBFPrediction* para predição de desempenho, houve, em média, 492 servidores físicos sobrecarregados e 12.807 violações de acordos de nível de serviço a menos que ao empregar o algoritmo *LastFiveMeasurements*. Essa análise, porém, não deve ser realizada de modo isolado, pois o tempo de simulação necessário para executar todos os ambientes virtualizados requisitados foi, como citado anteriormente, maior nas simulações com emprego do algoritmo de predição *RBFPrediction*. Assim, caso se priorize a não-violação de acordos de nível de serviço, e se opte pelo uso de um algoritmo de predição de desempenho, o uso do algoritmo *RBFPrediction* para predição de desempenho é indicado.

Nas figuras que apresentam a quantidade de máquinas físicas ativas ao empregar o algoritmo de predição de desempenho *LastFiveMeasurements* (Figura 5.1a), o algoritmo *RBFPrediction* (Figura 5.1b) e na simulação que não utiliza um algoritmo de predição (Figura 5.1c), pode-se perceber que o número de servidores ativos é levemente maior ao empregar a heurística de consolidação de servidores *Slim BFD*. Os experimentos que empregaram a heurística *Slim BFD* utilizaram, em média, 264 servidores físicos a mais em relação aos experimentos que utilizaram a heurística *FirstFit*, e 222 servidores físicos a mais, em relação aos experimentos com uso da *Grouping VSVBP*. Nessa mesma heurística de consolidação, é perceptível, pelas figuras 5.6a, 5.6b e 5.6c, que a quantidade de ambientes virtualizados no estado PENDENTE também é maior que nos demais casos (em média 1055 e 935 ambientes virtualizados a mais no estado PENDENTE, em relação às heurísticas *FirstFit* e *Grouping VSVBP*). Esse conjunto de fatores pode explicar o motivo pelo qual há menor número de violações de acordos de nível de serviço quando essa estratégia de consolidação é empregada (ver figuras 5.3b, 5.4b e 5.5b), afinal, tem-se mais recursos disponíveis, e menos ambientes virtualizados estão em execução.



(a) *servidores sobrecarregados* **(b)** *violações de acordos de nível de serviço*

Figura 5.4: *Quantidades de servidores sobrecarregados (5.4a) e violações de acordos de nível de serviço (5.4b) por tempo de simulação, com utilização do algoritmo de predição de desempenho* LastFiveMeasurements.



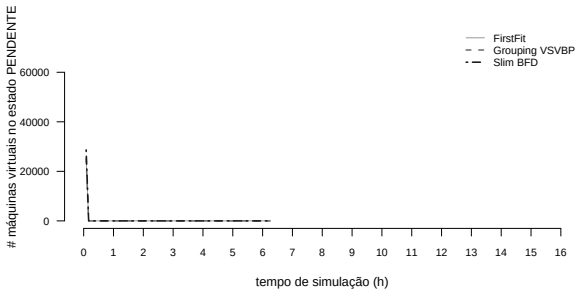
(a) *servidores sobrecarregados* **(b)** *violações de acordos de nível de serviço*

Figura 5.5: *Quantidades de servidores sobrecarregados (5.5a) e violações de acordos de nível de serviço (5.5b) por tempo de simulação, com utilização do algoritmo de predição de desempenho* RBFPrediction.

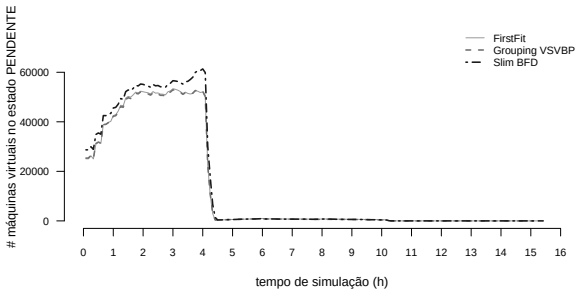
Esse uso maior de máquinas físicas afeta também a capacidade residual, que mede a quantidade de recursos em servidores físicos ativos que não estão alocados (como apresentado na Seção 4.1). Ela é maior ao empregar a heurística *Slim BFD* que ao empregar as heurísticas de consolidação de servidores *FirstFit* e *Grouping VSVBP*. A capacidade residual ao longo da execução das simulações é apresentada nas figuras 5.7a, 5.7b e 5.7c. Logo, caso o provedor aceite pagar o custo de mais máquinas físicas ativas para diminuir a quantidade de violações de SLA, indica-se o uso da heurística de consolidação de servidores *Slim BFD*.

Os experimentos permitiram avaliar também a distribuição do *stretch* dos ambientes virtualizados. O *stretch*, como apresentado na Seção 4.1, é a razão entre o tempo decorrido entre a requisição de execução de uma máquina virtual e a finalização de sua execução, e o tempo total em que a mesma esteve em execução. Dessa forma, quanto maior o *stretch* de uma máquina virtual, maior o tempo relativo que o usuário precisou aguardar para obter os resultados de suas execuções. A Figura 5.8 apresenta a distribuição dos valores de *strech* das máquinas virtuais executadas, nas simulações em que a heurística de consolidação de servidores *FirstFit* foi empregada. O experimento que simulou o emprego do algoritmo *LastFiveMeasurements* registrou os valores de *stretch* mais baixos para as máquinas virtuais (ver Figura 5.8a), de modo que o usuário seria pouco impactado pelas alocações tardias e preempções, as quais ocorrem por não haver servidores físicos capazes de disponibilizar a quantidade de recursos predita para todas as máquinas virtuais.

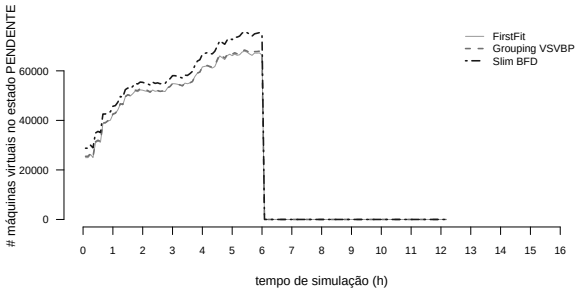
A simulação que não empregou um algoritmo de predição (ver Figura 5.8c) apresentou os maiores valores de *stretch*, sendo 1.382,56 o maior valor de *stretch* registrado para uma máquina virtual nesse experimento. O maior valor de *stretch* registrado na simulação com emprego do algoritmo *RBFPrediction* (ver Figura 5.8b) também foi alto: 921,06. Valores altos para essa métrica deveriam



(a) *algoritmo* LastFiveMeasurements

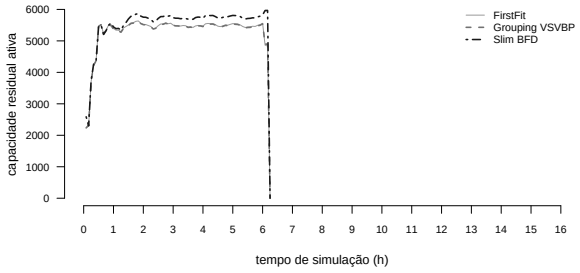


(b) *algoritmo* RBFPrediction

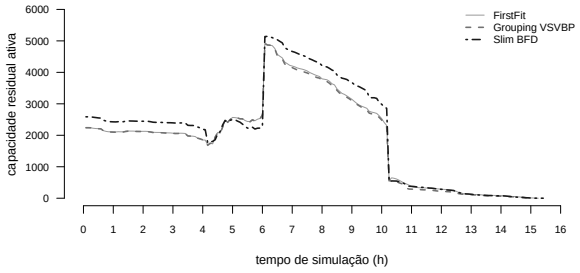


(c) *sem algoritmo de predição de desempenho*

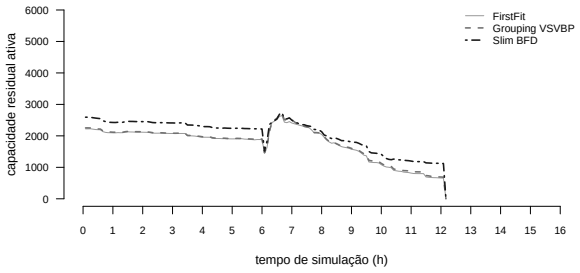
Figura 5.6: *Quantidade de ambientes virtualizados no estado PENDENTE em cada tempo de simulação, ao utilizar os algoritmos de predição de desempenho* LastFiveMeasurements *(5.6a),* RBFPrediction *(5.6b) e sem uso de um algoritmo de predição de desempenho (5.6c).*



(a) *algoritmo* LastFiveMeasurements

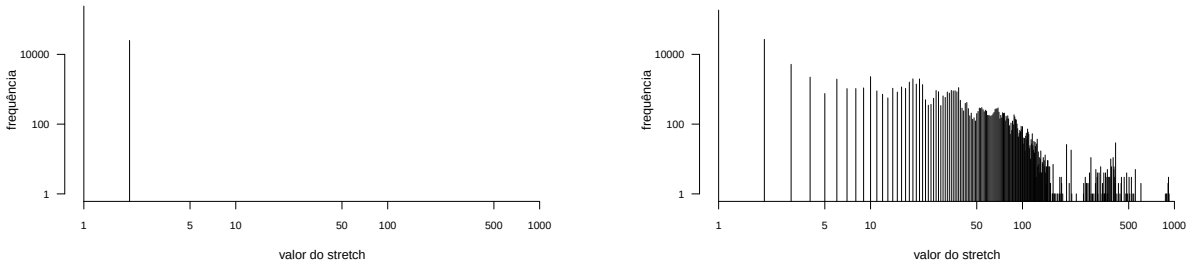


(b) *algoritmo* RBFPrediction



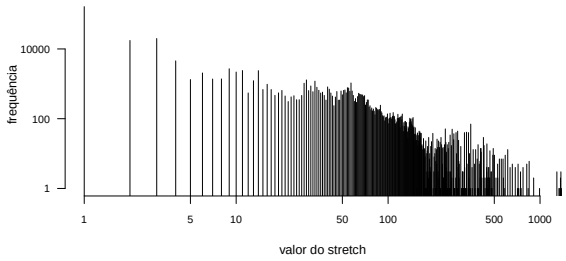
(c) *sem algoritmo de predição de desempenho*

Figura 5.7: *Capacidade residual em cada tempo de simulação, ao utilizar os algoritmos de predição de desempenho* LastFiveMeasurements *(5.7a),* RBFPrediction *(5.7b) e sem uso de um algoritmo de predição de desempenho (5.7c).*



(a) *algoritmo* LastFiveMeasurements

(b) *algoritmo* RBFPrediction



(c) *sem algoritmo de predição de desempenho*

Figura 5.8: *Distribuição dos valores de* stretch *dos ambientes virtualizados nas simulações com uso da heurística* FirstFit *de alocação de máquinas virtuais, com emprego dos algoritmos de predição de desempenho* LastFiveMeasurements *(5.9a) e* RBFPrediction *(5.9b), e sem uso de um algoritmo de predição de desempenho (5.9c).*

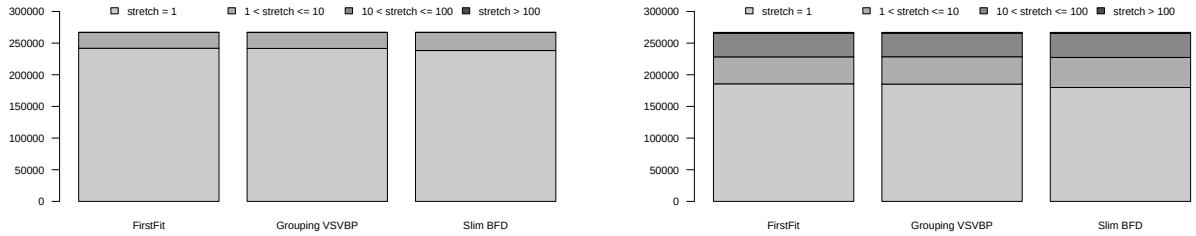
ser evitados, visto que o objetivo de uma plataforma de Computação em Nuvem é fornecer recursos de computação aos mais diversos usuários, torna-se inviável considerar que uma aplicação que deveria ser executada por, por exemplo, cinco segundos, termine sua execução somente após 1h23min, para um valor hipotético de *stretch* igual a 1.000.

As figuras 5.9a, 5.9b e 5.9c exibem a distribuição dos valores de *stretch* registrados em cada experimento. Como a distribuição dos valores foi semelhante ao empregar uma mesma estratégia de predição, as análises ao empregar as heurísticas de consolidação de servidores *Slim BFD* e *Grouping VSVBP* se tornam análogas à análise anterior, com o emprego da heurística *FirstFit*. Assim, o uso do algoritmo de predição de desempenho *LastFiveMeasurements* é indicado caso valores limite de *stretch* sejam definidos em acordo de nível de serviço.

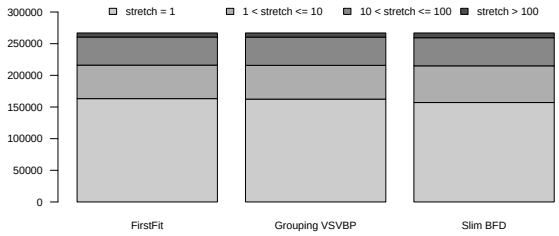
Em análise da quantidade de ambientes virtualizados migrados, percebe-se que não houve migrações nos experimentos em que não foram utilizadas estratégias de predição de desempenho, ainda que tenham sido registradas violações de acordos de nível de serviço nesses mesmos casos (ver Figura 5.3b). Essa situação ocorre pois ao abdicar do uso de um algoritmo de predição de desempenho, dá-se ao simulador a informação de que cada ambiente virtualizado irá utilizar, em qualquer intervalo de tempo, a mesma quantidade de recursos requisitada pelo respectivo usuário. Com base nesse dado, o simulador confia que os servidores físicos atuais conseguirão prover os recursos necessários no próximo intervalo de tempo a todos os ambientes virtualizados nele alocados, ou seja, não há necessidade de migração pois acredita-se que as máquinas físicas não ficarão sobrecarregadas.

Esses dois valores devem ser então interpretados da seguinte maneira: quanto maior a quantidade de migrações, maior era a probabilidade de haver violações em acordos de nível de serviço, e então os ambientes virtualizados foram migrados para máquinas físicas diferentes[3]; e quanto maior o

[3]Caso nenhuma máquina física tenha recursos ainda não alocados para disponibilizar a esses ambientes virtualizados, eles são colocados no estado PENDENTE, com alta prioridade. Esses casos não são contabilizados como migrações de máquinas virtuais

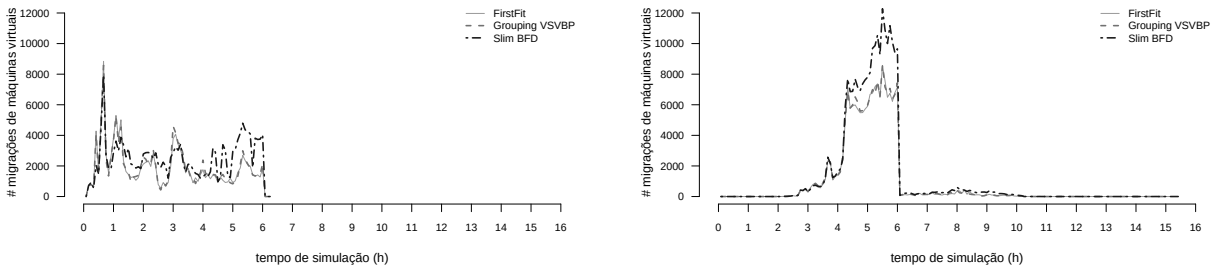


(a) *algoritmo* LastFiveMeasurements **(b)** *algoritmo* RBFPrediction



(c) *sem algoritmo de predição de desempenho*

Figura 5.9: *Distribuição dos valores de* stretch *dos ambientes virtualizados nas simulações com emprego dos algoritmos de predição de desempenho* LastFiveMeasurements *(5.9a) e* RBFPrediction *(5.9b), e sem uso de um algoritmo de predição de desempenho (5.9c).*



(a) *algoritmo* LastFiveMeasurements **(b)** *algoritmo* RBFPrediction

Figura 5.10: *Quantidade de migrações de ambientes virtualizados por tempo de simulação, com uso dos algoritmos de predição de desempenho* LastFiveMeasurements *(5.10a) e* RBFPrediction *(5.10b).*

número de acordos de nível de serviço violados, maior foi a imprecisão do desempenho predito dos ambientes virtualizados no tempo anterior, de modo que o algoritmo em uso subestimou a quantidade de recursos que seriam utilizados. As figuras 5.10a e 5.10b exibem a quantidade de máquinas virtuais que foram migradas em cada intervalo de tempo da simulação ao utilizar os algoritmos de predição de desempenho *LastFiveMeasurements* e *RBFPrediction*, respectivamente. Ao utilizar o algoritmo *RBFPrediction*, não há migrações no início das simulações devido à restrição de serem necessárias, no mínimo, 50 mensurações de uso de recursos para se executar predição de desempenho. Após esse período, começam a ser realizadas uma grande quantidade de migrações, cerca de 3 a 4 vezes a mais que com uso do *LastFiveMeasurements*. Apesar disso, o emprego do algoritmo de predição *RBFPrediction* reduz a quantidade de servidores sobrecarregados e violações de acordos de nível de serviço em relação ao emprego do *LastFiveMeasurements*. Logo, caso o custo de migrações seja baixo em uma infraestrutura de Computação em Nuvem, indica-se empregar o

algoritmo de predição de desempenho *RBFPrediction*.

As análises neste capítulo evidenciam vantagens e desvantagens no uso das diferentes estratégias de gestão de recursos computacionais em uma plataforma de Computação em Nuvem. Decisões de quais estratégias devem ser utilizadas ficam a cargo do provedor de serviço, o qual deve verificar quais características são mais desejáveis, com base em suas estratégias de mercado.

Capítulo 6

Conclusões

O presente trabalho analisou os efeitos sobre a quantidade de servidores físicos ativos e outras características relacionadas à qualidade do serviço prestado por uma plataforma de Computação em Nuvem ao empregar diferentes estratégias de gerenciamento de recursos computacionais. Tanto a quantidade de servidores físicos quanto as características das tarefas executadas refletiram características de uma plataforma real de um provedor de Computação em Nuvem, o que aproximou o desafio de alcançar bons resultados nas simulações dos desafios reais enfrentados por um provedor desses serviços.

Na análise dos experimentos, percebe-se que, para o traço de execução utilizado, uma mudança na estratégia de predição de desempenho afeta mais a execução dos ambientes virtualizados do que mudanças na estratégia de consolidação de servidores. Das heurísticas de consolidação de servidores avaliadas, o *Slim BFD* apresenta resultados animadores ao diminuir a quantidade de violações de acordos de nível de serviço (ocorrem nesse caso, em média, 493 e 429 violações de acordos de nível de serviço a menos, em comparação com as heurísticas *FirstFit* e *Grouping VSVBP*, respectivamente), porém requer uma maior quantidade de máquinas físicas ativas (que usam, em média, 252 e 193 servidores físicos ativos a mais, em relação às mesmas heurísticas, respectivamente.)

Nos experimentos, o algoritmo de predição de desempenho *LastFiveMeasurements* apresentou melhores resultados ao necessitar de menos tempo para executar todas as máquinas virtuais, diminuir a quantidade de migrações realizadas e registrar valores mais aceitáveis para o *stretch* das máquinas virtuais. O ônus dessas vantagens recai sobre a quantidade de violações de acordos de nível de serviço, que chega a ser, em média, 20 vezes maior que a quantidade de violações de acordos de nível de serviço ao empregar o algoritmo de predição de desempenho *RBFPrediction*.

A proposta de utilizar uma estratégia de predição de desempenho a fim de diminuir a quantidade de máquinas físicas ativas não foi alcançada como esperado. Dentre possíveis fatores que contribuíram para esse cenário está o fato de que o traço de execução utilizado foi retirado de uma plataforma de Computação em Nuvem que utiliza diversos métodos para melhorar a eficiência no uso dos recursos: tanto sobrealocação de recursos, como apresentado por Reiss e outros pesquisadores [RTG+12], quanto outras estratégias protegidas por questões de segredo industrial. Não se sabe, também, se as tarefas executadas na plataforma do Google demandariam mais recursos, porém foram limitadas devido ao limite físico de recursos disponíveis.

Foi também apresentado o SimMyCloud, simulador de plataformas de Computação em Nuvem desenvolvido com o objetivo de auxiliar na análise de diversos efeitos ao empregar diferentes estratégias de gestão de recursos nessas plataformas. O simulador foi disponibilizado à comunidade no GitHub, sob a licença MIT [AC14, Pai], e um Guia do Usuário foi disponibilizado na página do projeto – http://www.ime.usp.br/~cassiop/simmycloud.

6.1 Considerações finais

A grande quantidade de fatores que influenciam os desafios envolvidos no gerenciamento de recursos computacionais em uma plataforma de Computação em Nuvem acarretou em mais simpli-

ficações no modelo de simulação do que o desejado, como os custos nulos para ativação de servidores e migração de máquinas virtuais. Duas versões do SimMyCloud chegaram a ser desenvolvidas, e muitas funcionalidades foram adicionadas desde as primeiras simulações executadas, visando, principalmente, tornar o ambiente simulado mais próximo do real. Muitas alterações ainda são necessárias no mesmo, algumas delas citadas na Seção 6.2, mas ao deixá-lo aberto à comunidade, acredita-se que as melhorias serão implementadas de acordo com as demandas dos pesquisadores que optarem por utilizar o arcabouço.

Os resultados obtidos com os experimentos executados indicaram como a definição de uma estratégia para gerenciar os recursos computacionais pode influenciar no uso eficiente dos mesmos. Análises de resultados obtidos durante a realização da pesquisa resultaram mais em alterações no simulador, tanto por necessidades técnicas quanto para aproximar o ambiente simulado de situações reais, do que no desenvolvimento e ajustes das estratégias em uso. Assim, o estágio atual do SimMyCloud permite agilizar pesquisas futuras que tenham uma etapa de avaliação dos efeitos ao empregar diferentes estratégias para a gestão de recursos em uma plataforma de Computação em Nuvem.

6.2 Sugestões para pesquisas futuras

Pesquisas futuras podem aproveitar o arcabouço construído e experimentar o efeito de novas estratégias para a gestão de recursos computacionais em plataformas de Computação em Nuvem. Durante o desenvolvimento da pesquisa percebeu-se uma carência de heurísticas para lidar com a alocação de ambientes virtualizados, visto que devem ser consideradas tanto demandas quanto capacidades em múltiplas dimensões, e com tamanhos variados. Também podem ser exploradas melhorias em algoritmos de predição de desempenho, a fim de reduzir, por exemplo, o *stretch* das máquinas virtuais.

Pode ser interessante explorar a possibilidade de comparar o emprego das estratégias em diferentes traços de execução. Para execução deste trabalho optou-se por utilizar um único traço de execução, mas tal decisão inviabilizou a avaliação de possíveis vieses adicionados pelo mesmo. Os perfis das aplicações costumam variar em diferentes traços, e poderia ser possível analisar, por exemplo, se uma estratégia de predição de desempenho é mais assertiva com um perfil de aplicação do que com outras. Além de outros traços, adaptações no mesmo traço de execução podem ser realizadas, como, por exemplo, verificar os efeitos ao empregar as diferentes estratégias de gestão de recursos ao considerar uma plataforma de Computação em Nuvem com 50% a mais de servidores físicos, mantendo a proporção quanto a cada tipo de servidor.

Melhorias no SimMyCloud podem passar a considerar um intervalo de tempo que seria necessário para inicializar um servidor físico, ou colocá-lo em um modo de baixo consumo de energia. A distribuição física dos servidores é outro fator que pode ser levado em consideração, que permitiria, por exemplo, distribuir ou concentrar a execução de máquinas virtuais de um mesmo usuário, de acordo com objetivos considerados na heurística de consolidação de servidores.

O traço de execução utilizado neste trabalho foi o ClusterData2011_1, disponibilizado à comunidade em Novembro de 2011 [Wil11]. Porém, em Novembro de 2014, uma nova versão do traço de execução foi disponibilizada, o ClusterData2011_2 [Wil14], a qual adiciona um campo às informações de uso de recursos de cada tarefa: um exemplo de quantidade de CPU utilizada pela tarefa durante um intervalo aleatório de 1 segundo dentro do período medido. Com esse novo campo disponível, trabalhos futuros podem considerar tal valor como uma medição realizada, tanto para fins de predição de desempenho quanto para verificação de servidores sobrecarregados.

Referências bibliográficas

[AC14] Cássio P. Alkmin e Daniel Cordeiro. SimMyCloud, simulando o gerenciamento de recursos virtualizados em plataformas de Computação em Nuvem. 2014. vii, 2, 21, 23, 24, 37

[AFG+09] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica e Matei Zaharia. Above the clouds: A berkeley view of cloud computing. Relatório Técnico UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Fev 2009. 5

[Bra] Filipe Daniel Alves Brandão. vpsolver - Vector Packing Solver based on an arc-flow formulation with graph compression - Google Project Hosting. Disponível em https://code.google.com/p/vpsolver/. Acessado em 02/11/2014. 15

[CFH+05] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt e Andrew Warfield. Live migration of virtual machines. Em *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, páginas 273–286. USENIX Association, 2005. 2, 7

[CG14] Faruk Caglar e Aniruddha Gokhale. iOverbook: intelligent resource-overbooking to support soft real-time applications in the cloud. Em *7th IEEE International Conference on Cloud Computing (IEEE CLOUD)*, Anchorage, AK, USA, Jun–Jul 2014. vii, 11, 12

[DKJ11] Shyam Kumar Doddavula, Mudit Kaushik e Akansha Jain. Implementation of a fast vector packing algorithm and its application for server consolidation. Em *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, páginas 332–339. IEEE, 2011. 2, 8, 15

[euc14] Open source private and hybrid clouds from Eucalyptus, 2014. Disponível em http://www.eucalyptus.com/. Acessado em 06/11/2014. 7

[FDF05] Renato Figueiredo, Peter A. Dinda e Jose Fortes. Guest editors' introduction: Resource virtualization renaissance. *Computer*, 38(5):28–31, 2005. 7

[FNCDR11] Tiago C. Ferreto, Marco A. S. Netto, Rodrigo N. Calheiros e César A. F. De Rose. Server consolidation with migration control for virtualized data centers. *Future Generation Computer Systems*, 27(8):1027–1034, 2011. 8, 23

[GJ79] Michael R. Garey e David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979. 9

[GoG] GoGrid. Best of Cloud + Best of Hosting | GoGrid. Disponível em http://www.gogrid.com/. Acessado em 21/09/2014. 10

[GZ13] Michaël Gabay e Sofia Zaourar. Variable size vector bin packing heuristics - Application to the machine reassignment problem. Setembro 2013. 8, 9, 17

[Hel10] Joseph L. Hellerstein. Google cluster data. Google research blog, Janeiro 2010. Publicado em http://googleresearch.blogspot.com/2010/01/google-cluster-data.html. 8, 25

[HJLW14] Rongdong Hu, Jingfei Jiang, Guangming Liu e Lixin Wang. Efficient resources provisioning based on load forecasting in cloud. *The Scientific World Journal*, 2014. 11

[HKS99] Rainer Hegger, Holger Kantz e Thomas Schreiber. Practical implementation of nonlinear time series methods: The TISEAN package. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 9(2):413–435, 1999. 28

[Ish10] Renato Porfirio Ishii. *Otimização de operações de entrada e saída visando reduzir o tempo de resposta de aplicações distribuídas que manipulam grandes volumes de dados.* Tese (Doutorado em Ciências de Computação e Matemática Computacional), Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2010. ix, 11, 12, 13, 28

[KBKK06] Gunjan Khanna, Kirk Beaty, Gautam Kar e Andrzej Kochut. Application performance management in virtualized server environments. Em *Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP*, páginas 373–381. IEEE, 2006. 23

[Liu11] Huan Liu. A measurement study of server utilization in public clouds. Em *Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on*, páginas 435–442. IEEE, 2011. 10

[MC10] Afonso C. Medina e Leonardo Chwif. *Modelagem e Simulação de Eventos Discretos.* Editora Campus, 3 edição, Julho 2010. 22

[MG09] Peter Mell e Tim Grance. The NIST definition of cloud computing. *National Institute of Standards and Technology*, 53(6):50, 2009. 5, 6, 29

[MT90] Silvano Martello e Paolo Toth. *Knapsack problems.* Wiley New York, 1990. 9

[ope14] OpenNebula - open source data center virtualization, 2014. Disponível em http://opennebula.org/. Acessado em 06/11/2014. 7

[ovi14] oVirt, 2014. Disponível em http://www.ovirt.org/. Acessado em 06/11/2014. 7

[Pai] Cássio Paixão. SimMyCloud. Disponível em http://www.ime.usp.br/~cassiop/simmycloud. Acessado em 30/10/2014. 37

[PTUW11] Rina Panigrahy, Kunal Talwar, Lincoln Uyeda e Udi Wieder. Heuristics for vector bin packing. Não publicado. Disponível em: http://research.microsoft.com/apps/pubs/default.aspx?id=147927, 2011. 8, 23

[RTG+12] Charles Reiss, Alexey Tumanov, Gregory R. Ganger, Randy H. Katz e Michael A. Kozuch. Towards understanding heterogeneous clouds at scale: Google trace analysis. Relatório Técnico ISTC–CC–TR–12–101, Intel science and technology center for cloud computing, Carnegie Mellon University, Pittsburgh, PA, USA, Abril 2012. Posted at http://www.istc-cc.cmu.edu/publications/papers/2012/ISTC-CC-TR-12-101.pdf. ix, 1, 10, 25, 27, 37

[RWH11] Charles Reiss, John Wilkes e Joseph L. Hellerstein. Google cluster-usage traces: format + schema. Technical report, Google Inc., Mountain View, CA, USA, Novembro 2011. Revisado em 20/03/2012. Disponível em http://code.google.com/p/googleclusterdata/wiki/TraceVersion2. 10, 12, 25, 27

[Ser] Amazon Web Services. AWS | Amazon Elastic Compute Cloud (EC2) - Scalable Cloud Hosting. Disponível em http://aws.amazon.com/ec2/. Acessado em 21/09/2014. 10

[SMLF09] Borja Sotomayor, Rubén S. Montero, Ignacio M. Llorente e Ian Foster. Virtual infrastructure management in private and hybrid clouds. *Internet Computing, IEEE*, 13(5):14–22, 2009. 7

[Vog08] Werner Vogels. Beyond server consolidation. *Queue*, 6(1):20–26, Janeiro 2008. 8

[Wil11] John Wilkes. More Google cluster data. Google research blog, Novembro 2011. Publicado em http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html. 38

[Wil14] John Wilkes. ClusterData2011_2. Google code, Novembro 2014. Publicado em https://code.google.com/p/googleclusterdata/wiki/ClusterData2011_2. 38

[ZCB10] Qi Zhang, Lu Cheng e Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1):7–18, 2010. 1