POSICIONAMENTO ONLINE BASEADO NO TRÁFEGO DE MÁQUINAS VIRTUAIS EM REDES DE DATA CENTER

Daniel de Souza Dias

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Luís Henrique Maciel Kosmalski Costa

Rio de Janeiro
Março de 2013

POSICIONAMENTO ONLINE BASEADO NO TRÁFEGO DE MÁQUINAS VIRTUAIS EM REDES DE DATA CENTER

Daniel de Souza Dias

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

Prof. Luís Henrique Maciel Kosmalski Costa, Dr.

Prof. Miguel Elias Mitre Campista, D.Sc.

Prof. Pedro Branconnot Velloso, Dr.

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2013

Dias, Daniel de Souza

Posicionamento online baseado no tráfego de máquinas virtuais em redes de data center/Daniel de Souza Dias. – Rio de Janeiro: UFRJ/COPPE, 2013.

XIV, 91 p.: il.; 29, 7cm.

Orientador: Luís Henrique Maciel Kosmalski Costa

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2013.

Referências Bibliográficas: p. 83 – 91.

1. Data center. 2. Computação em nuvem. 3. Posicionamento ciente de tráfego. I. Costa, Luís Henrique Maciel Kosmalski. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

Dedico este trabalho a todos os meus familiares e amigos.

Agradecimentos

Agradeço aos meus pais, irmã e toda a minha família que com muito carinho e apoio não mediram esforços para que eu chegasse até esta etapa da minha vida.

Ao meu orientador Luís Henrique pela paciência na orientação e incentivo que tornou possível a conclusão deste trabalho.

A Kamilla, que sempre consegue arrancar-me um sorriso e me presenteou com inúmeros momentos felizes.

Aos amigos e colegas pelo incentivo, apoio e bom humor constantes.

Agradeço aos professores Miguel Campista e Pedro Velloso pela participação na banca examinadora.

Agradeço aos funcionários do Programa de Engenharia Elétrica da COPPE/UFRJ, pela presteza no atendimento na secretaria do Programa.

Aos professores do Grupo de Teleinformática e Automação, por todo o auxílio acadêmico e profissional que recebi.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

POSICIONAMENTO ONLINE BASEADO NO TRÁFEGO DE MÁQUINAS VIRTUAIS EM REDES DE DATA CENTER

Daniel de Souza Dias

Março/2013

Orientador: Luís Henrique Maciel Kosmalski Costa

Programa: Engenharia Elétrica

Com a evolução dos computadores e das redes de telecomunicações em geral, gerou-se uma oportunidade para a melhoria dos serviços de troca de informações. Para conseguir implementar funcionalidades com poder de processamento de dados, escala de distribuição e acesso elevados é preciso uma infraestrutura robusta, denominada centros de processamento de dados (data centers). Existe uma grande dificuldade em achar a maneira otimizada de interconectar os computadores de um centro de processamento de forma econômica e que atenda a todas as demandas, já que os maiores data centers podem conter mais de 100.000 servidores. A possibilidade de migração de máquinas virtuais, advinda da virtualização, permite escolher corretamente a posição em que grandes trocas de dados ocorrem no centro de processamento de dados, sendo possível distribuí-las com o intuito de diminuindo a sobrecarga da rede, principalmente em momentos de pico de atividades.

Apresenta-se nesta dissertação um algoritmo para determinar o posicionamento de máquinas virtuais em centros de processamento de dados. O Algoritmo de Posicionamento de Serviços, APoS, analisa o comportamento das máquinas virtuais que trocam dados frequentemente, agrupa-as e posiciona-as no data center de forma que o fluxo de informações entre essas máquinas interfira de forma menos intensa nos fluxos de outras máquinas virtuais. Informações de uso de memória, uso de CPU, comunicação de redes e topologia da rede são usadas para determinar o correto posicionamento das máquinas virtuais. Através de simulações foi possível verificar que o algoritmo proposto consegue diminuir em até 80% o tráfego de dados no núcleo da rede do centro de processamento da dados, concentrando-o nas bordas da rede.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

ONLINE TRAFFIC-AWARE VIRTUAL MACHINE PLACEMENT IN DATA CENTER NETWORKS

Daniel de Souza Dias

March/2013

Advisor: Luís Henrique Maciel Kosmalski Costa

Department: Electrical Engineering

With the evolution of computers and telecommunication networks in general, there has been an opportunity to improve services that exchange information. To be able to implement functionality like high data processing power and wide distribution of content, we need a robust infrastructure, called data centers. There is great difficulty in finding the optimal manner to interconnect computers in a data center in an economical way that meets all the processing demands, as the largest data centers contain over 100,000 servers. Virtualization techniques allows to properly choose the position where large traffic exchange occur in the data center, it is possible to distribute them reducing the overhead areas of the network, especially at times of peak activity.

This thesis presents an algorithm to determine the placement of virtual machines in data centers. The proposed algorithm analyzes the behavior of virtual machines that exchange data often, groups them and places them in the data center so that the flow of information between these machines interfere less intense on the flows of other virtual machines. Information of memory usage, CPU usage, network communication and network topology is used to determine the correct placement of virtual machines. Through simulations we found that the proposed algorithm can reduce up to 80 % data traffic in the network core, concentrating it on the edges of the network.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

APoS	Algoritmo de Posicionamento de Serviços, p. 5, 15, 32, 34, 44, 53, 54, 56, 58, 63–71, 74, 75, 80–82
CDP	Cisco Discovery Protocol, p. 47
CPU	Unidade central de processamento, p. 2, 12–15, 26, 28, 34, 41, 44, 47, 54, 63–65, 67–71, 73, 75, 80, 81
CUDA	Compute Unified Device Architecture, p. 31
GB	Gigabytes, p. 63, 66
GHz	GigaHertz, p. 66
GPU	Graphics Process Unit, p. 31
ICMP	Internet Control Message Protocol, p. 47
ICN	Information-Centric Networking, p. 32
ICN	Information-centric networking, p. 32
IP	Internet Protocol, p. 22, 24, 25, 46, 54
LHC	Large Hadron Collider, p. 31
MAC	Media Access Control address, p. 22, 25
MB	Megabytes, p. 63, 66
MPI	Message Passing Interface, p. 31
OpenMP	Open Multi-Processing, p. 31
PHP	PHP: Hypertext Preprocessor, p. 32
PVM	Parallel Virtual Machine, p. 31
RAM	Random-access memory, p. 66

SNMP	Simple Network Management Protocol, p. 11, 47
TCP	Transmission Control Protocol, p. 13
ToR	Top of Rack, p. 9
VM	Virtual Machine, p. 15
WMI	Windows Management Instrumentation, p. 47

Capítulo 1

Introdução

Centros de processamento de dados, também chamados de *data centers*, agregam o poder computacional de vários computadores com o objetivo de possibilitar a execução de aplicações que necessitam de vastos recursos computacionais de processamento, de memória e de armazenamento de dados [1]. Essas aplicação podem ser usadas tanto para atender a uma grande quantidade de usuários quanto para executar tarefas que precisem de um grande poder computacional.

O objetivo desse modelo computacional é fazer o melhor uso possível de recursos computacionais distribuídos. Dispô-los em um mesmo ambiente permite atingir maiores taxas de uso dos recursos computacionais, energéticos e de pessoal necessário para sua operação. O uso de *hardwares* e *softwares* adequados, torna possível o solucionamento de problemas computacionais em larga escala de forma econômica [2–6].

Grandes empresas como o UOL, Google, Amazon, Microsoft, entre outras, oferecem serviços para seus usuários em que são necessários um alto poder computacional e uma alta taxa de comunicação. Pesquisa de endereços eletrônicos, correio eletrônico através de páginas da Internet, edição de documentos de forma colaborativa, armazenamento, sincronização e compartilhamento de arquivos remotos são algumas das ferramentas disponibilizadas que precisam de um poder computacional elevado. Quando consideramos o número de usuários que utilizam essa ferramenta, o poder computacional necessário é ainda mais intenso. Em 2012 existiam mais de 2,4 bilhões de usuários de Internet [7], sendo que, segundo o Google [8], no início de 2012 existiam mais de 425 milhões de usuários cadastrados utilizando uma conta de correio eletrônico do Google, o Gmail. Mais de 1,2 trilhões de buscas foram feitas em 2012 usando os serviços do Google [9].

O número de máquinas interligadas em um data center varia muito de acordo com o seu uso. Segundo Luo [10] um dos data centers do maior serviço de busca da China, o Baidu, terá em torno de 100.000 servidores, distribuído em 4.000 racks, totalizando 700.000 núcleo de computadores e mais de 4 Exabytes de armazenamento de dados.

Para o funcionamento de um data center, servidores são interligados permitindo a comunicação dos processos executados, formando uma rede de comunicação que abrange todo o data center. Essa rede é de extrema importância, pois nela transitam todas as requisições internas e externas aos serviços que são executados. Essa rede também é a porta de entrada e saída para o mundo externo, através do roteador de borda o data center é conectado a Internet. Sendo assim, o funcionamento dessa rede é crucial visto que qualquer aspecto negativo, como congestionamentos, alta latência e diminuição da banda de dados, podem interferir diretamente no tempo de resposta de uma requisição feita por um cliente.

1.1 Motivação

Para possibilitar uma melhor administração de data centers, que entre outras características precisa controlar a quantidade de processamento, uso de memória e tráfego de rede que necessários por cada servidor, técnicas de migrações tem sido usadas. Essas técnicas podem ser aplicadas tanto em ambientes onde existe a virtualização de máquinas [11] quanto em ambientes que permitem a movimentação da imagem do sistema operacional da máquina hospedeira, como o usado pelo sistema de busca da Microsoft, o Bing [12]. Com migrações é possível escolher exatamente o local onde um serviço será executado, permitindo o controle de diversas variáveis como o uso da rede, o controle do consumo de energia, a tolerância a falhas, entre outras, trazendo flexibilidade ao data center [13–15].

Ao utilizar a virtualização é possível aumentar o uso de um servidor. Serviços que necessitam de poucos recursos podem compartilhar um mesmo hardware, sendo consolidados em um mesmo servidor. Consolidar essas máquinas virtuais distribui melhor o uso dos recursos e consequentemente melhora a eficiência do data center, já que se compartilha a CPU, a memória e o disco.

Uma máquina virtual é uma interface com o *hardware* que se encontra abaixo dela, tendo características de fornecer essa interface aos recursos de forma eficiente e isolada. [16–19]. Cada uma das máquinas virtuais é criada garantindo uma fração da quantidade dos recursos de CPU, memória e disco disponíveis no hospedeiro. Esses recursos são atribuídos de forma única a uma máquina virtual, ou seja, uma determinada fração do recurso, como um endereço de memória, não deve ser compartilhado simultaneamente para duas máquinas virtuais. O compartilhamento feito dessa forma garante o isolamento dos recursos e a impossibilita a interferência entre máquinas virtuais no mesmo hospedeiro, garantindo o isolamento de serviços no data center.

Um ambiente de nuvem consiste em disponibilizar para os usuários uma plataforma dedicada para a implementação de serviços. Nesse ambiente táticas como

virtualização são amplamente utilizadas, permitindo disponibilizar um servidor para mais de um cliente ao mesmo tempo, viabilizando colocar múltiplos serviços em um mesmo *hardware*.

Uma nuvem é caracterizada por disponibilizar aos seus usuários uma grande quantidade de poder computacional sem que ele tenha que se preocupar com as particularidades envolvidas na operação e manutenção dos servidores. Uma nuvem provém uma interface entre os serviços que o usuário quer executar e a máquina que executa esse serviço. Ambientes de nuvem permitem os novos serviços disponíveis na Internet diminuindo os custos de operação [5]. Essa nova maneira de disponibilizar os serviços para os usuários diminui a necessidade de processamento do cliente e consequentemente permite o seu uso em diversas plataformas, principalmente nas plataformas com pouco poder de processamento, como os *smartphones*, *netbooks* e *tablets*.

Uma nuvem também diminui os custos para os usuários pois essa é executada sobre um data center [6]. Um data center permite executar operações computacionais de forma eficiente e com baixo custo. Como data centers são dedicados a fornecer poder computacional, táticas de redução de custos são amplamente utilizadas para conseguir obter o melhor custo possível. A virtualização, a refrigeração líquida, a alimentação por corrente contínua ou o uso de geradores para situações de emergência são táticas que nem sempre estão disponíveis facilmente para o consumidor comum, que não tem a capacidade, recurso ou motivação necessárias para montar um ambiente como o encontrado em um data center.

As empresas responsáveis pela implementação dos centros de processamento de dados devem planejar corretamente as demandas dos serviços executados sobre os data centers na hora do projeto da construção. Um data center de tamanho exagerado não é viável economicamente porque é necessária uma elevada quantia para a sua construção, sendo que servidores subutilizados não geram lucro, pelo contrário, somente mais despesas de operação e manutenção. Por outro lado um data center de capacidade insuficiente pode receber um número excessivo de requisições, causando congestionamentos, o que aumenta o tempo de resposta e pode até impossibilitar a realização de uma tarefa planejada. Em ambos os casos será gerada uma despesa extra para a execução das tarefas planejadas.

Uma maneira de cortar gastos é o cálculo correto da infraestrutura de rede. A interconexão física, as capacidades dos enlaces, a quantidade de redundância necessária pelas aplicações são alguns dos parâmetros considerados no projeto da rede do data center. Para conseguir um bom aproveitamento com um custo mais baixo, algumas topologias especiais podem ser utilizadas. A topologia mais comumente utilizada consiste em agregar um determinado número de máquinas em um mesmo comutador e então interligar esses comutadores, formando uma topologia do tipo

árvore [20].

Com essas simplificações, a conexão de rede entre servidores nem sempre tem disponível a capacidade máxima que as suas interfaces de rede dispõem, já que compartilham o mesmo meio de transmissão, surgindo áreas de congestionamento na rede [21–25]. Novas topologias, novas técnicas de engenharia de rede e novos serviços vêm sendo apresentados para tentar atenuar esse problema.

Estudos prévios mostram que existe congestionamento de enlaces em data centers operacionais [19, 26–28]. Nesses estudos foram coletados dados reais de data centers que executam aplicações de computação distribuída, como os baseados em aplicações de MapReduce (a ferramenta de MapReduce é apresentada em detalhes na Seção 3.2.2), em centros de processamento de dados que alugam os seus espaços para diversos clientes, centros de processamento empresariais e universitários com diversas demandas e centros de processamento com serviços de busca da Internet. Esses estudos indicam que existem problemas de rede que devem ser observados e atenuados ao máximo. Em todos os trabalhos foi observado o congestionamento de enlaces específicos dos data centers, dependendo do tipo de topologia, do tipo de aplicação executada no data center, e principalmente do posicionamento do enlace na infraestrutura do data center. Em data centers que operam com topologia de árvore quanto mais distantes das folhas forem os enlaces, chamados de enlaces de núcleo, mais alta será a taxa de utilização, em alguns casos chegam a apresentar congestionamentos severos, diminuindo a velocidade de comunicação entre grandes seções do data center.

Benson *et al.* [27, 28] observam que a distribuição de máquinas virtuais em data centers não é aleatória. Os operadores escolhem arbitrariamente onde um serviço deve ser estabelecido, seguindo políticas pré-determinadas mas não de forma ótima ou totalmente autônoma. Isso traz a possibilidade de criar algoritmos ou métodos capazes de automatizar a localização de máquinas virtuais em data centers, baseados nas necessidades impostas pelo controlador do data center ou baseados em normas e restrições impostas pelos operadores.

O tráfego em um data center é baseado no serviço em que ele fornece. Como característica comum a todos temos que um data center não é utilizado para somente um grande serviço, eles executam múltiplos serviços simultaneamente [27, 28]. Podemos considerar que uma determinada tarefa que utilize conceitos de programação distribuída entre diversos servidores necessite se comunicar de forma mais intensa e mais frequente com os servidores que participam dessa tarefa do que com outro servidores. Dessas características podemos definir grupos, onde cada grupo está relacionado a um serviço.

Em cada um dos grupos teremos uma troca de informações interna intensa, podendo agravar problemas de congestionamento dos enlaces se esses serviços forem

localizados em áreas críticas dentro do data center. É necessário encontrar formas de identificar e aproximar os grupos relacionados, possibilitando a escolha da localização dos servidores em lugares que o tráfego seja melhor distribuído dentro de um data center, minimizando o congestionamento de enlaces específicos.

1.2 Objetivo

Conforme explicitado, existe um grande crescimento no uso e na importância de centros de processamento de dados através do crescimento da demanda de aplicativos na nuvem. Devido ao projeto do data center, economizando no número de enlaces, problemas de congestionamento dos enlaces internos de data centers ocorrem e devem ser atenuados para manter a qualidade dos serviços prestados. Antes de considerar a aquisição de mais recursos de infraestrutura de rede, deve-se considerar outras abordagens para atenuar o congestionamento dos enlaces.

Congestionamentos podem ser causados por defeitos nos softwares, que usam mais banda do que o necessário para executar as comunicações. Esse é um defeito de difícil diagnóstico e nem sempre de fácil correção. Outro fator de congestionamento são falhas na topologia, que pode conter enlaces não operantes ou que não estão presentes na topologia, resultando no uso de outros enlaces para compensar esta falha. É necessário localizar e recuperar esses enlaces faltosos.

Uma outra abordagem para os congestionamentos é que não existem defeitos ou falhas na rede mas ela se encontra sobrecarregada. Nesse caso podemos usar o ajuste do posicionamento de máquinas virtuais para distribuir o tráfego da rede, melhorando o nível de uso do núcleo e consequentemente o nível de congestionamento da rede, sem que seja necessária a aquisição de novos equipamentos.

Neste trabalho propõe-se o APoS, Algoritmo de Posicionamento de Serviços, um algoritmo para melhorar a distribuição de tráfego no data center, alocando serviços em áreas pré-determinadas para minimizar os gargalos de rede. O APoS foi idealizado a partir da necessidade de alocar dinamicamente os serviços, representados por máquinas virtuais, dentro de um data center considerando a visão da topologia de rede.

Neste trabalho, um serviço é considerado contido em uma máquina virtual exclusiva. Desse modo, ao monitorar uma máquina virtual, conseguimos as características do serviço desejado, que são informações de consumo de CPU, memória e de comunicação com outros serviços (máquinas virtuais). Podemos considerar que uma máquina virtual possui mais de um serviço, mas somente conseguiremos mapear o conjunto destes, sendo estes transformadas em um único serviço pelo algoritmo.

O APoS é capaz de identificar nas topologias de um data center locais em que a comunicação entre os servidores de máquinas virtuais tem uma maior banda,

chamados de partições. Com essa identificação, máquinas virtuais que necessitam de uma troca constante de dados podem ser beneficiadas ao permanecerem em uma partição, contribuindo para o menor uso das partes da topologia onde existe uma restrição maior quanto a banda passante disponível.

Outra parte importante do trabalho foi a identificação de serviços relacionados dentro de um data center. Através da análise da matriz de tráfego entre as máquinas virtuais são identificados os conjuntos de máquinas virtuais que trocam mais dados entre si do que com outras máquinas virtuais. Ao descobri-los é possível agrupá-los em uma mesma partição, contribuindo para o aumento do uso das porções não congestionadas do data center e consequentemente diminuindo o uso das porções mais utilizadas no data center.

A ideia básica é selecionar os serviços que se comunicam muito, através do monitoramento do tráfego de rede, e alocá-los o mais perto possível dentro do data center, fazendo com que menos dados tenham de trafegar no núcleo da rede, permanecendo nas bordas da topologia.

Para avaliar o APoS, é proposto um modelo para a entrada de dados, que prevê o tráfego entre as máquinas virtuais baseados nos estudos de Benson *et al.* [27]. Os resultados mostram que em determinados níveis altos de demanda o algoritmo proposto é capaz de reduzir em 80% o uso do núcleo da rede, distribuindo esse tráfego para as partições designadas.

Para gerar os resultados foi desenvolvido um simulador próprio de redes de data center. Esse simulador precisou ser desenvolvido pois não foi encontrado na literatura um simulador com todas as características necessárias, como a geração de uma matriz de tráfego a partir de modelos matemáticos, a implementação do algoritmo de posicionamento de serviços, o monitoramento das partições de servidores e que conseguisse escalar para mais de 16.000 máquinas virtuais.

Além da proposta APoS, outras contribuições deste trabalho são: a modificação do algoritmo Girvan-Newman para a detecção de padrões de comunidades, reduzindo a sua complexidade neste ambientes de troca de mensagens e o desenvolvimento de um simulador para o estudo do problema de alocação de máquinas virtuais em um data center, observando os diferentes usos da rede distribuídos pela topologia.

1.3 Organização da dissertação

Este trabalho está organizado da seguinte forma. O Capítulo 2 apresenta o estado da arte e os trabalhos relacionados. O Capítulo 3 descreve um ambiente típico de data center, nele são identificados as principais topologias de rede utilizadas em data center e os tipos de data centers que podem ser beneficiados com a proposta são elucidados. O Capítulo 4 define o problema e apresenta os principais métodos

utilizados para a resolução. O Capítulo 5 apresenta, em detalhes, o algoritmo proposto nesta dissertação para a resolução do problema, o algoritmo é dividido em etapas e todas cuidadosamente demonstradas. O Capítulo 6 apresenta o simulador desenvolvido para o estudo do problema e a avaliação do algoritmo proposto. Primeiro é descrito o funcionamento do simulador e depois a sua implementação e os resultados são comentados. Por fim, o Capítulo 7 conclui esta dissertação e apresenta os trabalhos futuros.

Capítulo 2

Estado da Arte

Diversos aspectos das redes de data center são de interesse atual da comunidade científica [12, 29–39]. Os desafios em aberto vão desde a localidade para a sua construção até maneiras de manter sua operação a um baixo custo. Os mecanismos de funcionamento de centros de processamento de dados são complexos, principalmente devido ao seu porte, que pode passar de centenas de milhares de servidores em um mesmo local [40].

Com o aumento da oferta e do uso de aplicações através da Internet existe uma demanda crescente para o uso de centros de processamento de dados. Verdi *et al.* mostram que a popularização dos serviços de nuvem, tais como correio eletrônico baseado na *web*, *sites* de busca e redes sociais, associados ao aumento da conectividade através de banda larga e redes ópticas, impulsionaram o modelo de comunicação centrado no servidor. Cada vez mais o processamento e o armazenamento estão sendo movidos dos computadores pessoais para grandes provedores de serviço. Essa mudança faz com que o uso e a importância de data centers aumentem, já que eles são os responsáveis por executar o processamento e o armazenamento dos dados desses serviços de nuvem.

Dados do Banco Mundial [41] mostram que, em 2000, 7% da população mundial tinha acesso à Internet, já em 2005 este quantitativo aumentou para 16%. Em 2010 o número de pessoas com acesso passou para 30%, chegando a mais de 2 bilhões de usuários, com isso praticamente dobrando a cada 5 anos. Em 2012 temos 2,4 bilhões, ou 34% da população mundial com acesso à Internet [7].

Com mais usuários consequentemente existe uma maior demanda por serviços hospedados na nuvem. Considerando que cerca de 66% da população ainda não conta com acesso à Internet, existe uma grande possibilidade de crescimento dos serviços de nuvem, alavancando desse modo o uso e a importância de data centers. Koomey [14] analisa dados de gastos de energia de data centers entre 2000 e 2005. Segundo os dados levantados por ele, de 2000 até 2005 a base instalada de data center teve um crescimento maior do que 100% em todas as partes do mundo, chegando a

quase 26 mil centros de processamento de dados únicos.

2.1 Organização física do data center

Um centro de processamento de dados é uma instalação especificamente construída para abrigar uma elevada quantidade de computadores que se comunicam entre si e executam serviços. Para alcançar uma elevada eficiência aliada a um baixo custo de implementação, os principais componentes são construídos de forma padronizada. Além disso, todos os centros de processamento de dados têm necessidades específicas, como um local para abrigar os servidores, energia elétrica, pessoal para a manutenção e uma infraestrutura de rede de dados para interligar os servidores.

Primeiramente os servidores são acondicionados em grandes armários chamados de *racks*. Os *racks* são dispostos no data center formando corredores, de acordo com a melhor distribuição possível. Cada *rack* tem uma conexão de energia elétrica, que será distribuída para todos os servidores, e um ou mais enlaces de redes, que também serão distribuídos entre os servidores daquele *rack*.

Como normalmente o número de enlaces que chega a um *rack* é menor do que o número de servidores que existe nesse *rack* é necessário um meio de compartilhar esses enlaces. O equipamento utilizado para compartilhar os enlaces de rede é chamado de comutador, ou *network switch*. Um comutador funciona recebendo dados na forma de pacotes através de um enlace e encaminhando esses pacotes somente para a porta em que esse dado deve ser enviado.

Comutadores que ficam em um *rack*, normalmente localizado no topo dos *racks*, são denominados de comutadores de topo de *rack* ou *Top of Racks switches* ou ainda pela sigla ToR. Os outros comutadores são denominados de acordo com a sua posição na hierarquia dentro do centro de processamento de dados.

O equipamento responsável pela conexão de dados com o exterior do centro de processamento de dados é o roteador de borda. Esse equipamento é um roteador capaz de mapear os endereços internos do centro de processamento de dados com as requisições externas dos clientes. Esse roteador normalmente é ligado ao núcleo da rede, conseguindo uma melhor comunicação com todos os servidores do data center.

2.2 Topologias

As análises de data centers disponíveis na literatura para consulta e disponibilizadas pelas empresas fornecedoras de equipamentos mostram que algumas topologias são mais utilizadas, tanto por custo como por serem mais facilmente implementadas, gerenciadas ou por motivos específicos, como a ampliação de uma estrutura legada, política de uso de soluções de um determinado fabricante ou por receio na

implementação e no suporte de novas tecnologias. Não existem padronizações de formato da topologia ou uma topologia que atenda perfeitamente a todos os casos, cada situação exige uma topologia.

Segundo a literatura, [21–28] a topologia mais difundida em data centers é a de árvore, representada pela Figura 3.1, onde os servidores são interligados por comutadores seguindo o padrão de árvore com altura de dois ou três níveis. Os servidores são as folhas e existem dois ou três níveis de comutadores interligando esses servidores. A fabricante Cisco [20] mostra em seu guia de projeto de infraestruturas de data centers as melhores práticas para interligar os equipamentos dentro de um data center. Nesse guia a topologia indicada é a de árvore, com variações de capacidade de tráfego no núcleo e do tamanho da rede.

As outras topologias propostas pela literatura para uso em data center tentam melhorar a disponibilidade de comunicação entre os servidores sem aumentar muito o gasto com equipamentos ou melhorando o encaminhamento de pacotes dentro da topologia, formando múltiplos caminhos e conseguindo um nível maior de redundância. Propostas como *fat-tree* [21], representado na Figura 3.3, aumentam o número de enlaces dos níveis mais altos sem precisar utilizar equipamentos de custo elevado. Outras propostas como o BCube [25], representado na Figura 4.5, baseiam-se em estruturas definidas recursivamente, interligadas de maneira a permitir a ampliação do data center enquanto é uma arquitetura robusta, com múltiplos caminhos e tolerância a falhas. Cada topologia é indicada para um determinado ambiente, sendo que cada uma delas possui suas peculiaridades quanto ao endereçamento dos servidores e do roteamento de pacotes pela rede. Essas e outras topologias serão melhor descritas na Seção 3.1.

2.3 Tráfego

Devido à alta complexidade, às diversas topologias existentes e principalmente à variedade de aplicativos que são executados nesse ambiente, diversos aspectos das redes de data center são analisados na literatura. Kandula *et al.* [26] estudam um grande data center que executa aplicações do tipo MapReduce. Eles coletam dados de rede e descobrem que nas topologias de árvore 86% dos comutadores que fazem parte da camada do núcleo e da agregação (as topologias são apresentadas com mais detalhes na Seção 3.1) apresentam algum período de congestionamento maior do que 10 segundos.

Os métodos comumente usados para inferir a matriz de tráfego através de dados SNMP (Simple Network Management Protocol) dos comutadores não produzem resultados satisfatórios, pois o protocolo SNMP somente registra o número de pacotes ou o número de bytes que fluem em uma porta. Além disso, o intervalo entre as re-

quisições dos dados deve ser alto, em torno de cinco minutos, para garantir o correto funcionamento do equipamento. Assim, a aquisição de dados não tem a acurácia necessária para mapear a matriz de tráfego real. Nesse cenário, ao comparar com os dados em nível de pacote, existe um erro médio de 60% na estimação do tráfego.

Benson *et al.* [27] estudam data centers com topologias de árvore. Ao analisar data centers com árvores em 3 camadas eles conseguem identificar que existe uma concentração do tráfego em comutadores de topo de *rack*.

O estudo também revela que padrões de virtualização e consolidação de máquinas virtuais ainda não são evidenciados e os operadores escolhem a localização onde os serviços serão instalados, ou seja, não são posicionados de forma aleatória.

Também não foram encontradas evidências que comprovem que exista um algoritmo ou que exista um padrão para esse posicionamento. Nesses data centers os comutadores de núcleo têm uma grande utilização, com uma elevada variação em um mesmo dia. Já comutadores da camada de agregação e de topo de *rack* apresentam uma menor utilização e uma menor variação no tráfego. Esse trabalho mostra que o posicionamento de máquinas virtuais em data centers é importante e deve ser diferente de acordo com o tipo de serviços implementados no data center. Também mostra que a distribuição de tráfego é dinâmica, sendo bastante modificada ao decorrer do dia.

Em outro trabalho, Benson *et al.* [28] analisam 19 centros de processamento de dados com diferentes topologias e aplicações, usando tanto dados coletados por SNMP como dados ao nível de pacotes. Os data centers analisados têm topologia de árvore, com duas ou três camadas e os aplicativos que são executados sobre cada data center são distintos, fazendo com que o comportamento do tráfego seja diferente em cada um deles. Entre outras observações, eles percebem que os fluxos de dados chegam em um enlace com a característica de Ligado/Desligado (ON/OFF). Ou seja, existem momentos em que muito tráfego está percorrendo um enlace, seguido por um momento em que pouco tráfego está sendo transmitido pelo enlace. Esse tráfego pode ser caracterizado por uma curva de distribuição log-normal. Essas conclusões são utilizadas para modelar o gerador de tráfego descrito no Capítulo 6.

2.4 Algoritmos propostos na literatura

Outras propostas sugerem a otimização do posicionamento de serviços através da localização de máquinas virtuais em um data center. O Sandpiper [32] implementa de forma automática a detecção e a migração de máquinas virtuais que estão com os seus recursos de memória, CPU ou de interface de rede sobrecarregando a máquina hospedeira, desde que existam recursos em outros servidores do data center. O Sandpiper tenta identificar e solucionar sobrecargas de processamento, uso de memória

e interface de rede monitorando o uso de cada um dos recursos em cada um dos servidores e das máquinas virtuais, migrando as máquinas virtuais necessárias para aliviar o uso de um servidor, desde que exista essa capacidade disponível em outro servidor. Diferente do presente trabalho, o Sandpiper não leva em consideração o tráfego de rede na escolha do posicionamento das máquinas virtuais.

Ferreto *et al.* [35] discutem a migração de máquinas virtuais baseadas somente nos recursos necessários pela CPU. A abordagem é diferente das anteriores pois eles focam na migração de máquinas virtuais que apresentam mais variabilidade no uso de recursos, dada uma janela escolhida de tempo. Ao dar preferência às migrações de máquinas virtuais com maior variabilidade de consumo de CPU, reduzindo as migrações das que apresentam sempre o mesmo padrão de consumo de CPU, o número de migrações pode ser reduzido. Escolher migrar máquinas virtuais que estão aumentando de tamanho ou diminuindo de tamanho ajuda a atenuar o problema de desempenho, pois se a máquina virtual estiver precisando de recursos movê-la fará com que no futuro ela tenha os recursos necessários, e se uma máquina virtual está diminuindo, a sobrecarga da migração pode ter efeitos mínimos sobre a carga da máquina. Os autores desenvolvem uma solução baseada em programação linear, que tem um tempo de solução muito elevado, e uma heurística com menor tempo de solução. Os resultados dos dois algoritmos são próximos, tendo a otimização resultados melhores, mas a heurística usa somente uma fração do tempo necessário para a análise. O tempo necessário para a convergência da solução baseada em programação linear chega a 5h enquanto a heurística executa o mesmo processo em 5 segundos. Não foi definido se o algoritmo escala em tamanho, já que as simulações foram feitas usando 61 cargas de trabalho (jobs), sendo esses trabalhos alocados a no máximo 56 servidores.

Meng *et al.* [34] discutem o posicionamento de serviços através do posicionamento de máquinas virtuais nos centros de processamento de dados. Baseados nos custos fixos de banda requerida por cada máquina virtual e no custo de comunicação entre as posições das máquinas virtuais dentro de um data center, eles formulam um problema de minimização de acordo com um algoritmo de menor corte para achar as divisões e a localização correta das máquinas virtuais. Exigências de recursos de memória e de CPU não são usadas nos cálculos dos algoritmos, porque eles assumem que cada máquina virtual tem o mesmo tamanho e que cada servidor do centro de processamento de dados tem uma capacidade fixa de máquinas virtuais que podem ser alocadas a ele. Meng *et al.* mostram que o problema de posicionamento de máquinas virtuais levando em consideração o tráfego de rede é NP-Difícil, não sendo solucionado em tempo polinomial em função da entrada. Para a análise são utilizados dois data centers. No primeiro são capturados as taxas das transmissões entre as 17 mil máquinas virtuais que o compõe. No segundo são capturados dados

dos pacotes TCP de 68 máquinas das centenas instaladas neste data center. Entre outras conclusões, ao analisar os dados eles observam que no caso do menor data center (68 máquinas virtuais) a latência não tem uma forte correlação com a matriz de tráfego, indicando que a latência não é uma boa medida quando se quer controlar a demanda de tráfego. O algoritmo foi desenvolvido para ser executado de forma *offline*, não conseguindo responder rápido o suficiente à mudança na demanda, visto que o tempo para convergência chega a mais de 10 minutos em uma simulação com 1.024 nós.

Wang *et al.* [36] propõem um modelo de consolidação de máquinas virtuais com restrições de banda de comunicação impostas por dispositivos de rede, como comutadores e interfaces de rede dos servidores que hospedam as máquinas virtuais. Nesse trabalho as máquinas virtuais com demandas de tráfego conhecidas, que devem ser previamente calculadas, são consolidadas em um número de servidores com limite de capacidade idêntico ao necessário. O modelo não está preocupado com a engenharia de tráfego do núcleo da rede, somente modelando a consolidação das máquinas virtuais em cada um dos servidores da rede do centro de processamento de dados. O modelo não utiliza requisitos de CPU, memória ou de tráfego no núcleo em seus cálculos.

Biran *et al.* [38] descrevem um problema de minimização usando cortes em redes de grafos para determinar a localização de máquinas virtuais em um data center, baseado no consumo dos recursos da rede, de CPU e memória de cada máquina virtual. A formulação é complexa e não escala com o tamanho, então os autores também criaram duas heurísticas baseadas no algoritmo de minimização proposto. Para o funcionamento do algoritmo é necessária a transformação da topologia do data center em uma árvore para depois achar todos os cortes críticos e então aplicar a matriz de tráfego nessa transformação, achando o melhor posicionamento das máquinas virtuais. Maneiras de aplicar o algoritmo a topologias diferentes de árvores, como a BCube, não são descritas no trabalho e o cálculo da transformação em uma árvore não pode ser feito sob demanda, já que o gasto computacional para a conversão é alto. A cada pequena mudança na topologia ou no número de servidores, o algoritmo de transformação deve ser analisado novamente antes de aplicar o algoritmo de posicionamento, elevando o custo de uso do algoritmo. Para diminuir a complexidade e consequentemente o tempo de convergência do algoritmo, os autores supõem que cada usuário tem um conjunto de máquinas virtuais específicas e que somente as máquinas virtuais que fazem parte de um mesmo grupo podem se comunicar, então todas as máquinas virtuais já são previamente agregadas antes do início da análise do posicionamento, fazendo com que o algoritmo não possa ser executado em um ambiente onde máquinas virtuais não são diferenciadas por usuários ou por grupo de trabalho. O algoritmo proposto nessa dissertação não usa a divisão em grupos

para facilitar os cálculos do algoritmo, ao invés disso ele tenta localizar quais são os grupos de máquinas virtuais que mais trocam informações.

Bodik *et al.* [12] estudam o problema de posicionamento de máquinas virtuais em centros de processamento de dados utilizando dois aspectos simultaneamente, a influência do posicionamento no uso do núcleo da rede, medido através da banda utilizada no núcleo, e a minimização de falhas catastróficas devido à falta de energia elétrica nas diferentes zonas do data center. Eles analisam o tráfego de um *cluster* de testes, aumentando os dados de tráfego adquiridos para escalar o tamanho do data center. Nesse estudo eles concluem que existe uma correlação forte entre algumas máquinas virtuais. Também observam que após a remoção de 10% das arestas que menos transportam tráfego o grafo passa a ter alguns grupos distintos de máquinas virtuais que trocam uma quantidade maior de dados entre si. Baseados na propriedade de K-cortes mínimo[1] e fazendo trocas de posições das máquinas virtuais dentro do data center eles propõem a minimização de uma função de tolerância a falhas e de tráfego do núcleo. Três heurísticas variadas do algoritmo são propostas, conseguindo redução de até 60% do tráfego do núcleo. A análise do algoritmo é feita somente para um *cluster* voltado para serviços *web* de testes da Microsoft. Toda a análise é feita para topologias do tipo árvore e uma maneira para implementar o algoritmo em outros tipos de topologia não é apresentada, nem como são feitas as trocas de máquinas virtuais. Não são levados em consideração os recursos de CPU e memória consumida pelas máquinas, na proposta as máquinas só podem consumir uma quantidade pré-determinada de recursos.

Nesta dissertação é apresentada uma proposta para posicionamento de máquinas virtuais em ambientes de data center através da criação do Algoritmo de Posicionamento de Serviços (APoS), sendo uma ampliação da proposta presente no artigo *Online Traffic-aware Virtual Machine Placement in Data Center Networks* [42]. A partir de uma modificação no algoritmo de clusterização proposto por Girvan-Newman, o APoS localiza máquinas virtuais que trafegam uma grande quantidade de dados entre elas, agrupando-as em comunidades. Essas comunidades devem ser alocadas à partições determinadas pela topologia do data center, aproveitando o uso de áreas com alta banda de conexão. A alocação é feita a partir de heurísticas de empacotamento (*bin-packing*).

O APoS se diferencia das outras propostas pois trata tanto da classificação das máquinas virtuais em unidades que se comunicam intensamente quanto do correto posicionamento dessas unidades, alocando-as nos servidores físicos corretamente, de acordo com a capacidade de hospedar esse serviço e minimizando o impacto

[1]Na matemática, o k-corte mínimo é um problema de otimização combinatória que requer encontrar um conjunto de arestas cuja remoção particiona o grafo em k componentes conectados. Estas arestas são referidas como k-corte. O objetivo é encontrar o conjunto dessas arestas que compõe o k-corte com o menor peso.

para o data center como um todo. Além disso, para permitir o uso em ambientes com grandes mudanças no padrão do tráfego, o tempo de execução do algoritmo é pequeno o suficiente para permitir o seu uso em ambientes com mais de 16.000 máquinas virtuais, conforme será demostrado no Capítulo 6 de resultados.

A Tabela 2.1 mostra as principais características das diversas propostas de posicionamento de máquinas virtuais em centros de processamento de dados. Os trabalhos, embora de diversas áreas e também abordando outros problemas, analisam aspectos importantes para a operação de centros de processamento de dados, sempre com a preocupação de criar um ambiente mais eficiente e economicamente viável. O uso de algoritmos para posicionamento de serviços, seja através de migrações de máquinas virtuais que contêm cada um dos serviços individualmente ou através do correto posicionamento de serviços que executam sobre um servidor físico é importante pois ajuda na redução do uso do núcleo da rede do data center, melhorando a troca de dados que trafegam pela rede, possibilitando o aumento do número de serviços executando no data center ou adiando a necessidade da ampliação da capacidade de comunicação do núcleo da rede de um data center.

No próximo capítulo será estudado o ambiente em que o algoritmo é executado, ou seja, o ambiente dos data centers.

Tabela 2.1: Comparação das propostas de posicionamento de máquinas virtuais

Propostas	**Recursos**	**Escala das simulações**	**Grupos de VMs**	**Métodos de cálculo do posicionamento**
Wood *et al.* (Sandpiper)	CPU e Mem.	35 VMs	Não	Arcabouço para a migração de VMs
Ferreto *et al.*	CPU	264 jobs	Não	Heurística para migrar VMs em crescimento/decrescimento
Meng *et al.*	Tráfego da rede	68 VMs	Sim	Algoritmo para calcular menor corte do grafo
Wang *et al.*	Interface de rede	9k VMs	Não	Consolidação de VMs, não otimiza núcleo
Biran *et al.*	CPU, Mem. e Tráfego de rede	3k VMs	Sim	Heurísticas de menor corte do grafo
Bodik *et al.*	Tráfego de rede e Energia	?	Não	K-cortes com *swap* de serviços por minimização de uma função custo
Dias e Costa (APoS)	CPU, Mem. e Tráfego de rede	16k VMs	Não	Heurística para encontrar e empacotar comunidades de VMs

Capítulo 3

Arquiteturas de Data Center

Este capítulo apresenta as principais características do ambiente do data center. Inicialmente é apresentada a topologia em árvore, a mais usada segundo a literatura, e em seguida são apresentadas as novas topologias propostas na literatura, observando suas vantagens e desvantagens quanto à concentração de tráfego em enlaces. Na segunda subseção apresentam-se as ferramentas e os serviços mais utilizados em data centers, como os serviços são alocados e o porquê da necessidade de aproximar os serviços que são executados em um data center, com suas vantagens e suas desvantagens.

3.1 Topologias de redes de data center

Segundo a literatura, a topologia mais utilizada em centros de processamento de dados é a árvore [21–28] com duas ou três camadas. Nessa topologia representamos os servidores ou um conjunto de servidores como folhas de uma árvore onde cada camada acima desta é composta por comutadores. As ligações entre as estruturas da árvore simbolizam as ligações físicas entre os comutadores de um data center. A Figura 3.1 representa uma forma simples para se ligar servidores em uma topologia de árvore usando três camadas.

Os servidores normalmente são fisicamente agrupados em *racks* dentro de um data center, sendo esses ligados a um mesmo comutador chamado de topo de *rack*. O comutador de topo de *rack* representa a primeira camada da árvore. Na segunda camada, denominada agregação, temos comutadores que conectam os diversos comutadores de topos de *rack*, sendo normalmente a banda dos enlaces entre esses dois comutadores mais elevada do que a ligação entre um servidor e um comutador, o que permite uma comunicação com maior banda no data center. Em topologias de árvore com três camadas, temos uma terceira camada denominada núcleo, responsável por agregar o tráfego de todos os comutadores da agregação.

Em topologias com somente duas camadas não existe a camada de agregação,

sendo o núcleo o responsável por ligar todos os comutadores de topo de *rack*. Ao subir na árvore a quantidade de enlaces é reduzida mas a banda de cada ligação deve ser maior.

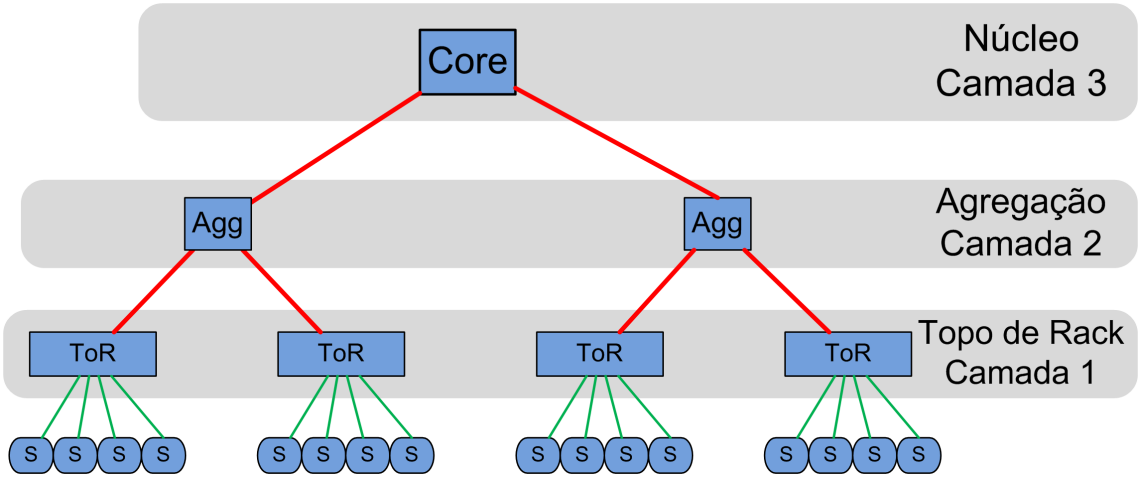


Figura 3.1: Data center representado por uma árvore.

3.1.1 Árvore

Em topologias de árvore não é possível manter a capacidade de comunicação plena entre todos os servidores devido a alguns fatores. Quanto maior é a banda de um enlace, mais caro são os equipamentos necessários para essa ligação. Também existe uma limitação tecnológica para a capacidade dos enlaces. Na escolha da implementação do data center fatores relacionados ao custo ou a viabilidade técnica fazem com que seja necessária a diminuição da banda dos enlaces entre os comutadores de diferentes níveis. Assim a capacidade disponível para a ligação com as camadas superiores nem sempre é igual a soma de todos enlaces que são ligados àquela camada.

A capacidade de transmissão nas camadas de agregação e do núcleo, apesar de ser bastante elevada com múltiplos enlaces de até 10Gbit/s, é menor do que a soma da capacidade de comunicação de todos os servidores, ou seja, se todos os servidores tentarem se comunicar através da agregação e/ou do núcleo ao mesmo tempo eles não conseguirão atingir a velocidade máxima de comunicação. Essa diferente capacidade em cada camada é chamada de fator de *oversubscription*.

Topologias em árvore têm pontos fracos, sendo os pontos importantes para esse trabalho a falta de redundância e a diminuição da banda agregada ao subir nas camadas da árvore. Na raiz, chamada de núcleo da rede, a banda disponível para transportar dados entre os ramos da árvore é bem menor do que o total que pode ser enviado pelos servidores. Em um caso onde todos os servidores se comunicam

com outros servidores há congestionamento.

Usando a topologia da Figura 3.1 e definindo que os enlaces são de 1Gbit/s e existem 16 servidores (folhas) nessa topologia, quatro comutadores de camada 1 e dois comutadores de camada 2, e um comutador de camada 3, podemos calcular qual é a velocidade real que cada servidor tem disponível para se comunicar.

Baseado nas informações da Cisco [20], para calcular o fator de *Oversubscription* conseguimos determinar que: a camada 2 tem um fator de redução de 2 para 1 (2:1), chegam dois enlaces e sai somente um enlace; e a camada 1 tem um fator de redução de 4 para 1 (4:1), tem quatro servidores conectados a cada comutador e somente um enlace que leva a camada superior. Para calcular o total somamos os dois valores, chegando a um fator de *Oversubscription* de 6:1 para toda a árvore.

Para calcular a banda disponível, temos que cada servidor consegue se comunicar a 1Gbit/s com a camada acima, porém essa camada tem somente 1Gbit/s de comunicação com a próxima camada, fazendo com que cada servidor possa se comunicar a 250Mbit/s. Ao analisar a topologia como um todo, esses 250Mbit/s ao passar pela agregação e o núcleo têm o seu valor diminuído pela metade (2:1), então a banda disponível para uma máquina é de 125Mbit/s, oito vezes menor que a banda do enlace nas folhas.

Existem formas fáceis de atenuar esses problemas na topologia de árvore. Ao aumentar a velocidade dos enlaces das camadas mais altas, por exemplo passando de 1Gbit/s para 10Gbit/s, é possível melhorar o fator de *Oversubscription* e a velocidade total. Uma maior banda e múltiplos caminhos para conectar os servidores também são possíveis ao aumentar o número de enlaces nas camadas mais altas. Como exemplo, usando as indicações do guia de projetos de data centers da Cisco [20], é possível criar uma topologia com um fator 416 Mbit/s, através de enlaces de 10Gbit/s no núcleo, com dois comutadores de núcleo com 32 portas de 10Gbit/s ligados a cada um dos 32 comutadores de topo de *rack*, com capacidade para 48 servidores, chegando a 1.536 computadores interligados a 416Mbit/s. A Figura 3.2 retirada do guia de projetos de data center [20] ilustra essa montagem.

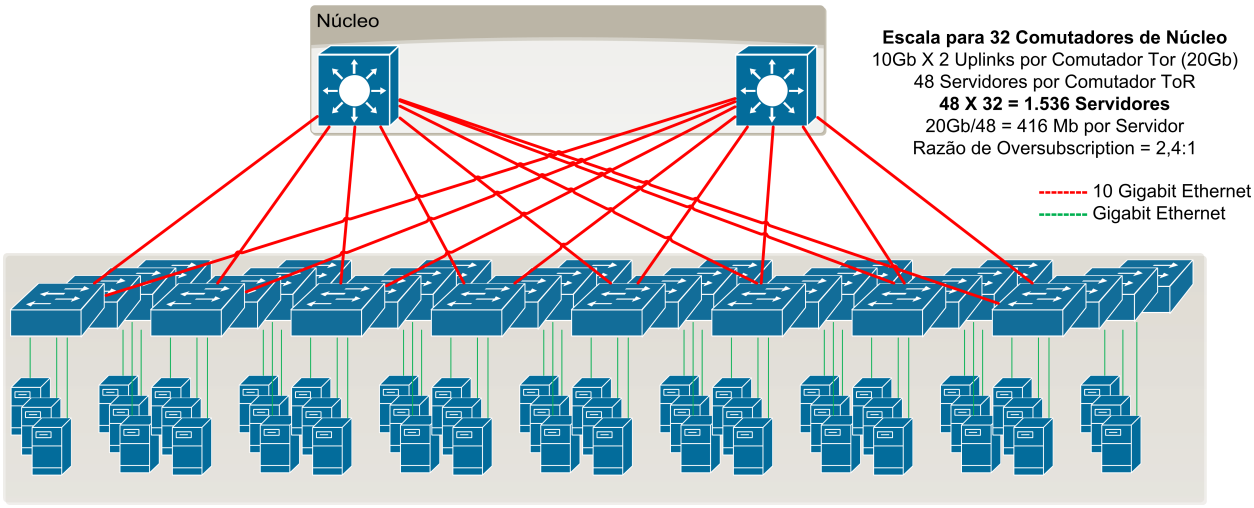


Figura 3.2: Projeto de data center. Adaptado de [20].

Existem diversas propostas com diferentes alternativas à topologia de árvore tradicional. Apesar de não serem largamente implementadas em data center de produção são bastante difundidas na literatura e em montagens experimentais. Existem quatro propostas que se destacam, são elas: a *fat-tree* [21], VL2 [22], PortLand [23], Bcube [25]. Descrevemos as características dessas principais propostas a seguir.

3.1.2 *Fat-tree*

Al-Fares *et al.* [21] propõem uma nova abordagem para a topologia de árvore *fat-tree* [43]. Eles se baseiam no uso de comutadores *commodity* de custo mais baixo do que os usados no núcleo das redes de data center. Eles propõem um novo esquema de ligação que com o custo mais baixo consegue atingir uma banda de bisseção[1] igual a de uma árvore comumente encontrada em centros de processamento de dados. O algoritmo desenvolvido para criar a topologia tem a capacidade de criar ambientes mais conectados, com uma maior banda de bisseção, mas para isso precisa modificar a topologia de rede, aumentando o número e o custo dos equipamentos de rede.

A proposta consiste em aumentar o número de conexões internas e de comutadores do data center de uma maneira econômica. Para isso, a *fat-tree* utiliza *commodity switches* ou comutadores simples, que não possuem recursos avançados de roteamento ou gerenciamentos robustos. Como o custo do cabeamento representa menos de 10% do custo da implementação de um data center [44], essa topologia é

[1]A banda de bisseção consiste no valor da banda entre dois segmentos de mesmo tamanho de uma rede. Normalmente se refere à segmentação no pior caso, obtendo o menor valor possível para a banda na divisão da rede em dois segmentos.

mais econômica do que a implementação de equipamentos mais robustos no núcleo da rede. A Figura 3.3, adaptada do artigo [21], ilustra a topologia *fat-tree*.

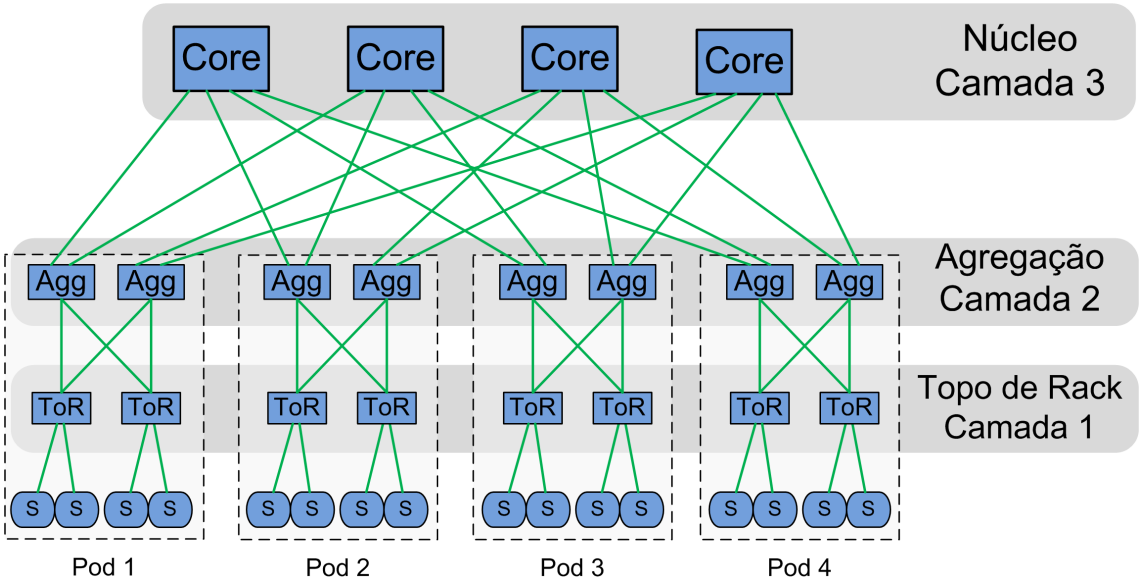


Figura 3.3: Topologia *fat-tree*. Adaptado de [21].

A regra de formação da topologia consiste em criar *pods* que são um conjunto de comutadores formados por duas camadas, a de borda e a de agregação. Na primeira camada, os comutadores de borda usam metade das portas para se conectar aos servidores e a outra metade das portas é usada para se ligar aos comutadores de agregação. Esta segunda metade das portas da camada de agregação é conectada aos comutadores do núcleo, uma ligação para cada comutador, precisando da metade do número de portas em comutadores de núcleo. Então, existe o mesmo número de comutadores na camada de agregação e na camada de borda, com o limite do número de servidores e o número de *pods* definidos pelo número de portas utilizadas dos comutadores.

Essa topologia permite o uso de múltiplos caminhos usando enlaces com a mesma banda. No entanto, apresenta restrições quanto ao endereçamento dos servidores, precisando de um algoritmo de roteamento sofisticado, capaz de escolher entre os múltiplos caminhos disponíveis e capaz de lidar com o endereçamento fixo, atrelado ao posicionamento relativo de cada um dos *pods* na topologia. A migração entre servidores nessa topologia precisa de cuidados adicionais, modificando os endereços das máquinas migradas, já que cada servidor precisa de um endereçamento fixo e dependente da posição na topologia.

3.1.3 PortLand

Tentando melhorar a questão de roteamento e de endereçamento na *fat-tree*, Nirajan *et al.* propõem o PortLand [23]. O Portland modifica o endereçamento e o roteamento em árvores de multicaminhos, como uma árvore *fat-tree*. Através da criação de um arcabouço para controlar o endereçamento, a resolução de endereços e o roteamento na camada de enlace, o PortLand cria ferramentas para um maior controle dos operadores do data center, garantindo que o endereçamento IP de um servidor ou de uma máquina virtual não precise ser modificado ao ser migrado. Através de algoritmos e protocolos eficientes, a proposta melhora a flexibilidade, permitindo uma melhor distribuição de endereços IPs, permitindo uma escolha mais rápida de rotas para fluxos e uma tolerância a falhas, recuperando-se rapidamente de problemas que podem acontecer nos enlaces.

O PortLand baseia-se no roteamento utilizando endereços da camada de enlace do *Ethernet*, onde são criados endereços MACs virtuais para representar os dispositivos. Essa endereçamento é feito por um controlador centralizado, capaz de receber requisições de tradução de endereçamento de todos os componentes da rede. Como esse servidor controla todos os endereçamentos, é possível garantir um ambiente sem *loops* e com o uso de múltiplos caminhos.

Nessa solução ao atingir um ponto crítico onde algum enlace da rede é sobrecarregado, a única forma de melhorar o desempenho da rede, sem recorrer a métodos de engenharia de tráfego, é ampliar a topologia através da introdução de novos equipamentos e de novos enlaces.

3.1.4 VL2

Baseados também em árvores do tipo *fat-tree*, Greenberg *et al.* propõem o VL2 [22]. A principal diferença em relação a *fat-tree* original é o endereçamento e o roteamento de caminhos dentro do data center, que através da escolha balanceada dos enlaces que um pacote deve seguir consegue um melhor aproveitamento dos recursos.

Através de endereços IPs diferenciados, uma faixa para os servidores e uma outra faixa para os equipamentos de encaminhamento da rede e um controlador central, cada um dos servidores é associado a um endereço do comutador de topo de *rack* ao qual o servidor esta conectado. Dessa forma o roteamento é feito somente pelos endereços dos equipamentos de rede sendo o comutador de topo de *rack* responsável por resolver o servidor correto quando os pacotes de dados chegam. Como desvantagem temos que o VL2 precisa de um controlador central para resolver endereços e o empacotamento dos pacotes, para que os comutadores de borda consigam encaminhar corretamente os pacotes aos servidores.

Outra característica da proposta é o uso de comutadores com enlaces de maior capacidade na agregação e no núcleo da rede, conseguindo obter um desempenho maior usando menos equipamentos de rede do que a proposta original da *fat-tree* e consequentemente conseguindo um roteamento com menor variação de caminhos. Do mesmo modo que as topologias anteriores, não existe possibilidade de otimização se o data center contiver alguma porção congestionada, precisando de ampliação da capacidade dos equipamentos para diminuir o congestionamento.

3.1.5 DCell

Guo *et al.* propõem a DCell [24], uma nova topologia baseada em recursão, onde são criadas células de múltiplos níveis. Nas células de níveis mais baixos existem n ligações internas, feitas através de um comutador. O próximo nível é formado por $n + 1$ células ligadas através de uma outra interface de rede de cada um dos servidores. Para os próximos níveis são adicionadas novas portas de rede e novas células.

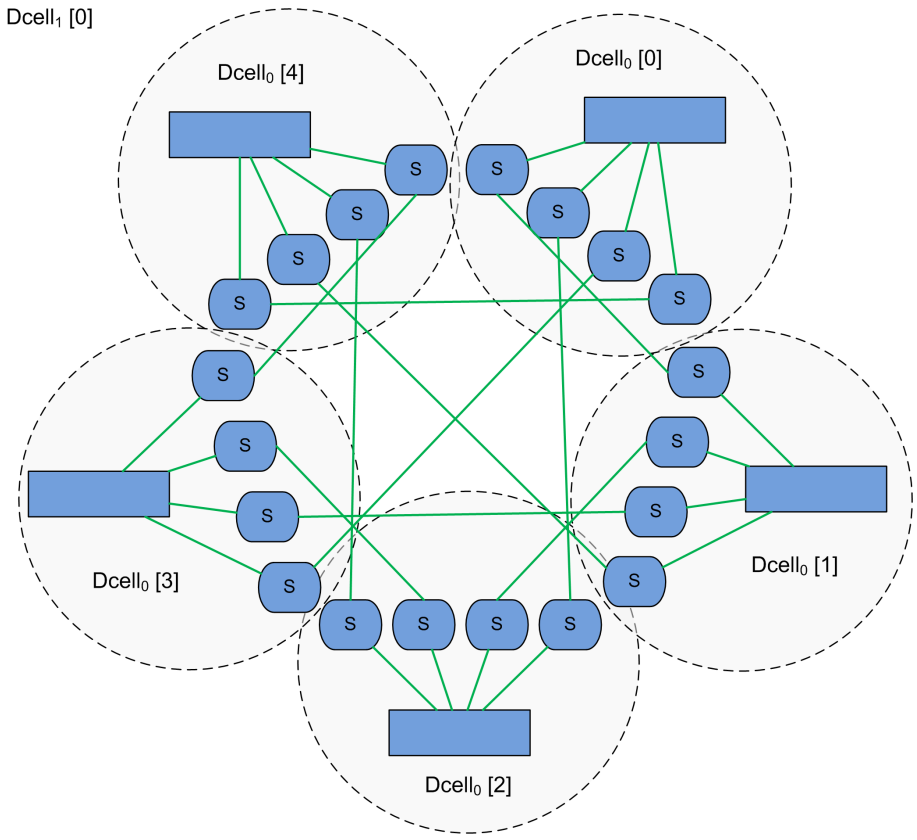


Figura 3.4: Topologia DCell. Adaptado de [24].

A Figura 3.4 ilustra uma rede em que cada célula possui quatro servidores existindo, então, cinco células interligadas. O número máximo de máquinas é definido

através do número de portas que os servidores e os comutadores podem ter. Com seis servidores por célula, com três níveis de células é possível criar um estrutura para mais de três milhões de servidores.

Com a topologia DCell também é proposto um algoritmo de roteamento, onde cada conexão entre células é feita entre um servidor de cada uma das células, sendo o servidor responsável por encaminhar o tráfego. Como o servidor é responsável por rotear e por manter as tabelas de roteamento, alguma fatia de processamento de cada um dos servidores é utilizada, diminuindo a capacidade de processamento total do data center. Um método de endereçamento entre a camada de enlace e a camada de rede (camada 2.5) foi criado para permitir o roteamento entre os servidores e permitir *multicast*.

O autor mostra que a proposta tem duas vezes mais banda agregada em relação a uma árvore tradicional de duas camadas. Porém, se o padrão de tráfego necessitar de acesso a todas as máquinas, o número reduzido de conexões que saem de cada célula faz com que o desempenho seja reduzido. Como desvantagem temos que para aumentar a capacidade é necessário, além do aumento do número de comutadores e de enlaces, o aumento do número de placas de rede em cada um dos servidores.

3.1.6 BCube

Guo *et al.* posteriormente propuseram a BCube em [25]. Essa topologia foi desenhada especificamente para centros de processamento de dados baseados nos servidores em contêineres, onde os data centers são montados em módulos, somente precisando de energia e conexão de rede para conectar os contêineres ao data center. No nível mais baixo, BCube_0, a construção consiste em ligar os n servidores de uma BCube_0 a um comutador. No próximo nível, BCube_1, uma placa de rede é adicionada aos servidores que fazem parte do BCube e esses são interligados através de comutadores. Cada servidor de um BCube_0 é ligado a um comutador diferente, fazendo-se necessário n comutadores. A Figura 4.5, adaptada da proposta [25], ilustra a montagem da topologia proposta.

Segundo os autores, comparada com a DCell, a BCube não tem gargalos de desempenho e fornece uma capacidade maior de rede. Se comparada com a topologia de árvore *fat-tree*, a BCube apresenta melhor comunicação de um com vários servidores ao mesmo tempo, além de ser construído com comutadores simples.

Para escolher os caminhos, a BCube propõe um protocolo de endereçamento (camada 2.5) e um novo protocolo de roteamento. Na camada de enlace é usado o *spanning tree* e para o endereçamento e o roteamento foi criado um endereçamento baseado na posição na topologia, com o encaminhamento de pacote feito pelos servidores de modo semelhante à DCell e a camada de rede é baseada em endereços

IPs. No caso de congestionamento de enlaces não existe possibilidade de otimização se o data center tiver alguma porção congestionada, precisando de ampliação da capacidade dos equipamentos para diminuir o congestionamento.

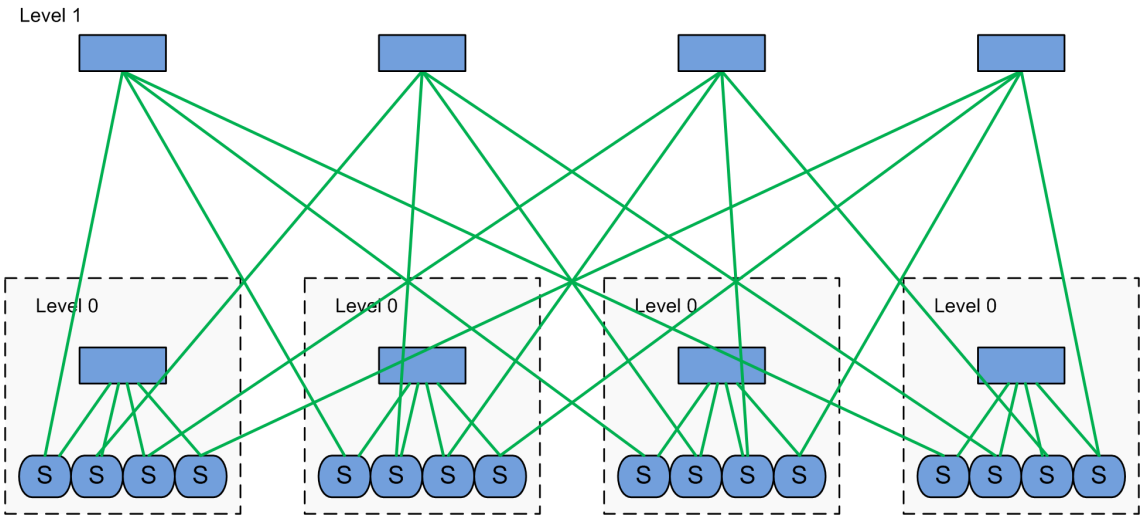


Figura 3.5: Topologia BCube. Adaptado de [25].

As principais características sobre as topologias apresentadas são exibidas na Tabela 3.1. Nessa tabela temos os nomes das propostas, o tipo de topologia utilizada, a possibilidade de uso de múltiplos caminhos para a conexão entre dois servidores, o tipo de endereçamento utilizado e o tipo de roteamento necessário para o encaminhamento correto de pacotes. Análises mais profundas são encontradas na literatura, como em [45], onde Couto *et al.* analisam a robustez das principais topologias de data center.

Tabela 3.1: Comparação das topologias de data center

Propostas	**Topologia**	**Múltiplos caminhos**	**Endereçamento**	**Roteamento**
Árvore	Árvore	Não	IPs fixos	Roteadores
Fat-Tree	*Fat-tree*	Sim	IPs fixos	Comutadores
Portland	*Fat-tree*	Sim	MACs Virtuais	Comutadores + Controlador centralizado
VL2	*Fat-tree*	Sim	IPs distintos	Comutadores + Controlador centralizado
DCell	Recursiva	Sim	Camada 2.5	Servidores como roteadores
Bcube	Recursiva	Sim	Camada 2.5	Servidores como roteadores

3.2 Ferramentas e serviços utilizados em data centers

As ferramentas utilizadas nos data centers têm o objetivo de monitorar e controlar, ajudando na administração de grandes data centers. O controle de um data center objetiva alcançar as melhores taxas de uso de recursos sem que haja algum prejuízo para o que está sendo executado nesse data center. Ou seja, ao controlar um data center procura-se maximizar o número de processos executados em um determinado período gerando uma oportunidade de maiores ganhos econômicos para o provedor do serviço. As principais ferramentas utilizadas para obter esse controle são apresentadas na Seção 3.2.1.

Os serviços executados sobre um data center são os mais diversos. Devido à grande quantidade de data centers [14] e à sua natureza privada, não é possível obter uma lista exaustiva de todos os seus usos e aplicações. Assim, na Seção 3.2.2 analisamos os principais serviços listados na literatura.

3.2.1 Ferramentas

Algumas ferramentas ajudam a obter melhor retorno dos investimentos feitos em data centers. A virtualização, que permite o isolamento de serviços enquanto executa diversas instâncias em um mesmo *hardware*, é a principal das ferramentas usadas em data centers comerciais para aumentar a produtividade dos mesmos. A virtualização, ao fazer a interface entre o *hardware* e o *software*, permite que as máquinas virtuais sejam movidas entre diferentes servidores. Esse processo, chamado de migração, é o que permite ao data center atender a demandas de processamento flexíveis.

Virtualização

A virtualização é uma técnica que separa a aplicação e o sistema operacional do *hardware*. A virtualização é composta por uma camada denominada hipervisor, que é um sistema operacional instalado diretamente sobre o *hardware* com um *software* específico capaz de prover recursos computacionais a uma camada superior. Nessa camada superior estão as máquinas virtuais, que recebem os recursos virtuais de CPU, memória, armazenamento de disco, interface de rede e outros dispositivos de entrada e saída do hipervisor, como ilustrado na Figura 3.6.

Com a virtualização vários sistemas operacionais podem rodar simultaneamente em um mesmo servidor, sem que esses sistemas tenham conhecimento que estão sendo executados sobre o mesmo *hardware*. O hipervisor cria cada um dos *hardwares*

virtuais isoladamente, dessa forma uma máquina virtual não consegue interferir diretamente nos processos executados em outra máquina virtual.

O controle da distribuição dos recursos é realizado pelo hipervisor, o responsável pelo gerenciamento de recursos que cada máquina virtual receberá e também responsável pela criação, pelo desligamento e pela migração das máquinas virtuais. O hipervisor pode alocar dinamicamente recursos para cada máquina virtual, aumentando e diminuindo conforme necessário.

O hipervisor Xen [13] é capaz de alocar dinamicamente recursos de memória e CPU [46–48]. Outros recursos, como os de entrada e saída não podem ser controlados diretamente, já que são compartilhados entre todas as máquinas virtuais e o hipervisor.

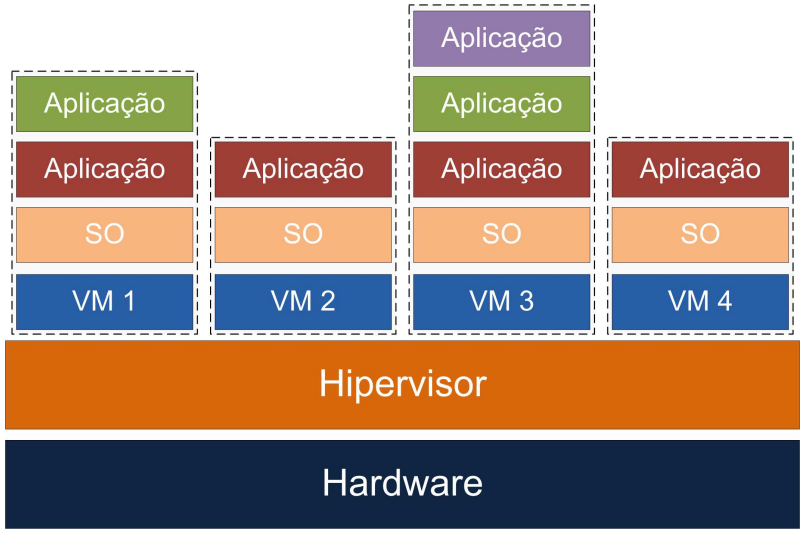


Figura 3.6: Diagrama de um ambiente virtualizado com hipervisor.

Ao usar o mesmo hipervisor em todos os servidores do seu data center, existe uma compatibilidade entre os *hardwares* virtuais criados, independente do *hardware* físico em que esse servidor se encontra, sendo assim possível a transferência dessas máquinas virtuais para outros servidores, gerando uma migração. Essa migração libera recursos computacionais na origem e passa a usar recursos do destino, ajudando a controlar e gerenciar o data center.

Migrações

A migração é uma funcionalidade que permite o ajuste do uso de recursos de um data center. Através dela é possível determinar o posicionamento de serviços e aplicações, conseguindo uma consolidação dos servidores em que as características do data center são mais bem aproveitadas.

A energia pode ser melhor distribuída ao alocar o mesmo volume de processamento a todos os servidores do data center, ou pode ser concentrada em um região que tenha melhor disponibilidade. A temperatura de uma região pode ser modificada com o posicionamento de máquinas virtuais, se a refrigeração estiver ruim em uma região a carga dos servidores atingidos pode ser redistribuída para diminuir a produção de calor na área afetada. Ao migrar todas as máquinas virtuais de um servidor específico a manutenção de áreas do data center pode ser planejada sem que seja necessário interromper algum serviço.

Além das funções descritas, a migração de máquinas virtuais pode ser utilizada para modificar aspectos dos dados trafegados na rede, escolhendo os locais onde o tráfego ocorre. O ato de migrar máquinas virtuais gera na rede uma sobrecarga momentânea, já que todo o conteúdo da memória deve ser transportado de um servidor para o outro. Essa sobrecarga pode ser julgada como benéfica se a migração conseguir aliviar alguma sobrecarga importante ao data center, seja ela qual for.

A ideia básica de migrar uma máquina virtual é copiar o estado atual da máquina para a outra localidade [46]. O estado atual de uma máquina é determinado pelos dados presentes na memória dessa máquina virtual. Assim, para realizar uma migração, é preciso pausar brevemente a máquina virtual, copiar a memória para a outra localidade e continuar a execução já na nova localidade. Os dados de disco são disponibilizados através de um servidor de imagens, que as compartilha por toda a rede, possibilitando que todas as máquinas virtuais as acessem.

Além dessa sobrecarga, ao mudar o posicionamento de uma máquina virtual que se comunica intensamente com outras, podemos afetar a rede de duas maneiras: positivamente, se o tráfego trocado entre as máquinas for melhor distribuído na topologia; e negativamente, se o tráfego passar por mais equipamentos para que a comunicação seja feita, podendo gerar congestionamento no data center.

Em data centers que, por algum motivo de escolha operacional, não trabalham com virtualização, precisamos definir o posicionamento dos serviços na topologia. Através da criação de uma imagem do sistema operacional do servidor é possível movimentar essa imagem para uma outra máquina hospedeira. Esse processo é usado pelo sistema de busca da Microsft, o Bing [12]. Dependendo do tipo de trabalho executado podemos escolher os locais de execução de uma aplicação ou podemos escolher mover um serviço executado em um local para outro. Nesses casos, é possível controlar o posicionamento das aplicações de forma semelhante ao que acontece na migração.

3.2.2 Serviços

Não é possível obter uma lista exaustiva de todos os serviços utilizados em centros de processamento de dados. Os data centers são controlados por inúmeras organizações distintas e usados para as mais diversas tarefas. Uma empresa que precisa de grande volume de processamento de dados pode buscar isto através do aluguel de espaço em um centro de processamento de dados, na nuvem, ou na construção do seu próprio centro de processamento de dados. Como essa necessidade é variável e intrínseca ao modelo de negócio da empresa, os serviços são desconhecidos e muitas vezes feitos sob encomenda especificamente para aquela função. No entanto, é possível classificá-los de acordo com alguns grandes grupos a partir de dados encontrados na literatura [5, 27, 28, 49, 50].

É possível classificar os serviços de acordo com a finalidade do processamento de dados. Três grandes áreas são comumente citadas como as que se beneficiam com o uso dos data centers, são elas: grandes processamentos de dados brutos, como sites de busca ou *Big data*; processamentos complexos, com muitos dados e muitas variáveis, usados principalmente na área científica; provimento de informações e processos de controle para clientes, como páginas de Internet, bancos de dados diversos e aplicativos voltados para o controle de processos específicos.

Big data

Existe uma grande tendência das empresas de acumular dados dos clientes. Com o barateamento do armazenamento de dados, tornou-se mais interessante guardar dados referentes às mais diversas transações e estatísticas do que filtrá-las e descartá-las. Esses dados vêm sendo processados cada vez mais para que características antes invisíveis às análises sejam conhecidas [12, 49, 51].

As ferramentas que processam grandes quantidades de dados precisam usar o paralelismo para conseguir lidar com toda a informação em um espaço curto de tempo. Para lidar com grandes massas de dados mantendo o custo baixo são necessárias ferramentas específicas. Uma das mais comuns para trabalhar com grandes volumes de dados é o Hadoop [51–53]. O Hadoop foi desenvolvido para ser utilizado com uma quantidade elevada de servidores de baixa confiabilidade, sendo assim o Hadoop é um software para computação distribuída de modo confiável e escalável.

Além dos mecanismos para garantir a redundância e a confiabilidade dos dados, existe um mecanismo para executar aplicações em paralelo, o MapReduce. O MapReduce é usado para processar grandes quantidades de dados, podendo ser usado para encontrar padrões, criar um ordenamento de itens, analisar *logs* de acesso *web*, construir indexação de páginas da *web*, aprendizado de máquinas, entre outros [51, 53, 54]. O MapReduce consiste basicamente em duas etapas. A primeira,

chamada de *map*, é responsável por pegar os dados brutos e retirar alguma informação desses dados. Essa etapa pode ser distribuída de acordo com a possibilidade da divisão dos dados de entrada. Se existem muitos dados e eles podem ser divididos em n partes, então podemos usar n máquinas para essa etapa. Na segunda fase, chamada de *reduce*, os dados classificados pela fase *map* são lidos por um número pré-determinado de máquinas que agrupam esses dados, enviando-os ao finalizar o processamento para uma máquina responsável por agrupar os dados, que terá o resultado final, como ilustrado na Figura 3.7.

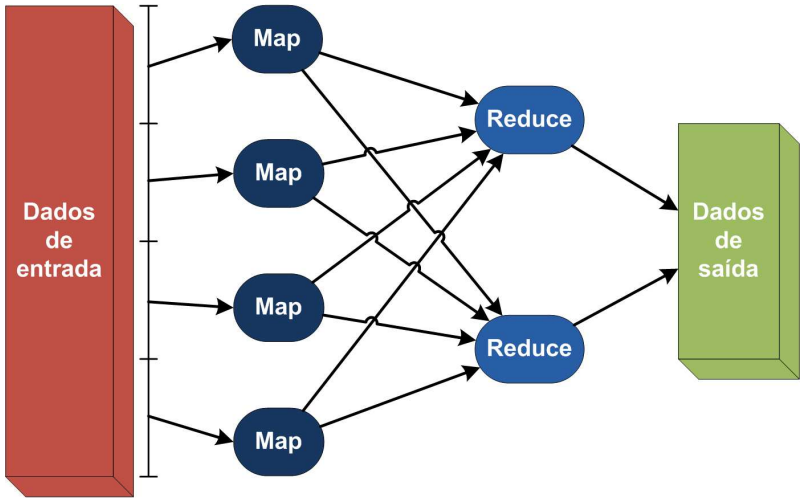


Figura 3.7: Fluxo de dados no modelo MapReduce.

Computação científica

A ciência sempre precisou recorrer a simulações para conseguir testar as suas teorias ou para conseguir coletar todos os dados de um experimento de forma rápida e barata. Quando a complexidade das simulações científicas aumenta, o tempo de execução também aumenta. Para melhorar o tempo de execução a programação distribuída pode ser utilizada.

Modelos de programação distribuída como o MPI (Message Passing Interface), o OpenMP (Open Multi-Processing), o CUDA (Compute Unified Device Architecture) e o PVM (Parallel Virtual Machine) permitem que o problema seja dividido em múltiplas partes, que durante o processo de convergência do resultado troquem informações e sinalizações, permitindo que um único resultado seja encontrado, sem que haja grandes perdas de eficiência [5, 50, 55].

O MPI é um conjunto de rotinas para trocas de mensagens em ambientes de memória distribuída. Ele é usado como método para que computadores interligados por meio de uma rede de dados executem um programa escrito para utilizar processamento paralelo [55].

O OpenMP é uma interface de programação que permite o desenvolvimento de aplicações com processamento paralelo através da especificação de pontos do código onde o processamento pode ocorrer dessa forma [55].

O CUDA é uma linguagem de programação que permite o uso dos processadores da placa de vídeo (GPU, formada por vários processadores em paralelo) para a execução de cálculos [55].

O PVM é um conjunto de rotinas para trocas de mensagens que funciona de maneira similar ao MPI [55].

Segundo Baker *et al.* [56], existem grandes partes da ciência que usam a computação distribuída, destacando-se as simulações de fenômenos que não podem ser medidos através de experimentos físicos, processamento de dados de experimentos científicos, as aplicações biomédicas (como a identificação de dinâmicas das proteínas, biocatálise, química quântica relativística de actinídeos), aplicações de projeto virtual de materiais e aplicações de modelagens climáticas globais são alguns exemplos do uso da computação científica. Ainda podem ser beneficiados renderizações de imagens e cálculos diversos, como a determinação do preço de geração de energia elétrica através de modelos computacionais [55].

Como exemplo de uso científico podemos citar os experimentos realizados pelo LHC (Large Hadron Collider ou grande colisor de hádrons), operado pela Organização Europeia para a Pesquisa Nuclear (CERN). Experimentos realizados por esse acelerador de partículas são usados para obter dados sobre colisões de feixes de partículas, tentando explicar a origem da massa das partículas elementares, encontrar outras dimensões do espaço, entre outros mistérios relacionados à matéria. Durante a execução dos testes, uma quantidade gigantesca de dados são gerados e precisam ser processados e armazenados. Os sensores do LHC geram em torno de 15 Gigabytes por segundo de informação, chegando a coletar 15 Petabytes de dados por ano. Durante o funcionamento o CERN já foram produzidos e armazenados mais de 100 Petabytes de dados, sendo que 75 Petabytes foram gerados somente pelo LHC. Para coletar e armazenar essa grande quantidade de dados são usados data centers com unidades de armazenamentos robóticas que guardam uma grande parte desses dados em fitas magnéticas. Além disso, esses dados precisam ser compartilhados para mais de 34 países que fazem pesquisas em parceria com o CERN [57, 58].

Distribuição de conteúdo

A distribuição de conteúdo é a tarefa mais visível para os usuários da Internet. Existem várias formas para se obter conteúdo, sendo elas divididas em duas grandes categorias, a arquitetura de cliente-servidor e a arquitetura distribuída.

Para entender o conteúdo distribuído através da arquitetura de cliente-servidor podemos usar a arquitetura atual dos sites da Internet. Todo o *site* precisa estar

hospedado em algum local, e esse local precisa ser capaz de atender às demandas específicas de cada *site*. Páginas estáticas, com pouca interatividade e que não recebem muitas visitas por dia não precisam de muitos recursos computacionais. As páginas com conteúdos dinâmicos, que precisam de bancos de dados, que devem executar algum processamento de acordo com os pedidos dos usuários ou que enviam uma grande quantidade de informações precisam de uma infraestrutura robusta para suportar todas as requisições dos usuários. Quanto maior o número de usuários simultâneos maior a necessidade de processamento e de banda para entregar o conteúdo desejado[5, 59, 60].

Linguagens como JavaScript, PHP, Python podem processar os pedidos dos clientes nos servidores. Quando um desses serviços é usado por um número muito grande de pessoas, é necessário um grande poder de processamento do lado dos servidores. Esses aplicativos devem ser implementados em um ambiente em que seja possível um alto volume de processamento de requisições, ou seja, os centros de processamento de dados. Como exemplos de aplicações que necessitam ser implementadas em data centers encontram-se as aplicações da *web* 2.0 com muitos clientes simultâneos [61], aplicações que disponibilizam dados, como vídeos, músicas ou imagens e aplicações colaborativas, onde mais de um usuário precisa ter acesso à informação e à possibilidade de alterar o seu conteúdo [60].

Uma nova maneira de compartilhar conteúdo baseado em uma arquitetura distribuída vem sendo desenvolvida por Jacobson *et al.* Através da Information-centric networking (ICN), ou rede baseada em informações, que consiste em um novo meio para compartilhar informações através da Internet. A proposta muda a forma de obter uma informação, ao invés do requerente da informação estabelecer uma conexão direta com um único servidor que contém essa informação, o requerente pede somente a informação que ele deseja, deixando o onde e como para o sistema de roteamento da ICN [62, 63].

A ICN identifica a informação através do nome do conteúdo e da coleção que esse conteúdo faz parte. Com essas informações todos os dispositivos que acessam uma mesma coleção tem os seus arquivos sincronizados em uma hora oportuna. Essas informações, junto com as informações de outras coleções, são armazenadas em um "grafo social", que contém todos os dispositivos e as suas respectivas coleções. [63]

Requisições dos usuários são feitas através de pacotes de interesse contendo um prefixo do nome e da coleção, identificando o conteúdo desejado. O roteador do conteúdo da ICN que é responsável por localizar e entregar a informação para o requerente, criando as ligações necessárias para a transmissão do dado e armazenando um cópia em cache da informação coletada.

O capítulo a seguir apresenta a definição do problema que é tratado pelo algoritmo de posicionamento APoS.

Capítulo 4

Posicionamento de Serviços em Data Centers

O entendimento do problema de posicionamento de serviços é essencial para produzir uma solução de posicionamento de máquinas virtuais que traga benefícios à rede do data center. Neste capítulo, na Seção 4.1, é exibido o problema de posicionamento de serviços e as suas peculiaridades baseados nos estudos exibidos no Capítulo 2 e no Capítulo 3. Depois de entender o problema, na Seção 4.2 são descritos os recursos utilizados na resolução dos problemas encontrados. As ferramentas utilizadas para ajudar a chegar ao resultado são apresentadas e os conceitos mais importantes relacionados à proposta são explicitados.

4.1 O problema de posicionamento de serviços

Como dito anteriormente, o problema consiste em encontrar a posição correta para os serviços executados em um data center. O posicionamento de serviços nesta dissertação consiste em identificar as máquinas virtuais que trocam uma maior quantidade de dados entre si do que com outras máquinas virtuais do mesmo data center. Ao agrupar essas máquinas conseguimos manter um tráfego intenso dentro desse grupo, sendo esse maior do que o tráfego trocado com as máquinas virtuais fora desse grupo. Ou seja, um *cluster* de máquinas virtuais é formado quando o tráfego dentro deste cluster é maior do que o tráfego que sai deste cluster. Após a identificação dos clusters, é necessária a alocação destes em servidores de forma que a localização desses clusters beneficie a topologia como um todo. Esse posicionamento objetiva redistribuir o tráfego que antes passava pelo núcleo da rede para as bordas da rede, diminuindo a ocorrência de enlaces congestionados no núcleo.

O problema de identificar os grupos de máquinas virtuais e alocá-las de forma a evitar o congestionamento dos enlaces do núcleo baseado somente na troca de dados

e em partições onde as máquinas são ligadas é NP-Completo [34, 36], não podendo ter a sua resposta encontrada em tempo polinomial. Desse forma, ao tentar melhorar o resultado incluindo mais variáveis de entrada, como energia ou tolerância a falhas, o problema se torna mais complexo.

Tendo em vista a dificuldade de determinar o posicionamento ótimo de serviços, esse trabalho se concentra em procurar o posicionamento de serviços em um data center levando em consideração quatro variáveis: o processamento que uma determinada máquina virtual precisa para executar um serviço, a quantidade de memória exigida pelo serviço e pelo sistema operacional hospedeiro desse serviço, a topologia do data center, e a matriz de tráfego do data center.

Usando essas quatro variáveis o algoritmo de posicionamento APoS determina uma arrumação para as máquinas virtuais em que a troca de informações entre essas máquinas seja concentrada a uma área predeterminada do data center, onde a banda de comunicação disponível é beneficiada por passar por menos comutadores, diminuindo a utilização dos enlaces da camada de núcleo e agregação do data center.

O algoritmo funciona agrupando máquinas virtuais que trocam uma elevada quantidade de informações em uma janela de tempo. Após o agrupamento, essas máquinas são designadas a servidores físicos que têm capacidade de comportar as suas demandas de processamento e de memória, de uma forma em que o tráfego trocado entre essas máquinas virtuais seja concentrado em uma região predefinida do data center. Essa análise é feita pela matriz de tráfego do data center, que é o quanto cada máquina virtual e, consequentemente, cada serviço precisa se comunicar com algum serviço relacionado para funcionar corretamente.

Para conseguir concentrar o tráfego do data center, a última variável é a topologia do data center. É preciso localizar e mapear as zonas do data center que possuem um elevado grau de comunicação. Foi desenvolvida uma solução, apresentada na Seção 4.2.3, para encontrar essas áreas com alta taxa de comunicação nas topologias mais discutidas na literatura.

4.2 Recursos usados para desenvolver a solução

A solução proposta interpreta os dados da matriz de tráfego do data center, de consumo de CPU e memória e a topologia do data center. Para procurar os padrões que consigam relacionar máquinas virtuais com demandas de tráfego comuns e alocá-las aos locais específicos, o algoritmo precisa do auxílio de algumas ferramentas. Essas ferramentas serão descritas na Seção 4.2.1.

4.2.1 Comunidades

Uma comunidade é um grupo de nós que são intensamente relacionados, mais relacionados entre si do que com os seus vizinhos externos. Uma definição formal, segundo Porter *et al.* [64]: "uma comunidade consiste em um grupo de nós que são relativamente densamente conectados entre si mas esparsamente conectados a outros grupos na rede". A Figura 4.1 representa um grafo com as comunidades existentes realçadas.

Vários algoritmos são propostos na literatura [64–69], cada um aplicado a uma área distinta.

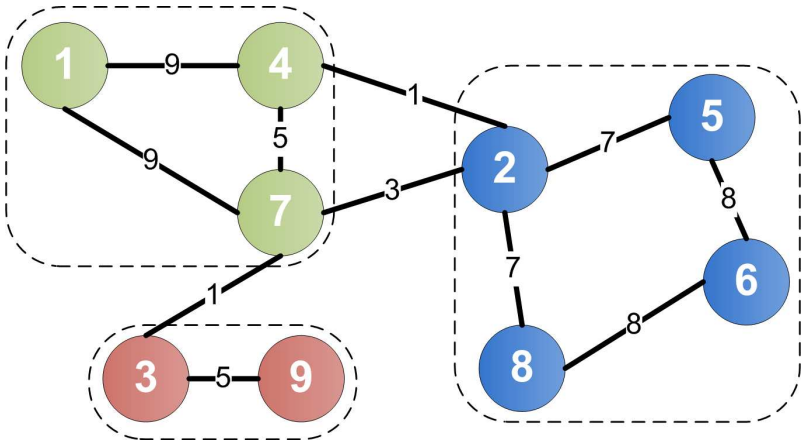


Figura 4.1: Exemplo de grafo com comunidades identificadas.

Porter *et al.* [64] fizeram um levantamento identificando os principais algoritmos existentes para a detecção de comunidades. Algumas áreas se destacam, como os métodos tradicionais baseados em *k-means*, algoritmo de Kernigham-Lin, baseados em centralidade (Girvan-Newman), *k-clique percolation*, otimização modular e particionamento espectral. A seguir apresenta-se as principais características desses métodos.

K-means é uma das técnicas mais simples para resolver o problema de detecção de *clusters*. Consiste em inicialmente escolher o número K de centróides. Esses centróides definem a que *cluster* os nós mais próximos pertencem. A partir dos nós de um *cluster* previamente definido são calculados novas posições para os centróides e uma nova iteração é feita para definir a que centróide cada nó pertence. Através de passos sucessivos os centróides são movidos até que a melhor divisão possível seja encontrada. Essa divisão é encontrada quando os centróides não se movem mais em iterações sucessivas [70].

A proposta Kernigham-Lin baseia-se em maximizar uma função Q que está relacionada ao número de arestas dentro de cada grupo de nós com o número de nós de outros grupos. O algoritmo funciona recursivamente, a cada chamada aumenta

o número de comunidades até que o número final de comunidades desejado seja encontrado. O resultado deste método depende fortemente da escolha das primeiras partições [64].

O método de centralidade de vértices, proposto por Michelle Girvan e Mark Newman, usa o conceito de centralidade de arestas. Uma aresta tem um alto valor de centralidade se passam por ela muitos menores caminhos entre os nós que fazem parte do grafo, com isso, as arestas que apresentam valores altos de centralidade são usadas para a comunicação entre comunidades. Ao remover essas arestas do grafo a tendência é que os nós que restaram formem comunidades desconexas. A cada remoção de arestas o valor da centralidade de cada aresta deve ser recalculado, porque ao remover uma aresta outra aresta poderá ter o valor de centralidade aumentado. Esse é um algoritmo de divisão para detectar comunidades, desconstruindo o grafo até que pedaços menores isolados sejam formados [67].

Alguns algoritmos usam a ideia de k-clique, que consiste em um conjunto de vértices em um grafo que tem todas as ligações feitas, ou seja, um subgrafo completo. Como uma comunidade tem características de conter várias pequenas k-cliques, o algoritmo funciona localizando e agrupando as k-cliques próximas [67].

A modularização é uma medida de divisão de redes em módulos. Redes com alta modularidade possuem conexões densas entre os nós que fazem parte do módulo e dispersas com outros módulos. Essa propriedade pode ser utilizada para encontrar comunidades ao analisar todos os conjuntos de nós procurando altos valores de modularidade. Como essa busca é muito custosa de ser feita para todos os nós, heurísticas foram criadas utilizando esse método e outras ferramentas para conseguir analisar grandes redes [65, 66].

O particionamento espectral, um tipo de modularização, consiste em analisar a matriz de adjacências da rede, retirando propriedades do seu laplaciano. O método mais simples inicialmente divide a rede em duas componentes e aplica a matriz laplaciana, procurando o segundo elemento com maior valor. Esse elemento ao ser retirado divide a rede. Ao repetir esse processo várias comunidades podem ser encontradas. Como vantagem, esse método pode ser executado de forma paralela. Como desvantagem se as partições iniciais não forem escolhidas em tamanho e número corretos a solução pode ser trivial, formando comunidades com um único elemento [68].

Esses algoritmos foram estudados para tentar achar um que melhor se adequasse ao problema. O algoritmo deveria ser rápido o suficiente para ser executado em poucos minutos, capaz de achar comunidades em grafos de até 100 mil nós e facilmente aplicável a cálculos sobre a matriz de tráfego. O número de comunidades não deve ser conhecido no início do algoritmo, permitindo ambientes com grandes variedades de tamanho de comunidades. Outra limitação é que as comunidades não

deveriam ter tamanho definido, podendo conter somente um elemento a até todos os elementos do conjunto.

Ao aplicar essas restrições, descartamos algoritmos como o *K-means* pois precisa de um tamanho inicial de comunidades, o Kernigham-Lin por precisar de um ponto de parada para o número de comunidades e por estar fortemente atrelado a escolha inicial das partições. O método de particionamento espectral é prejudicado por não se ter o conhecimento do número e dos tamanhos das partições, então não deve ser usado.

Algoritmos de modularização procuram minimizar funções de custo. Esse tipo de algoritmo é complexo e não escala para múltiplos nós facilmente. O algoritmo que melhor se adequou ao nosso caso foi o algoritmo por centralidade de vértices, proposto por Girvan e Newman.

O algoritmo foi escolhido principalmente pela forma como o algoritmo é executado, podendo ser facilmente implementado em redes que representam os padrões de comunicação em data centers. Como desvantagem existe o tempo computacional elevado para a execução do algoritmo, principalmente na fase em que todos os menores caminhos devem ser calculados. Para otimizar o algoritmo permitindo a sua execução em tempos menores, foi feita uma modificação que será descrita em detalhes na Seção 5.3.1.

4.2.2 Grafos

A matriz de tráfego pode ser usada para modelar a rede como um grafo. Cada um dos vértices representa um endereço, sendo este ocupado por uma máquina virtual com um ou mais serviços sendo executados. Cada uma das arestas representa uma transferência de dados entre duas máquinas virtuais, representado o total de dados trafegados nas duas direções de um enlace. É importante ressaltar que nesse grafo nenhuma parte da topologia é representada.

O grafo construído é completo, ou seja, cada um dos vértices é conectado a todos os outros, existindo todas as arestas possíveis. Em um data center, todos os computadores têm os recursos necessários para se comunicar com todos os outros, formando um grafo completo. A quantidade de dados trocados entre as máquinas virtuais é representada por um peso em cada aresta. Se um endereço não enviou pacotes para outro endereço o peso da aresta é zero, e quanto maior a troca de dados, tanto envio quanto recebimento, maior o peso de uma aresta.

Uma matriz de tráfego pode conter muitos valores baixos em relação aos outros ou valores nulos, tornando a análise mais fácil ao descartar dados abaixo de um limite. O cálculo desse limite será melhor analisado no Capítulo 5. Para ilustrar a montagem desse grafo, mostramos a matriz de entrada na Figura 4.2 e a geração do

grafo correspondente na Figura 4.3. Elementos com valor nulo na matriz de tráfego são transformados em arestas com peso nulo, que não são representadas no grafo.

$$A = \begin{bmatrix} 0 & 9 & 9 & 0 & 0 & 0 & 0 & 0 \\ 9 & 0 & 5 & 1 & 0 & 0 & 2 & 0 \\ 9 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 7 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 8 & 10 & 0 \\ 0 & 0 & 0 & 7 & 8 & 0 & 3 & 0 \\ 0 & 2 & 0 & 5 & 10 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figura 4.2: Matriz de tráfego fictícia.

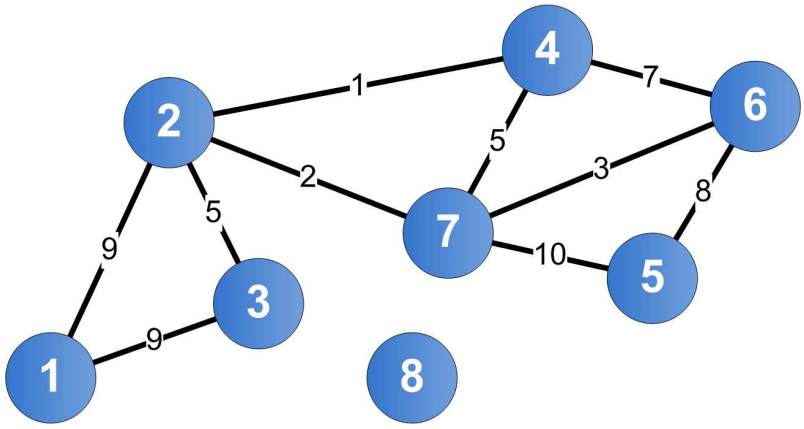


Figura 4.3: Grafo gerado a partir da matriz de tráfego.

Usando ferramentas aplicadas ao estudo de grafos é possível analisar diversos aspectos da interação entre as máquinas virtuais tais como: menores caminhos, centralidade, caminhos distintos, caminhos únicos, entre outros. Uma medida importante que os grafos permitem realizar é a detecção de comunidades usando o método de k-clique ou o Girvan-Newman [64, 70].

4.2.3 Particionamento da topologia

O particionamento da topologia é essencial para a eficiência do algoritmo de posicionamento. Particionar a topologia significa escolher as regiões do data center em que a comunicação será feita de maneira a reduzir o uso de outras áreas do data

center. Significa escolher os servidores que devem ser utilizados em conjunto para que os dados que trafegam entre as máquinas virtuais posicionadas nesses servidores consigam usufruir de uma alta banda para troca de dados.

Ao dividir todo o data center em regiões e alocar as comunidades de máquinas virtuais a essas divisões, o objetivo é diminuir a comunicação entre essas regiões. Em uma divisão correta do data center, os meios de comunicação entre as partições serão feitos somente pelas camadas mais altas da rede, a camada do núcleo e da agregação em topologias baseadas em árvores por exemplo. A divisão objetiva minimizar o uso de enlaces em que existe um gargalo de capacidade de transmissão de dados.

Ao colocar dentro de uma mesma partição uma comunidade que trafega muitos dados internamente, conseguimos que grande parte do tráfego permaneça nessa comunidade, não deixando a partição. Como consequência menos dados precisam trafegar nas camadas mais altas do data center.

Cada topologia apresenta a sua peculiaridade na hora da divisão em partições. As mais fáceis de dividir, as topologias baseadas em árvore, têm divisões naturais, baseadas na sua divisão por níveis. Em uma árvore, as ramificações de cada nível de cada elemento podem ser consideradas partições, independente de quantos elementos existem por nível. A Figura 4.4 mostra uma árvore com particionamento a partir do seu segundo nível, a camada 2.

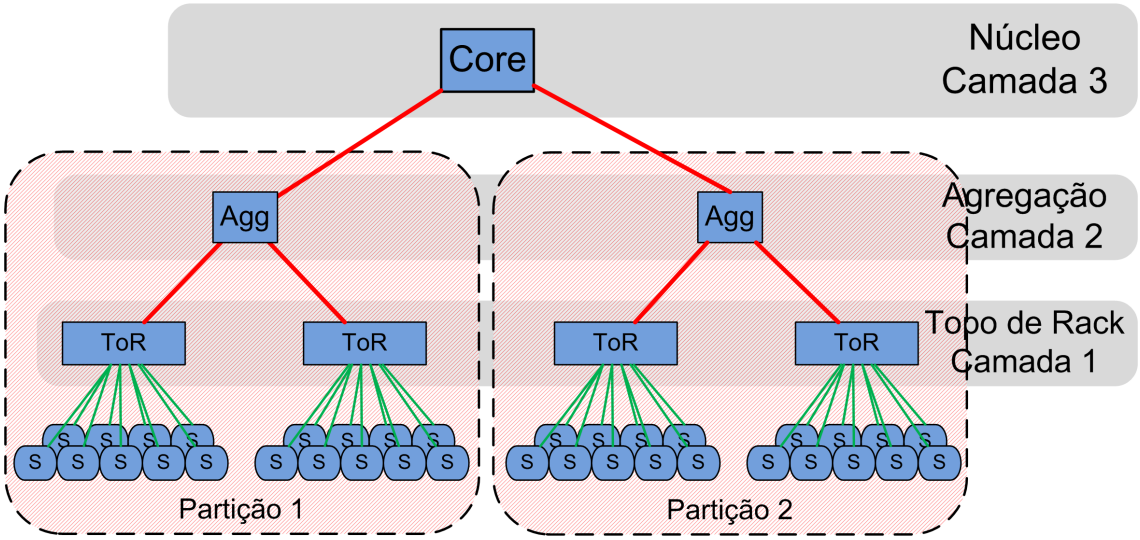


Figura 4.4: Exemplo de divisão de uma topologia em árvore.

Em topologias formadas por recursão, como a DCell e a BCube, a divisão será diferente. Devido à regra de formação dessas topologias, criadas por níveis, a divisão pode ocorrer na mudança de nível. Por exemplo, uma célula de nível mais baixo, DCell_0, é uma boa divisão para o algoritmo, mas essa divisão pode ser feita mais acima na recursão, entre os DCell_1. O BCube segue o mesmo princípio.

A Figura 4.5 mostra um exemplo de divisão da topologia BCube com 2 níveis. Como dito, a topologia foi particionada no nível mais baixo, agrupando os servidores que estão ligados a um mesmo comutador.

Como o APoS não diferencia os enlaces utilizados para a comunicação ou equipamentos é possível simplificar as ligações entre as partições ao agrupar os links que interligam as partições. Os algoritmos de roteamento utilizados pelos DCell e pelo BCube propõem que o tráfego seja distribuído entre todos os equipamentos, usando toda a topologia para entregar os dados, sem privilegiar algum enlace. Com isso, ao estabelecer uma ligação virtual entre as partições formada pelos enlaces que ligam os servidores aos comutadores do nível superior, é possível simplificar a determinação de partições pelo APoS, transformando o BCube e também o DCell em uma árvore de um nível.

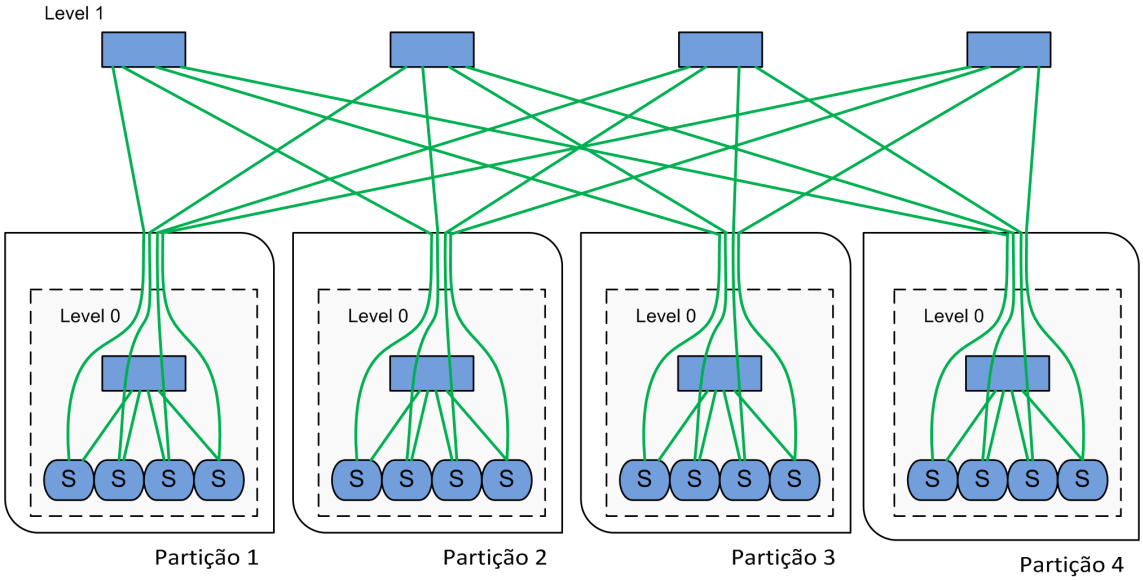


Figura 4.5: Exemplo de divisão de uma topologia BCube.

Outra abordagem para a divisão em partições da topologia, olhando mais para o modo de operação de um data center, é procurar quais são as máquinas ligadas diretamente a um mesmo comutador. Essas máquinas têm disponível a banda máxima possível para trocar dados, não ocasionando gargalos em outras máquinas ao transferir uma grande quantidade de dados para outras máquinas virtuais. Em topologias baseadas em árvore seria como escolher todos os servidores que se encontram em um mesmo *rack* e ligados ao mesmo comutador de topo de *rack*.

Para escolher de forma mais precisa a localização das máquinas virtuais é possível fazer divisões sucessivas da topologia: uma primeira execução do algoritmo com uma divisão das ramificações da parte mais central da topologia, depois uma segunda execução em paralelo dividindo novamente as porções da topologia e alocando so-

mente as máquinas virtuais que foram selecionadas na primeira divisão. Esses passos podem ser seguidos até o nível de um *rack*, onde os servidores que melhor alocam aquelas máquinas virtuais serão selecionados. Cada execução em uma subdivisão otimiza o uso dos enlaces em que houve essa divisão.

A Figura 4.6 exemplifica a subdivisão de uma topologia de árvore para a execução do algoritmo em mais de uma rodada, ressaltando que a partir da segunda divisão é possível a execução em paralelo do algoritmo, reduzindo o tempo de convergência dos dados. Nesse caso os dados de máquinas virtuais e topologias em cada divisão podem ser tratados independentemente pois a escolha de local dentro de uma subdivisão não irá influenciar no tráfego entre as divisões iniciais.

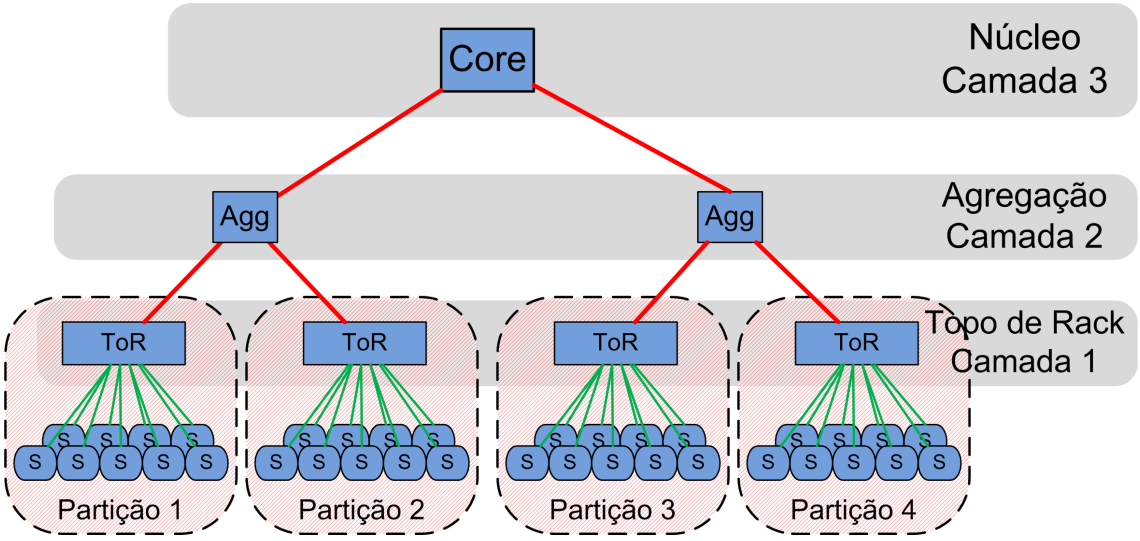


Figura 4.6: Subdivisão da topologia.

4.2.4 Empacotamento

Para alocar corretamente as comunidades encontradas pelo algoritmo nas partições definidas precisa-se de um método de empacotamento. O data center define um cenário onde as partições não necessariamente possuem o mesmo tamanho e cada comunidade encontrada têm parâmetros distintos de uso de CPU, memória e comunicação de rede. Empacotar itens com tamanhos variados em recipientes de tamanhos distintos, é um problema conhecido como *bin packing*. O problema de *bin packing* é complexo do tipo NP-completo [71–74], sendo difícil encontrar a sua solução ótima.

Para resolver o problema de empacotamento deve-se usar heurísticas para a alocação das comunidades de máquinas virtuais às partições de servidores. Alguns algoritmos baseados em heurísticas são usados e estudados na literatura, sendo definidos como os mais velozes e os mais capazes de produzir uma solução próxima da

ótima em todos os casos. Algoritmos como o *first-fit*, *best-fit* e *worst-fit* são os mais usados em problemas desse tipo [72].

Esses algoritmos recursivos se baseiam na ideia de criar regras para a determinação da distribuição do posicionamento dos itens nos seus recipientes. Os métodos necessitam que o algoritmo seja executado em tempo real, ou seja, cada comunidade encontrada deve ser imediatamente alocada em um conjunto de servidores, por isso foram usados somente algoritmos *on-line* de alocação. Cada algoritmo funciona da seguinte maneira:

- O *first-fit* analisa os itens na ordem em que estão disponíveis, tentando alocá-los no primeiro recipiente disponível. Se não for encontrado um recipiente que o acomode, um novo recipiente será utilizado.

- O *best-fit* analisa os itens na ordem em que estão disponíveis, tentado alocá-los no recipiente mais cheio que ainda possui espaço para o item alocado. Se não for encontrado um recipiente que o acomode, um novo recipiente será utilizado.

- O *worst-fit* analisa os itens na ordem em que estão disponíveis, tentado alocá-los no recipiente mais vazio. Se não for encontrado um recipiente que o acomode, um novo recipiente será utilizado.

Para ilustrar as diferenças entre as heurísticas de *bin packing*, a Figura 4.7 mostra o estado inicial de uma distribuição de itens e o estado final de após a aplicação dos três algoritmos de empacotamento, sendo eles respectivamente o *first-fit*, *best-fit* e *worst-fit*.

A diferença entre os três algoritmos é somente na hora da escolha de onde alocar uma comunidade nas partições de servidores. O *first-fit* aloca na primeira disponível, o *best-fit* na mais ocupada e o *worst-fit* na menos ocupada, com isso temos características distintas no resultado final das alocações. Em uma análise rápida podemos dizer que o *first-fit* e o *best-fit* tendem a alocar as comunidades em algumas partições, nas primeiras no *first-fit* e nas mais cheias no *best-fit*. O *worst-fit* tende a distribuir melhor as comunidades entre todas as partições.

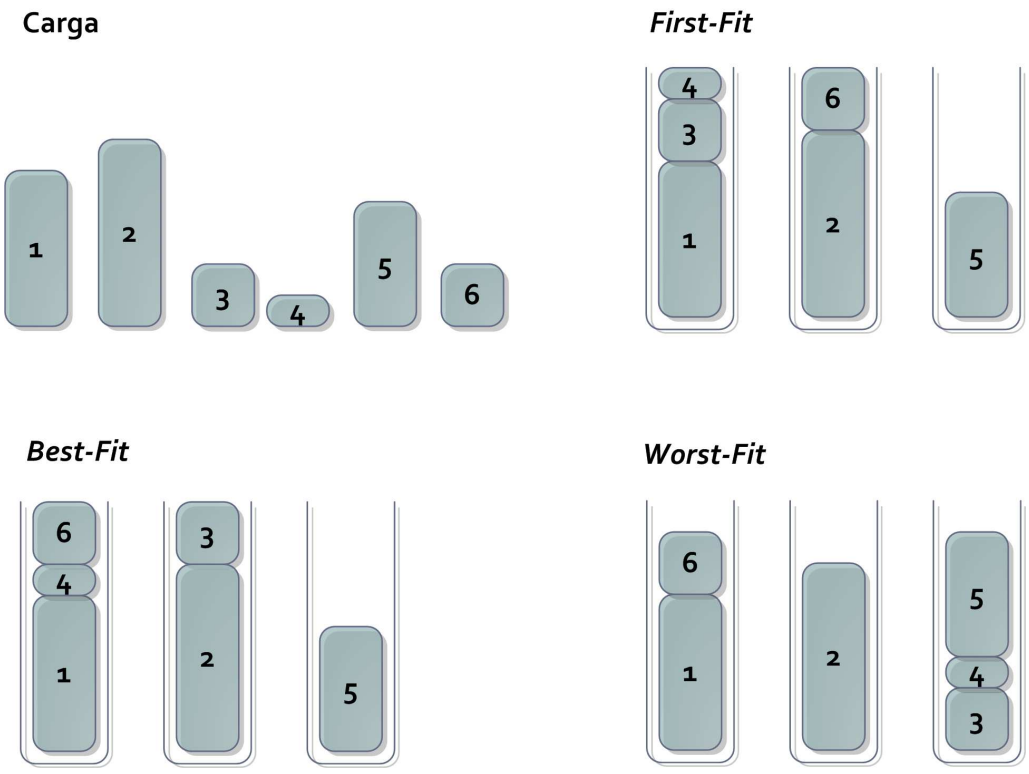


Figura 4.7: Algoritmos de *bin packing*.

Neste capítulo foi introduzido o problema de posicionamento de serviços em data center e apresentadas as ferramentas utilizadas para auxiliar na resolução do problema. No próximo capítulo será apresentado o algoritmo desenvolvido para tratar o problema de posicionamento de máquinas virtuais em data centers, o APoS, com detalhes de sua implementação.

Capítulo 5

Algoritmo de Posicionamento de Serviços

Neste capítulo é apresentado o Algoritmo de Posicionamento de Serviços proposto, o APoS. O APoS é um algoritmo de posicionamento de máquinas virtuais em centros de processamento de dados baseado na análise do tráfego trocado entre os serviços, posicionando-os de forma a diminuir o uso do núcleo da rede. Para isso, o APoS usa a estratégia de analisar as máquinas virtuais que precisam trocar dados frequentemente, as agrupa e as posiciona no data center de forma que o fluxo de informações entre essas máquinas interfira menos intensamente nos fluxos de outras máquinas virtuais.

O algoritmo tenta agregar máquinas virtuais que trocam muitos dados frequentemente e as posiciona em uma região concentrada do data center, onde o tráfego não precisará passar por muitos enlaces e comutadores, diminuindo o tráfego total que precisa circular pelo núcleo da rede.

O algoritmo é dividido em quatro etapas: a aquisição de dados, particionamento dos serviços, clusterização de máquinas virtuais e saída de dados.

Um fluxograma da execução do algoritmo de posicionamento de serviços é exibido na Figura 5.1. Este fluxograma objetiva um melhor entendimento dos passos do algoritmo, dessa forma, é dividido em etapas de acordo com a descrição do algoritmo neste capítulo. Para iniciar o processo, são necessárias quatro informações: a topologia do data center, os dados de CPU e memória dos servidores, a matriz de tráfego das máquinas virtuais e os dados de CPU e memória das máquinas virtuais. Com esses dados, são definidas na segunda parte do algoritmo as partições de servidores. A terceira parte do algoritmo forma as comunidades de máquinas virtuais, alocado-as aos servidores contidos nas partições. A quarta parte finaliza o algoritmo, produzindo a saída de dados.

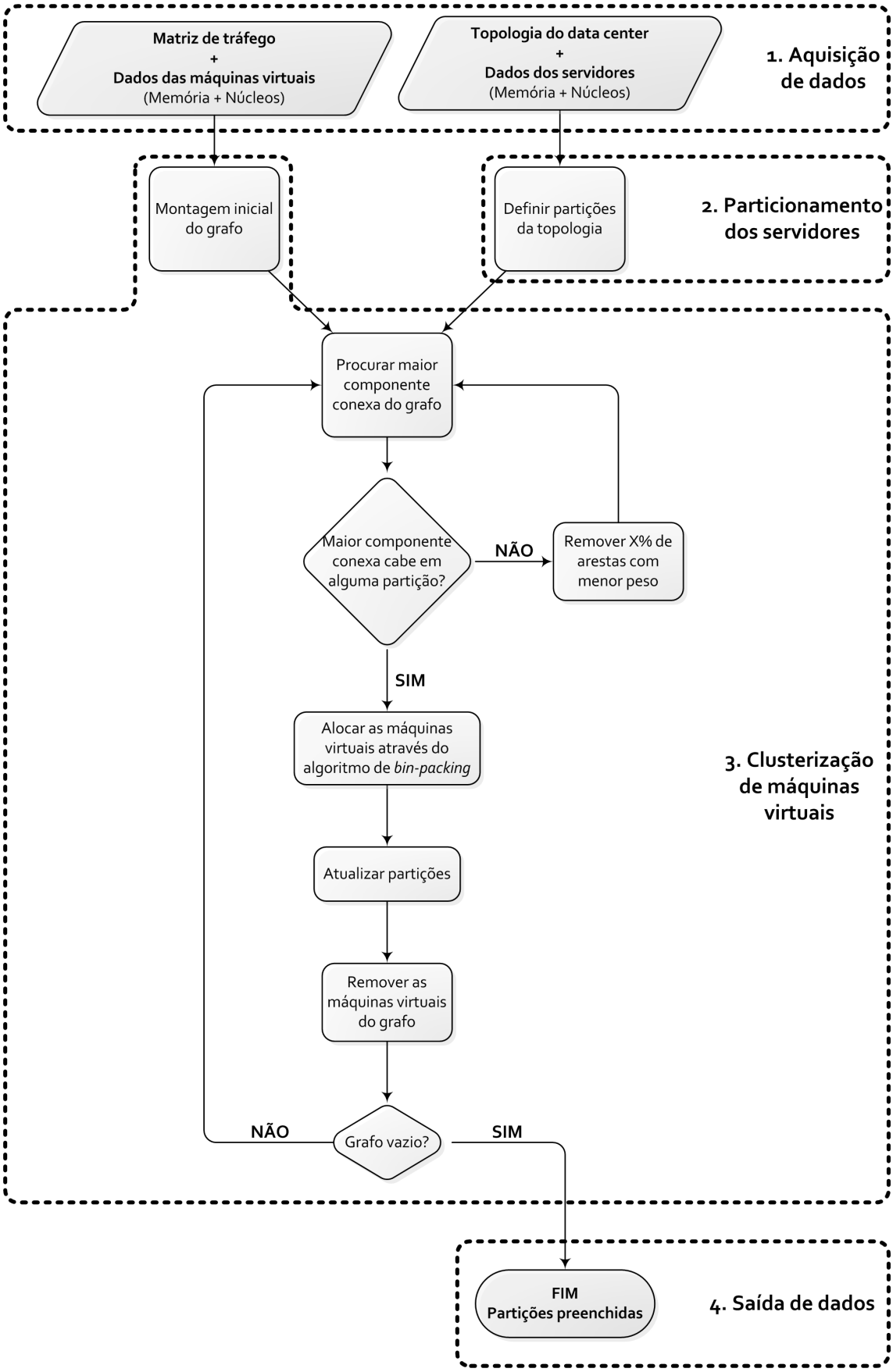


Figura 5.1: Fluxograma de execução do APOS.

5.1 Aquisição de dados

A aquisição de dados é a etapa responsável por obter as medidas de tráfego que fluem entre as máquinas do centro de processamento de dados e obter dados da topologia em que os servidores são ligados. O objetivo da aquisição de dados é montar uma matriz de tráfego que seja capaz de mostrar qual é a situação da transferência de dados entre os computadores do data center e montar uma representação da topologia em que seja possível analisar as regiões que possuem diferenças na banda de transferência de dados.

Montar a matriz de tráfego significa contabilizar quantos *bits* de dados cada endereço IP enviou para outro endereço IP em um durante um determinado período. Como um data center pode chegar a ter dezenas de milhares de servidores interligados e cada um deles com algumas máquinas virtuais com IPs distintos, temos um número elevado de pontos a serem sondados. Adquirir corretamente esses dados e enviá-los a um computador para concentrar e posteriormente analisar esses dados é essencial mas representa um desafio. Assim, o período de aquisição de dados deve ser cuidadosamente escolhido.

Primeiro é necessário definir uma janela de tempo de aquisição para que haja tempo suficiente para se mensurar os *bits* trafegados na rede por uma máquina virtual e enviá-los a um único computador da rede. O tempo dessa janela varia conforme as peculiaridades do ambiente e de acordo com a variabilidade do tráfego no data center.

Não é possível definir um tempo padrão ótimo para todos os data centers, portanto são definidas duas restrições que devem ser seguidas para se obter os dados necessários para a geração da matriz de tráfego. A primeira restrição é que todos os dados de um intervalo de tempo devem estar disponíveis concomitantemente no concentrador de dados, a janela deve ser grande o suficiente para que sejam lidos os dados enviados por todos os computadores. A segunda restrição é que os dados devem estar atualizados a cada rodada do algoritmo. Se é necessário executar o algoritmo a cada n segundos, é necessário que todos os dados sejam coletados em n segundos ou menos.

A matriz de tráfego será então preenchida com os valores de quanto um endereço se comunicou com o outro em um intervalo de tempo. Esse valor é constituído do somatório dos dados que foram enviados e dos dados que foram recebidos entre esses dois endreços. A unidade de medida não é importante desde que os dados sejam consistentes. Nas simulações executadas no Capítulo 6 são usados dados normalizados, não tendo ligação direta com a representação de *bit*, ou seja, o valor somente mostra o comportamento dos dados e não os valores reais de *bit* medidos em um data center.

A topologia deve ser identificada através da disposição dos comutadores e servidores da rede, da capacidade de cada enlace que interliga esses componentes e também da quantidade de memória e de processamento que cada um dos servidores do data center. A topologia é necessária para determinar quais são as porções do data center em que existe uma elevada banda disponível para a conexão entre os servidores e quais os recursos disponíveis nessas porções. Na Seção 5.2 é definido como esses dados são usados para formar uma partição de servidores.

Diferentes métodos podem ser usados para identificar essa topologia. Como um centro de processamento de dados não é modificado com novos enlaces ou com a retirada de equipamentos frequentemente, a topologia pode ser indicada manualmente para o algoritmo. É possível também inferir a topologia através de ferramentas específicas para essa finalidade [75, 76].

O trabalho de Pandey *et al.* [75] usa o protocolo SNMP (Simple Network Management Protocol) para descobrir a topologia de redes, desde comutadores e roteadores até os servidores que pertencem a uma rede, mesmo que essa esteja subdividida em outras redes.

Os fabricantes de equipamentos de rede também tem ferramentas para descobrir a topologia de rede, mas são voltados para os equipamentos de nível 3 (camada de rede), ou seja, roteadores [75]. A Cisco possui uma ferramenta chamada *Cisco Network Connectivity Monitor* [77] que através do protocolo SNMP consegue configurar e monitorar redes baseadas nos seus equipamentos. O software *Network Topology Mapper* da empresa Solawinds é capaz de detectar a topologia de rede através de múltiplos métodos, como o SNMP, ICMP (Internet Control Message Protocol), WMI (Windows Management Instrumentation), CDP (Cisco Discovery Protocol), VMware entre outros.

De qualquer maneira é necessário monitorar a topologia, os enlaces e os comutadores que fazem parte da topologia para que se tenha certeza que ligações não foram perdidas, atrapalhando no funcionamento do data center.

As capacidades de processamento e memória e, consequentemente, o número de máquinas virtuais que podem ser instaladas em cada servidor seguem o mesmo princípio. Como não são mutáveis podem ser manualmente aferidas ou podem ser aferidas através de sondas instaladas em cada um dos servidores, de forma parecida como é feito no trabalho Wood, o Sandpiper [32]. Nele são instalados pequenos programas no domínio 0 de cada um dos servidores de máquinas virtuais sendo eles capazes de aferir com precisão tanto a quantidade de memória e CPU instaladas como a quantidade usada por cada uma das máquinas virtuais e o disponível para ser alocado a novas máquinas virtuais.

5.2 Particionamento dos servidores

Como mostrado na Seção 3.1 existem diversas topologias, cada uma com as suas características distintas. Todas as topologias estudadas apresentam divisões bem definidas, como camadas na árvore e na *fat-tree* e níveis de células distintos na BCube e na DCell. Dessa forma podemos definir uma partição de várias formas em cada topologia e entre as diversas topologias, existindo uma grande possibilidade de escolha de partições. Isso se deve à grande diversidade de topologias disponíveis que podem ser implementadas em um data center.

O particionamento de servidores objetiva encontrar áreas no centro de processamento de dados em que os dados que são trocados internamente em uma partição utilizem mais banda do que os dados que são trocados entre partições. Ao fazer essa divisão e priorizar que máquinas virtuais com grandes transferências de dados fiquem em uma mesma partição, temos menos fluxos entre as partições e, consequentemente, os enlaces mais importantes e com menos capacidade de transmissão têm a quantidade de tráfego reduzida.

Para particionar uma topologia qualquer é possível utilizar os equipamentos que fazem a interface entre os diversos níveis. Nas topologias apresentadas é comum que o equipamento que se comunica com a camada acima ter uma alta taxa de transmissão no seu nível mas transmitir dados para a camada acima a somente uma fração dessa taxa de transmissão. Em uma árvore, nos comutadores com diferença de banda de transmissão entre os níveis, temos uma oportunidade para a formação de partições. Em uma topologia é possível formar partições em qualquer nível, desde os comutadores de núcleo até os comutadores de topo de *rack* ou mesmo os próprios servidores, compartilhando uma mesma interface de rede mais com um poder maior de encaminhamento quando as máquinas virtuais estão no mesmo servidor.

A partir da existência de uma multiplicidade de divisões, é possível dividir o problema de acordo com o nível da camada e usar a recursão para melhorar o posicionamento das máquinas virtuais. A cada divisão consecutiva da topologia escolhe-se o posicionamento das máquinas virtuais referentes àquela partição, ou seja, otimizando somente os enlaces que são usados para interligar as partições. Para o melhor posicionamento possível de todas as máquinas virtuais de um data center, conseguindo o menor uso possível para todos os enlaces que fazem parte do data center é necessário que o algoritmo seja dividido e executado recursivamente, modificando as partições e as máquinas virtuais em que o algoritmo é executado.

Para as subdivisões, em um primeiro passo o algoritmo analisa todas as máquinas virtuais do data center e escolhe formar grandes e pouco numerosas partições de servidores, a partir das conexões no seu nível mais alto. Após definir a posição das máquinas virtuais, o algoritmo é recursivamente chamado. Nessa nova rodada do

algoritmo, cada uma das divisões anteriores executará uma instância do algoritmo, com somente as máquinas virtuais designadas àquela partição. Nessa nova rodada acontecerá também uma nova divisão da partição, no segundo nível da topologia. O processo de recursão pode ser feito até que se chegue ao nível do servidor. A subdivisão do problema visa a otimização do uso dos enlaces de cada nível.

A escolha exata das etapas e dos enlaces dos particionamentos pode ser feita de duas maneira distintas: é possível escolher manualmente as partições e os servidores que a compõe ou é possível a determinação através da análise da topologia.

No segundo caso, na análise automática da topologia, algumas abordagens podem ser usadas. Qualquer nível bem definido de comutadores é um bom ponto para a divisão da topologia. Em uma árvore, *fat-tree* ou mesmo D-Cell e Bcube usar os servidores ligados aos comutadores de cada nível é uma boa divisão da topologia, onde cada comutador será uma partição.

Após definir o particionamento dos servidores é possível iniciar o processo de clusterização das máquinas virtuais.

5.3 Clusterização de máquinas virtuais

A clusterização de máquinas virtuais objetiva identificar as máquinas virtuais que trocam grandes volumes de informações e agrupá-las de forma a conseguir conter esses grandes fluxos em uma partição do data center. A identificação é feita através da análise da matriz de tráfego. A matriz de tráfego é transformada em um grafo, como descrito na Seção 4.2.2, e esse grafo é analisado à procura de comunidades, descritas na Seção 4.2.3. Existem vários diferentes métodos e abordagens para encontrar as comunidades, descritos por alto na Seção 4.2.1. Agora é apresentado o método usado no algoritmo de posicionamento, criado por Girvan-Newman, explicitando as alterações feitas para otimizar o seu uso no problema.

5.3.1 Algoritmo de Girvan-Newman

O método de Girvan-Newman baseia-se em determinar a centralidade de cada uma das arestas e remover as que possuem um valor elevado de centralidade, com o objetivo de desconectar o grafo para que seja possível determinar os subgrafos conexos dentro do grafo original, formando cada subgrafo uma comunidade.

A centralidade de uma aresta (*edge betweenness*) mede o número de caminhos mais curtos que passam por aquela aresta. É preciso calcular todos os caminhos mais curtos entre todos os vértices do grafo, contabilizando por quais arestas esses caminhos passam. O processo de cálculo de menores caminhos não é trivial, sendo um problema de elevado custo computacional, e a necessidade de fazer o processo

para todos os pares distintos de vértices do grafo eleva a complexidade do algoritmo.

Se em uma aresta passam muitos caminhos, é possível afirmar que ela é usada como comunicação entre comunidades, já que em uma comunidade os vértices estão próximos, normalmente diretamente ligados e são necessárias poucas arestas para a comunicação. O algoritmo então tenta isolar as comunidades cortando as ligações entre comunidades. Uma aresta que apresenta o maior valor de centralidade é removida do grafo. Se ao retirar essa aresta o grafo foi desconectado a divisão em duas partes forma duas comunidades distintas, onde cada subgrafo determina uma comunidade.

O processo de remoção de arestas precisa ser repetido até que o número desejado de comunidades seja encontrado. Como desvantagem temos que esse número de comunidades deve ser definido no início do algoritmo, então é necessário saber, estimar ou determinar quantas comunidades existem naquele grafo específico antes de iniciar o algoritmo.

Outra desvantagem do método é que a cada repetição do processo o cálculo de centralidade das arestas deve ser refeito. Ao remover uma aresta importante, que tem vários menores caminhos passando por ela, o grafo é alterado formando outros menores caminhos. O novo cálculo é importante para aumentar a eficiência do processo, já que outros vértices passam a assumir o papel de comunicação entre as comunidades.

O funcionamento do algoritmo como descrito por Girvan e Newman em [67] é simplesmente descrito como:

1. Calcular o *betweenness* para todas as arestas.
2. Remover a aresta com o maior valor de *betweenness*.
3. Recalcular o *betweenness* para todas as arestas afetadas pela remoção.
4. Repetir o segundo passo até que não existam mais arestas.

Esse algoritmo tem como objetivo identificar e remover as arestas que servem de pontes de comunicação entre as comunidades. O resultado final é um grafo com algumas arestas removidas e a divisão em subgrafos do grafo original, onde cada subgrafo representa uma comunidades. Na próxima seção será apresentado o funcionamento do algoritmo aplicado ao problema, que precisa de modificações para ser utilizado no ambiente proposto.

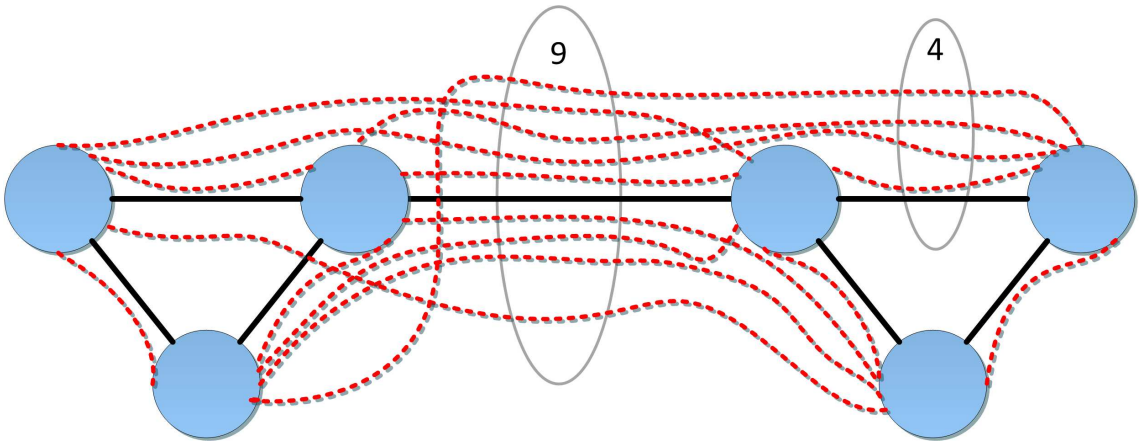


Figura 5.2: Exemplo de execução do algoritmo de Girvan-Newman.

A Figura 5.2 mostra a execução do algoritmo de Girvan-Newman para um caso simples. Nessa figura temos um grafo com 6 vértices, formando 2 comunidades de 3 vértices, sendo essas comunidades interligadas por uma aresta. As linhas contínuas representam as arestas e as linhas pontilhadas representam o menor caminho entre os vértices. É possível observar que a aresta que liga as comunidades é a que tem maior centralidade de todo o grafo, ou seja, passam 9 menores caminhos por essa aresta, e ao removê-la formamos duas comunidades de 3 vértices.

5.3.2 Modificações do algoritmo Girvan-Newman

O algoritmo Girvan-Newman de identificação de comunidades é adequado para o problema de identificação de máquinas virtuais relacionadas em um data center, só que algumas propriedades das redes de data center tornam o processo mais trabalhoso. Em um data center todas as máquinas virtuais podem se comunicar com todas as outras máquinas virtuais, e sem levar a topologia em consideração, essa comunicação é feita com o mesmo custo, portanto um grafo representando essa comunicação seria um grafo completo, onde todos os vértices estariam diretamente conectados e o algoritmo de Girvan-Newman não seria possível de ser aplicado.

Montar um grafo através da matriz de tráfego do data center pode ser feito de uma maneira que o tráfego entre as máquinas virtuais represente o peso de uma aresta. Arestas com pesos elevados são usadas intensamente e arestas com valores baixos ou nulos são usadas menos frequentemente. Ao fazer uma analogia com o método de Girvan-Newman podemos afirmar que o tráfego dentro de uma comunidade é mais elevado que o tráfego entre comunidades, já que uma comunidade se caracteriza pela intensa troca de dados entre os seus membros. Já as arestas com pesos mais baixos são usadas para transportar o tráfego entre comunidades ou não são usadas.

É possível usar outra abordagem para o problema onde os pesos dos vértices são usados para facilitar os cálculos que identificam as arestas que servem como comunicação entre as comunidades. Os pesos são inversamente proporcionais à importância da aresta, ou seja, se uma aresta possui peso baixo ela corresponde a uma aresta que é usada para a comunicação entre as comunidades.

Aplicando a ideia de usar os pesos para calcular a centralidade das arestas, modificamos o algoritmo para a cada rodada retirar as arestas com menores pesos e procurar as componentes conexas[1] do grafo. A cada rodada não é necessário recalcular os pesos, já que a retirada de uma aresta não irá influenciar o restante dos pesos.

O algoritmo modificado pode ser simplesmente descrito como:

1. Identificar as componentes conexas do grafo.
2. Tentar empacotar cada componente conexa.
3. Remover a aresta com o menor peso.
4. Repetir o segundo passo até que não existam mais arestas.

O algoritmo original de Girvan-Newman tem a complexidade alta devido a necessidade de calcular o valor da centralidade após cada remoção de arestas (encontrar todos os menores caminhos). O algoritmo proposto não precisa executar o cálculo da centralidade já que utiliza os pesos de cada uma das arestas para identificar a importância daquela aresta. Dessa maneira o algoritmo é menos custoso para ser calculado.

O algoritmo Girvan-Newman não estabelece um critério de parada para o processo de encontrar as comunidades, o algoritmo deve parar de ser executado quando o número de comunidades desejado seja encontrado, ou seja, o número de comunidades deve ser estabelecido antes da execução do algoritmo. Como o número de comunidades não é previamente conhecido em uma rede de data center, outra modificação foi feita para determinar corretamente o número de comunidades e para garantir que as máquinas virtuais que formam uma comunidade não sejam alocadas a uma partição de servidores que não tenha os recursos suficientes para mantê-las. A cada rodada do algoritmo todas as comunidades encontradas são testadas para ver se conseguem ser alocadas a alguma partição de servidores. Se for possível a alocação, essa comunidade é designada àquela partição e os vértices e as arestas

[1]Um grafo diz-se conexo se quaisquer que sejam os vértices distintos u e v existe sempre um caminho que os une. Quando tal não acontece o grafo diz-se desconexo. Um grafo desconexo pode ser divididos em subgrafos conexos, onde cada um dos vértices desse subgrafo estão interconectados. Esses subgrafos também são denominados de componentes conexas.

correspondentes são retirados do grafo que está sendo analisado. O método de escolha da ordem das partições a serem analisadas para a determinar a alocação das comunidades segue um dos algoritmos de *bin packing*, que são o *first-fit*, o *best-fit* e o *worst-fit*, descritos na Seção 4.2.4.

Dessa forma garante-se que as comunidades serão alocadas a partições com os recursos necessários e é definido um critério de parada para o algoritmo. O algoritmo sempre irá convergir para um resultado desde que existam recursos suficientes para alocar todas as máquinas virtuais. Agora será apresentado o algoritmo completo de localização e alocação de comunidades.

5.3.3 Algoritmo APoS

O algoritmo APoS pode ser dividido em passos. Primeiramente é obtido um grafo com informações do tráfego entre as máquinas virtuais e da quantidade de recursos de CPU e memória. O segundo passo trata cada umas da componentes conexas, tentando alocá-las aos servidores disponíveis. Então, um ciclo de repetições é iniciado e deve ser executado até que o grafo não tenha mais nenhuma aresta.

No ciclo primeiramente são identificados os componentes conexos do grafo. Cada um desses componentes tem a possibilidade de formar uma comunidade. É executado um teste para determinar se os componentes conexos cabem em alguma das partições existentes, se couber, esses componentes são comunidades e devem ser tratadas. As arestas e vértices de uma comunidade são alocados na partição designada e retirados do grafo de máquinas virtuais. Ao finalizar a análise de todos os componentes conexos se ainda existir alguma aresta no grafo de máquinas virtuais, as arestas com menor peso devem ser removidas e o ciclo de repetição deve ser reiniciado.

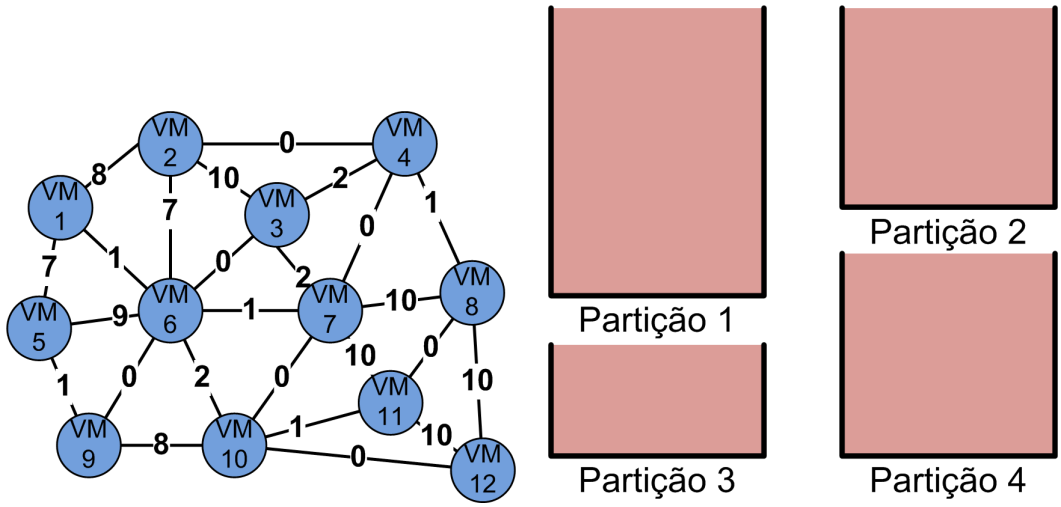
O Algoritmo 1 exibe o pseudocódigo criado para definir o posicionamento de serviços em data center. O algoritmo apresentado funciona da seguinte maneira: o passo 1 determina o grafo com todas as máquinas virtuais do data center, onde o vértice corresponde a um endereço, com definições de uso de CPU e memória e o peso de uma aresta corresponde ao tráfego entre dois endereços; o passo 2 determina as partições, com todos os seus dados de memória, CPU e quantidade de máquinas virtuais disponíveis para a alocação; o passo 3 é uma repetição que será executado enquanto existir alguma máquina virtual não alocada.

Algoritmo 1 Posicionamento de máquinas virtuais

$G \leftarrow$ Grafo de máquinas virtuais;
$P \leftarrow$ Partições de servidores;
while $G \neq \emptyset$ **do**
    Identificar todos os componentes conexos C em G;
    for all Sub-grafo $C \leq P$ **do**
        Retirar os vértices e arestas de C em G;
        Atualizar P com as novas alocações;
    end for
    Remover n arestas de G com o menor peso;
end while

O primeiro passo da repetição localiza todos os subgrafos, que são testados para ver se existe uma partição que comporte os subgrafos. No segundo passo da repetição é verificado para todas as componentes conexas se existe a possibilidade de alocação. Se existe, eles são designados a uma partição e os vértices e todas as suas arestas são removidos do grafo. Após a verificação, no passo 3 da repetição, as arestas com menores pesos são retiradas e o processo volta a realizar a repetição. O algoritmo somente é finalizado quando o laço de repetição termina, ou seja, quando o grafo não contém mais nenhum vértice.

Para ajudar na visualização do processo de posicionamento do APoS ilustramos o funcionamento do algoritmo. Primeiro, o grafo originário na matriz de tráfego é mostrado na Figura 5.3(a) e as partições, com nenhuma máquina virtual alocada, são representadas pela Figura 5.3(b).



(a) Grafo completo de todas as máquinas virtuais.

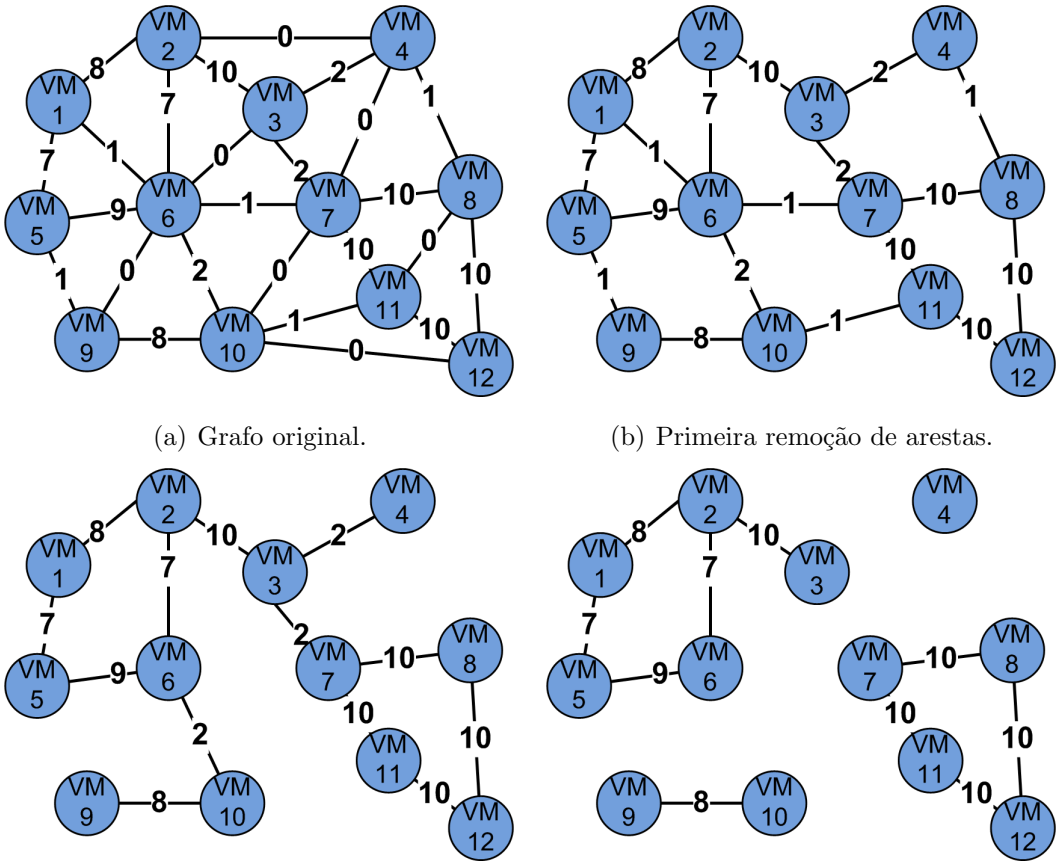
(b) Partições vazias.

Figura 5.3: Estado do data center ao iniciar o algoritmo.

O algoritmo é então executado, e a cada rodada remove as arestas com o menor valor do grafo. Inicialmente, na Figura 5.4(a) todos os vértices estão conectados. Por questões de melhor representação gráfica, algumas arestas com peso 0 não são exibidas na figura, já que o grafo completo contém 66 arestas.

Na primeira remoção de arestas, Figura 5.4(b), arestas com valor 0 são retiradas do grafo. Ainda não é possível encontrar uma componente conexa que tenha o tamanho necessário para ocupar somente uma partição. Na segunda remoção de arestas, Figura 5.4(c), as arestas com o valor 1 são retiradas. Da mesma forma que antes, ainda não é possível alocar um componente conexo a somente uma partição. Na última remoção de arestas, Figura 5.4(d), são retiradas as arestas com valor 2 e é possível observar a formação de comunidades, já que as componentes conexas separadas tem o tamanho necessário para serem alocadas nas partições.

Na Figura 5.5(a) é possível formar quatro comunidades que podem ser alocadas em quatro *clusters* distintos, já que o teste de verificação de tamanho mostrou que é possível alocar todas as comunidades nas partições disponíveis, como mostra a Figura 5.5(b). No final, temos todos os *clusters* de máquinas virtuais alocados a partições, como representado na Figura 5.5(c).



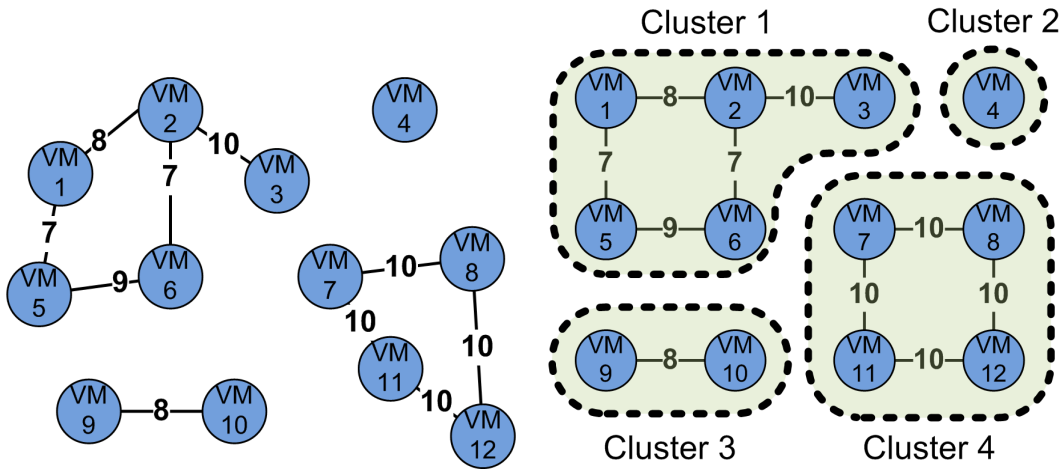
(a) Grafo original.

(b) Primeira remoção de arestas.

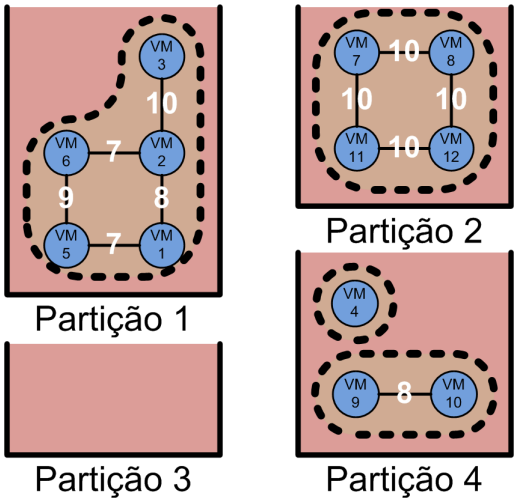
(c) Segunda remoção de arestas.

(d) Última remoção de arestas.

Figura 5.4: Etapas do processo de encontrar comunidades.



(a) Grafo final de todas as máquinas virtuais. (b) Clusterizando as comunidades encontradas.



(c) Clusters alocados às partições.

Figura 5.5: Estado do data center ao finalizar o algoritmo.

5.4 Saída de dados

A saída do algoritmo é o posicionamento final das máquinas virtuais nos servidores. Ao usar os dados de localização anteriores e os dados de localização do posicionamento proposto pelo algoritmo é possível sugerir as migrações que devem ser feitas para trazer os benefícios do APoS. Todas as migrações propostas devem ser analisadas previamente, já que as migrações trazem uma sobrecarga para a rede no momento em que estão sendo executadas, sendo preciso copiar todos os dados da máquina virtual do servidor com a localização antiga para o servidor na nova localização. Além disso ao migrar existe uma elevação no processamento da máquina de origem, que precisa se preparar para copiar os dados.

Para melhorar o algoritmo algumas regras devem ser seguidas. Uma máquina

virtual deve ter limites de migrações consecutivas, já que uma máquina com o padrão de comunicação com outras máquinas virtuais variável pode fazer com que ela seja relacionada a comunidades distintas a cada rodada. Migrar uma mesma máquina virtual em curtos espaços de tempo pode trazer instabilidade à máquina virtual, o que é indesejável para o cliente que possui essa máquina virtual, desperdiçando banda do data center em cada migração desnecessária. Um simples contador que tem o seu valor alterado a cada rodada pode ajudar a atenuar esse problema.

Outra restrição que deve ser seguida é a fixação de máquinas virtuais a servidores específicos. Esse requisito pode vir de várias exigências, mas as mais comuns são a necessidade de um *hardware* específico ou de uma capacidade maior de recursos de processamento, memória ou rede. Além dessas exigências essa restrição evita que máquinas virtuais sejam migradas com frequência, por exemplo, ao escolher uma máquina virtual ou um conjunto de máquinas virtuais que tem um tráfego elevado e já está posicionada de uma maneira a usar pouco o tráfego do núcleo conseguimos garantir o posicionamento correto dessas máquinas virtuais e o tempo de processamento do algoritmo será reduzido, já que menos cálculos de posicionamentos precisarão ser feitos.

Uma boa política para as migrações é migrar máquinas virtuais que consomem poucos recursos antes das máquinas que consomem mais recursos, já que migrar uma máquina que aloca muita memória é mais dispendioso do que migrar uma que usa pouco, e o resultado para a rede pode ser mais benéfico com as mudanças mais rápidas das máquinas virtuais. Mishra *et al.* descrevem alguma políticas para serem usadas na escolha das máquinas a serem migradas em seu trabalho *Dynamic resource management using virtual machine migrations* [11]. Migrar máquinas virtuais que transitam um maior volume de dados pelo núcleo da rede é a política mais agressiva possível para melhorar o uso do núcleo do centro de processamento de dados.

O algoritmo não consegue predefinir um local exato para as máquinas virtuais, sendo assim ele não garante que uma máquina virtual ou mesmo uma comunidade inteira será posicionada sempre no mesmo local a cada rodada do algoritmo. Para que o algoritmo evolua e se torne uma solução completa é necessário que essa questão seja resolvida, analisando os locais que façam com que menos migrações sejam necessárias. Uma abordagem seria executar o algoritmo diversas vezes e analisar em qual das soluções menos migrações devem ser executadas para a convergência do algoritmo.

O capítulo a seguir trata dos resultados obtidos através da simulação do algoritmo proposto e do ambiente de data center.

Capítulo 6

Resultados

Uma parte fundamental para avaliar e validar o trabalho desenvolvido é a simulação do APoS em diversos cenários, variando o tamanho do problema, tanto em número de servidores que compõe o data center quanto em números de máquinas virtuais que são executadas, o tipo de comportamento da comunicação entre os serviços, o método de procurar comunidades e o método de empacotar os *clusters*. Esse grau de liberdade para análise do problema é mais facilmente conseguido por simulações.

Testes em algoritmos na sua fase de desenvolvimento não devem ser feitos em ambientes operacionais, já que qualquer erro pode provocar a falha de todo o ambiente em que o teste é executado. Dessa forma, é necessário criar um ambiente que simule a realidade, sem que os efeitos indesejados atrapalhem o funcionamento de equipamentos que estão em operação. Montar um ambientes com equipamentos reais para mais de 10 mil máquinas virtuais é muito custoso, tornando essa abordagem impraticável. A solução encontrada foi simular o ambiente para colher os resultados.

Não foram identificados na literatura simuladores apropriados para o problema estudado, então foi necessária a construção de um simulador que modele apropriadamente a utilização dos enlaces da rede do data center dado o posicionamento dos serviços. Neste capítulo é apresentado o modelo de simulação utilizado e são fornecidos os detalhes sobre o simulador implementado. Em seguida, os resultados obtidos através da simulação são analisados.

6.1 Características dos dados de entrada

Antes de falar do simulador é necessário avaliar a entrada de dados do simulador. Como descrito na Seção 5.1 são necessários os dados da topologia do data center e os dados da matriz de tráfego do data center em uma janela de tempo. Não foi possível encontrar dados reais de matriz de tráfego de data centers para o uso no

simulador. Dados encontrados na literatura somente listavam dados referentes a uso de CPU e Memória, como o Google [78–80] ou arquivos de *jobs* de Hadoop, cujos dados se limitavam a quantidade de informação enviadas pela interface de *Ethernet* dos servidores que fazem parte do *cluster* do Hadoop [81].

Os dados de topologia foram retirados da literatura [20–28], observando-se que o mais comum é encontrar árvore de múltiplas camadas nos data centers [26–28]. Dessa maneira, optou-se por escolher uma topologia genérica, somente para observar os resultados do algoritmo no ambiente de data center.

O objetivo das simulações é determinar a eficácia do algoritmo sobre o problema de posicionamento de serviços, sendo assim, foi escolhida uma topologia com somente um nível. O uso de somente um nível não interfere no resultado e pode ser expandido para múltiplos níveis. O algoritmo é usado para reduzir o uso dos enlaces entre partições, evitando congestionamentos. Dessa forma, ao subdividir cada uma das partições não há mudança de tráfego nos enlaces do núcleo, a mudança ocorre somente nos enlaces entre as novas partições. Por exemplo, em um árvore na primeira divisão e execução do algoritmo é possível melhorar a distribuição do tráfego nos enlaces da camada do núcleo, sem se preocupar com os enlaces de camadas inferiores. Ao subdividir as partições, conseguimos melhorar a distribuição entre os enlaces da camada de agregação. Essas divisões podem ser feitas até o nível de servidor.

Para as simulações foi usada uma árvore de uma única camada, ou seja, cada uma das partições escolhidas estava ligada às outras a partir de um núcleo de apenas uma camada, como ilustra a Figura 6.1.

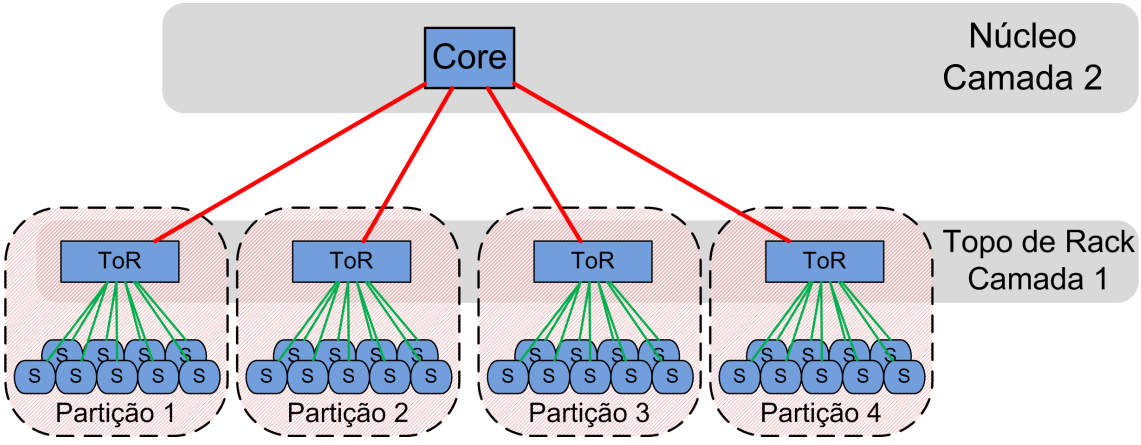


Figura 6.1: Exemplo de topologia utilizada na simulação.

Para a matriz de tráfego os dados de rede encontrados [27, 28, 78–81] não eram detalhados o suficiente, não contendo dados de tráfego de rede entre todas as máquinas e com escala suficiente para formar uma matriz contendo informações

com mais de 1.000 endereços, impossibilitando a criação de uma matriz de tráfego real. Os mesmos problemas são enfrentados de diversas maneiras pelos artigos da área. Os artigos relacionados a esse trabalho citados na Seção 2.4 usavam dados reais confidenciais de uma organização [12] ou criavam pequenos data centers para testes [32, 38] ou usavam dados gerados a partir de modelos desenvolvidos por outros estudos [34, 36].

Devido à escassez de dados reais em grande escala, principalmente para mais de 10 mil endereços, optou-se por gerar uma matriz de dados a partir de um modelo. Benson *et al.* [27, 28] têm trabalhos na área de levantamento de dados e caracterização de data centers reais. Em seus trabalhos, analisam 19 centros de processamento de dados com diferentes topologias e aplicações, descrevendo o tipo de tráfego encontrado nesses data centers. A partir desses estudos é usado um modelo para gerar uma matriz de tráfego, baseado nas conclusões de Benson *et al.*.

A caracterização proposta por Benson indica que o tempo entre a chegada de pacotes em uma determinada máquina no data center segue uma distribuição log-normal. O tráfego tem um comportamento de liga-desliga, caracterizado por períodos em que existem fluxos de dados seguindo uma distribuição log-normal e períodos em que não existe tráfego também seguindo uma distribuição log-normal. A partir dessas descobertas utilizamos essa distribuição para gerar os dados da matriz de tráfego estimada.

A distribuição log-normal exibida na Equação 6.1 representa a probabilidade do valor de um elemento sorteado. Nessa equação, o valor aleatório gerado dependerá de duas variáveis, a média, representada por μ e o desvio padrão, representado por σ. Os valores gerados não têm referência direta com valores de *bits* trafegados na rede, os dados são gerados de forma normalizada para facilitar o processo de interpretação da entrada do algoritmo.

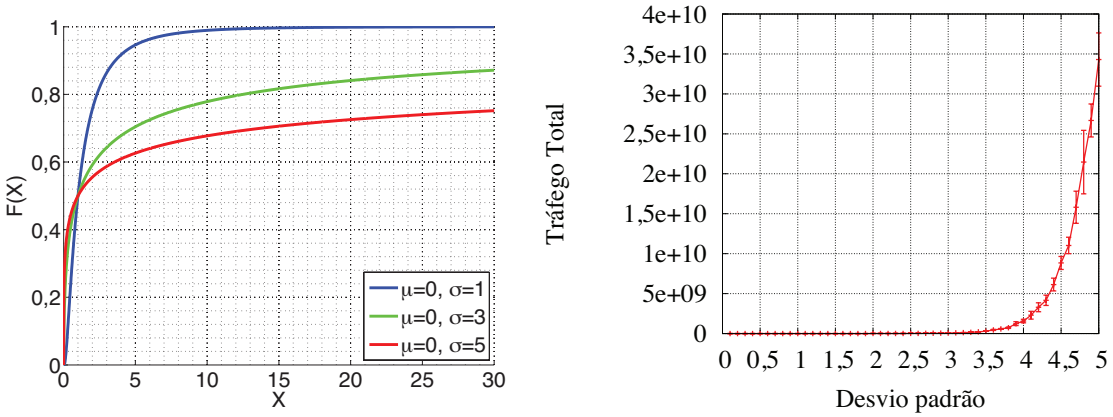
$$[H]f(x;\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln x-\mu)^2}{2\sigma^2}}, x > 0 \tag{6.1}$$

A média é usada para alterar o valor de referência da função, ou seja, adicionar um valor ao qual todos os números gerados tem como base. Ao escolher um valor maior do que zero, todos os resultados tem uma média diferente de zero. Como esse valor altera todos os números gerados da mesma forma, alterar o seu valor não tem valor na geração de uma matriz de tráfego, sendo assim, esse valor é fixado em 0 em todas as matrizes de tráfego geradas.

O valor de desvio padrão (σ) é alterado para mudar as características do tráfego simulado. Aumentando o valor do desvio padrão, aumentamos a chance do valor gerado ser mais elevado, ou seja, aumentamos os valores dos números gerados. A Figura 6.2(a) mostra a distribuição cumulativa de uma log-normal. Podemos obser-

var que quanto maior o valor do desvio padrão (σ) maior é a probabilidade do valor gerado ser alto. Nas simulações os valores de desvio padrão são variados entre 1 e 5. A Figura 6.2(b) mostra a média de 5 matrizes de tráfego geradas para cada valor de desvio padrão. Ao aumentar o valor do desvio padrão geramos mais tráfego no data center como um todo.

Esses valores específicos foram escolhido por questões de implementação do simulador. O simulador trabalha com números inteiros para a matriz de tráfego, então um (σ) menor do que 1 não traz diferença significativa entre os valores gerados pois com (σ) igual a 1 temos 80% dos números gerados com valores inferiores a 2. Já o (σ) igual a 5 foi estabelecido como limite superior pois números maiores geram valores muito grandes, ultrapassando os limites do compilador para as variáveis dos contadores no momento em que o tráfego total do data center é calculado.



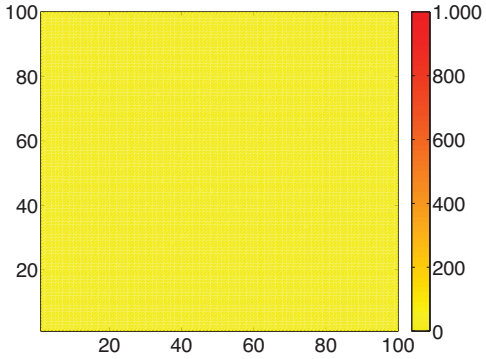
(a) CDF da log-normal.

(b) Tráfego total ao variar o desvio padrão.

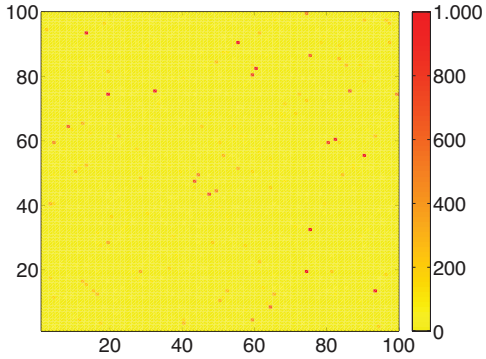
Figura 6.2: Variações devido ao desvio padrão.

A matriz de tráfego gerada com a média 0 e o desvio padrão de 1 pode ser caracterizada como um data center onde todas as máquinas virtuais têm probabilidades muito próximas de trocar a mesma quantidade de tráfego em uma janela de tempo. A Figura 6.3(a) ilustra uma matriz de tráfego com 100 elementos, onde o valor de cada elemento é representado por um mapa de calor de cores. Cores próximas ao amarelo representam valores próximos de 0, enquanto cores próximas ao vermelho representam valores altos, nos casos maiores que mil se a cor estiver no topo da escala.

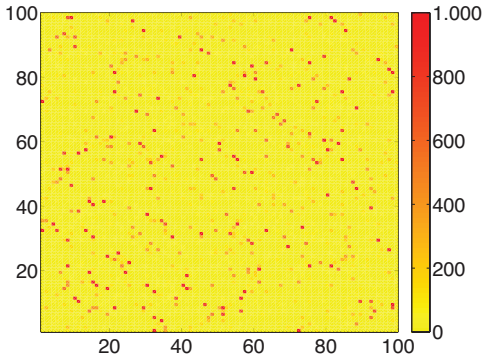
Ao variar o valor entre 1 e 5 conseguimos padrões distintos de tráfego. A Figura 6.3(b) representa a matriz de tráfego com o valor de desvio padrão igual a 2, a Figura 6.3(c) representa a matriz de tráfego com o valor de desvio padrão igual a 3, a Figura 6.3(d) representa a matriz de tráfego com o valor de desvio padrão igual a 4 e finalmente a Figura 6.3(e) representa a matriz de tráfego com o valor de desvio padrão igual a 5.



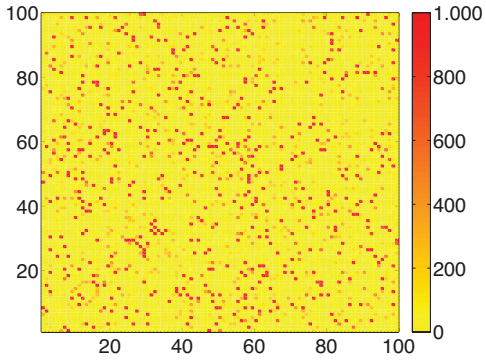
(a) Matriz de tráfego para $\sigma = 1$.



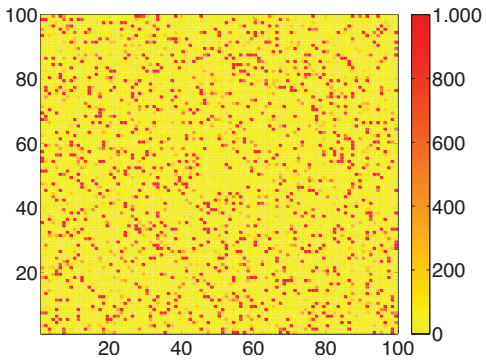
(b) Matriz de tráfego para $\sigma = 2$.



(c) Matriz de tráfego para $\sigma = 3$.



(d) Matriz de tráfego para $\sigma = 4$.



(e) Matriz de tráfego para $\sigma = 5$.

Figura 6.3: Matrizes de tráfego.

Variando entre 1 e 5 o valor do desvio padrão, conseguimos simular tipos distintos de data center. Valores baixos, menores do que 2,5, fazem com que o total de tráfego seja menor e que a diferença entre os dados enviados por duas máquinas distintas seja menor. Nesse caso temos um data center com tráfego bem distribuído entre todas as máquinas virtuais, dificultando a identificação de um grupo de máquinas virtuais que troque muitas informações entre si. Valores de desvio padrão acima de 2,5 tornam a matriz de tráfego do data center mais esparsa, ou seja, a diferença de tráfego trocado entre máquinas tende a ser maior. O valor total de dados trafegados

no data center é bem maior e mais concentrado, sendo mais fácil encontrar grupos de máquinas virtuais que troquem uma quantidade elevada de informações.

Dados como uso de CPU e memória de cada máquina virtual são sorteados através de uma função normal com média 0 e variância com valor 1000. O módulo do resultado obtido na distribuição foi dividido por 2 e o resto foi somado a 1 para ter o número de núcleos. De forma parecida temos os valores de memória. Para uma melhor visualização dos resultados, as CPUs foram dividas por núcleos necessários para a operação, variando entre 1 e 2, e a memória foi dividida em frações de 256MB, de 256MB até 2048MB. Os dados de CPU e memória de cada servidor são selecionados no início do algoritmo, somente por questões de simplicidade de visualização, tendo um valor definido de oito núcleos por servidor, 16GB de memória e capacidade para atender sete máquinas virtuais. Foram escolhidas sete máquinas virtuais baseado na escolha dos servidores com oito núcleos, dessa maneira cada máquina virtual será alocada a um núcleo e um núcleo será utilizado pelo hipervisor para controlar o ambiente. Nas simulações o número de máquinas por partição foi variado para modificar os tamanhos das partições.

6.2 O simulador

Alguns simuladores foram pesquisados para implementar o APoS e retirar dados de uso do núcleo da rede [82–85]. Simuladores a nível de pacotes não foram utilizados pois além da entrada de dados não ser a nível de pacote, o número de iterações que devem ser feitas para simular a troca de pacotes entre 16 mil máquinas durante uma janela de tempo não é viável, seria necessário muito esforço computacional para gerar os pacotes e interpretá-los, gerando muitos dados desnecessários para a análise, uma vez que é analisada somente a quantidade de tráfego trocada entre cada máquina. Uma matriz de tráfego para uma janela de tempo tem 1,5GB, essa mesma quantidade de dados a nível de pacotes geraria um volume muito maior de dados, dificultando a captura, o transporte, e processamento e o armazenamento dos dados.

Não foi possível colocar o algoritmo do APoS, com a modificação do algoritmo de Girvan-Newman e a matriz de tráfego como entrada no simulador CloudSim [83]. O simulador somente aceita como entrada uma matriz de latência e a topologia de dados de rede, por isso ele não foi utilizado.

Como nenhum dos simuladores analisados conseguia ter todas as características necessárias para a análise dos dados do problema, foi decidido criar um simulador específico para a situação. O simulador consiste basicamente em um contador do tráfego trocado entre as máquinas virtuais na janela de tempo, monitorando a quantidade de tráfego que fica dentro de cada partição e a quantidade de tráfego

Tabela 6.1: Opções de linha de comando do simulador

Comando	**Arg.**	**Opções**	**Função**
Número de máquinas virtuais	v	Número inteiro	Número de máquinas virtuais.
Opção de máquinas virtuais	o	Número inteiro	0 gerar Tráfego + CPU e Memória constantes; 1 para Tráfego, CPU e Memória aleatórios; 2 para entrar Tráfego, CPU e Mémórias por arquivos.
Número de partições	c	Número inteiro	Número de partições
Opção de partição	i	Número inteiro	0 para gerar partições com máxima ocupação possível; >1 para definir número de servidores.
Salvar matrizes	g	-	Salvar dados de CPU, Memória e Tráfego em arquivos.
Salvando matrizes para uso no input	e	-	Salvar dados de CPU, Memória e Tráfego em arquivos no diretório input.
Modo de debug	d	-	Modo Debug.
Modo de impressão na tela	p	-	Mostra na tela mais dados da execução.
Desvio padrão	s	Número inteiro	Valor do desvio padrão. O valor será dividido por 10 na entrada.
Valor de K	k	Número inteiro	Valor entre 0 e 100 para definir a percentagem de corte de arestas.
Bin Packing	a	Número inteiro	0 para *first-fit*, 1 para *best-fit* e 2 para *worst-fit*

atravessa o núcleo. Além disso o simulador deve conseguir ler uma topologia com as suas capacidades de CPU e memória e ter a habilidade de posicionar as máquinas virtuais de acordo com a escolha feita pelo algoritmo APoS e de forma aleatória.

Como o simulador precisa ser capaz de tratar a interação entre mais de 16 mil máquinas foi escolhido produzir um simulador utilizando a linguagem C++, uma das linguagens orientadas a objeto mais rápidas para a execução de cálculos numéricos.

O simulador foi escrito com um pouco mais de 2.000 linhas de código, sendo executado através de argumentos da linha de comando e leitura de arquivos de texto, gerando arquivos de texto com *logs* e informações na saída.

Os arquivos de entrada de dados são arquivos simples de texto em forma de matriz, com os dados numéricos separados por espaços e linhas separadas por caractere de nova linha. Os argumentos de linha de comando são apresentados na Tabela 6.1.

O simulador criado primeiramente lê uma matriz de tráfego, as informações de

CPU e memória das máquinas virtuais e as divisões da partições de servidores. Com esses dados executa o algoritmo APoS, indicando a posição de cada máquina virtual.

Após a alocação de todas as máquinas virtuais em partições de servidores as máquinas virtuais são relacionadas às comunidades e valores de uso de CPU e memória por comunidades são totalizados. As informações de alocação de comunidades, de máquinas virtuais, de uso de CPU e de memória em cada cluster são contabilizadas.

O tráfego total da matriz de tráfego é contabilizado. O tráfego interno e externo em cada uma das comunidades é contabilizado, assim como o tráfego interno e externo de cada partição. Com esses dados são calculadas as estatísticas de razão entre o tráfego intra/extra-clusters, diferenças de valores entre os algoritmos propostos, o aleatório e o APoS, entre outras métricas não relevantes para essa dissertação. O tempo de execução do algoritmo APoS também é calculado, com precisão de milissegundos.

Testes de sanidade foram executados durante e após a criação do simulador. Primeiro realizou-se o teste de leitura correta das informações, onde foi checado se os dados de entrada eram lidos corretamente pelo programa. Depois, foram realizados testes para verificar a implementação do algoritmo APoS, onde foi verificado se as máquinas eram alocadas corretamente, checando a quantidade de CPU e memória lidas pelo simulador. O algoritmo de posicionamento e de alocação foi testado para verificar o correto posicionamento das máquinas virtuais.

Também foram feitos testes para a verificação da consistência dos dados calculados. O somatório total dos fluxos de dados, o somatório de fluxos de dados em cada partição, e o somatório do fluxo de dados entre as comunidades encontradas.

As simulações foram executadas e comparadas com um algoritmo de alocação aleatória. Outros algoritmos propostos na literatura não puderam ser implementados por falta de requisitos como não usar dados de CPU ou memória nos cálculos ou por não conterem detalhes suficientes nas publicações para serem implementados.

O algoritmo usado para a comparação aloca as máquinas virtuais na ordem em que são apresentadas na matriz de tráfego, sendo que algoritmos de *bin packing* (*first-fit*, *best-fit* e *worst-fit*) são usados para garantir que os requisitos de CPU e memória sejam alocados corretamente, ou seja, as mesmas técnicas que são usadas no APoS são usadas no algoritmo randômico, menos a parte de identificação de comunidades.

Todas as simulações foram executadas em um computador pessoal equipado com um processador de seis núcleos Intel Core i7-3930K, rodando a 3,2GHz e equipado com 8GB de memória RAM. Embora o processador tenha múltiplos núcleos a implementação não foi desenvolvida de forma paralela, usando somente um dos núcleos. Durante as simulações até 10 instâncias da simulação foram executadas ao mesmo

tempo, não apresentando nenhum aumento significativo de tempo de execução. O número máximo de máquinas virtuais simuladas foi determinado pela quantidade de memória necessária para a execução. Ao simular com 16 mil máquinas virtuais somente a matriz de tráfego ocupa aproximadamente 1,5GB, o algoritmo todo sendo executado ocupa em torno de 6GB de memória, impossibilitando o aumento do tamanho ou a execução de mais de uma instância ao mesmo tempo.

6.3 Simulações

Para avaliar o desempenho do APoS, uma série de testes são executados variando o número de máquinas virtuais na topologia, a razão com que as arestas são removidas a cada etapa, o tamanho das partições e o tipo de data center através da mudança do desvio padrão. Em todos os testes o valor médio da matriz de tráfego gerada é 0, o modelo de servidor utilizado tem oito núcleos e 16GB de memória. As simulações são repetidas 5 vezes para cada um dos valores e são usados intervalos de confiança de 95%. As máquinas virtuais são simuladas usando um ou dois núcleos e precisando entre 256MB e 2048MB de memória. Todas as outras especificações de cada simulação são detalhadas ao exibir os resultados. Onde não existe indicação do tipo de algoritmo de *bin packing* utilizado nas simulações, existem poucas diferenças entre os resultados, dessa forma, os gráficos são gerados utilizando o algoritmo *worst-fit*. Uma avaliação entre os três algoritmos é exibida neste capítulo, mostrando que o algoritmo escolhido aloca de forma mais uniforme as máquinas virtuais nos clusters.

Na Figura 6.4 mostramos o tempo de execução do algoritmo. Esse tempo somente mostra o tempo da execução do algoritmo de posicionamento, a coleta de dados e a leitura dos dados pelo simulador não estão incluídos. O tempo de leitura é pequeno comparado com o tempo de simulação. Dois aspectos devem ser notados: o aumento do tempo exponencialmente com o número de nós e a redução do tempo de execução ao modificar a quantidade de arestas removidas por rodada, representada pela percentagem de arestas retiradas sobre o total de arestas existentes, o eixo X da Figura 6.4.

Para gerar a Figura 6.4 foram usadas 20 partições contendo de 10 até 100 servidores em cada uma e um desvio padrão com valor 4. Os valores da matriz de tráfego, de CPU e memória são gerados aleatoriamente de forma que a capacidade de processamento do data center fique entre 90% e 100% nas simulações. O algoritmo de empacotamento utilizado é o *worst-fit*. Cada simulação é repetida 10 vezes para encontrar a média do tempo utilizado.

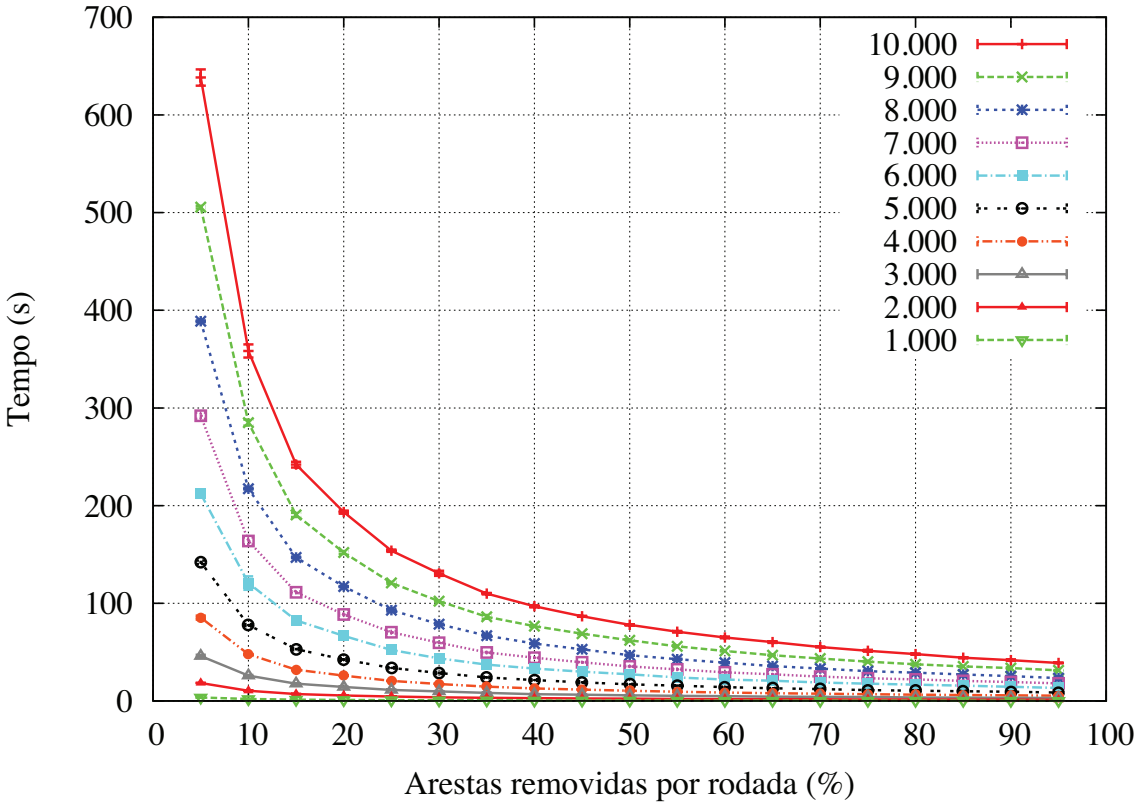


Figura 6.4: Tempo de simulação do APoS.

A quantidade de arestas removidas por rodada representa o fator com que as arestas são removidas durante a execução do APoS. O algoritmo funciona removendo um número de arestas por rodada para encontrar as comunidades. A métrica usada foi escolher a porcentagem de arestas da rodada a serem removidas. Por exemplo, se existem 100 arestas e se escolhe remover 10% das arestas, as 10 arestas que têm o menor peso serão removidas naquela rodada. A escolha desse percentual interfere diretamente no tempo de processamento, já que altera diretamente o número de rodadas e a qualidade das comunidades encontradas; porque ao retirar uma grande quantidade de arestas de uma só vez, informações sobre comunidades podem ser perdidas, ou seja, representa um compromisso entre o tempo de execução e a qualidade da solução encontrada.

Para avaliar o desempenho da parte responsável por encontrar as comunidades no algoritmo proposto, analisamos o número de comunidades encontradas quando variamos a quantidade de arestas removidas por rodada e quando variamos o desvio padrão. A quantidade de arestas removidas por rodada afeta diretamente a quantidade de comunidades encontradas. A Figura 6.5 mostra a quantidade de comunidades encontradas ao modificar o percentual de arestas removidas por rodada.

Os resultados da Figura 6.5 consideram 1.000 máquinas virtuais, 20 partições com 10 servidores cada e um desvio padrão com valor 4. Os valores da matriz de

tráfego, de CPU e memória são gerados aleatoriamente de forma que a capacidade de processamento do data center fique entre 90% e 100% nas simulações. O algoritmo de empacotamento utilizado é o *worst-fit*. Cada simulação é repetida 10 vezes para encontrar a média dos valores.

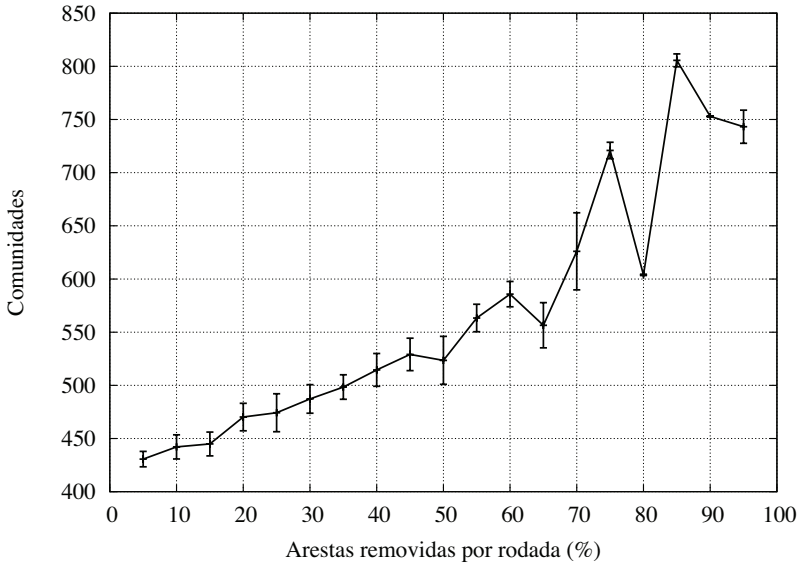


Figura 6.5: Comunidades encontradas por percentual de arestas removidas.

Ao observar a Figura 6.5 a menor quantidade de comunidades encontrada é quando retiramos 5% das arestas por rodada. Encontrar menos comunidades é desejável, pois temos mais máquinas virtuais relacionadas a cada uma das comunidades. A partir da remoção de 35% das arestas por rodada passamos a ter comunidades com o valor médio de menos de duas máquinas virtuais por comunidade, ou seja, mais da metade das comunidades tem somente um elemento, são comunidades triviais[1]. Entretanto ao remover 35%, ao invés de 10%, melhoramos o tempo de execução do algoritmo em mais de cinco vezes nos melhores casos. Nas simulações subsequentes são removidas 10% das arestas, já que as simulações têm o objetivo de testar o funcionamento do APoS e estamos em um ambiente controlado, não existindo restrições com o tempo de execução ou com o desempenho do algoritmo.

A avaliação do desempenho do APoS através da modificação do desvio padrão pode ser vista na Figura 6.6. O desvio padrão muda o tipo de matriz de tráfego, sendo mais fácil encontrar comunidades quando a diferença entre os valores gerados é maior. O número de máquinas virtuais é variado em 1.000, 4.000, 7.000 e 10.000, são retiradas 10% das arestas por rodada e temos 20 partições, cada uma com o número necessário para que o data center como um todo tenha 100% de utilização.

[1]A solução trivial do problema de identificação de comunidade é onde existem n elementos e n comunidades, ou seja, cada elemento forma uma comunidade trivial.

Para gerar a Figura 6.6, 10% das arestas foram removidas por rodada, e foi usado um desvio padrão com valor 4. O algoritmo de empacotamento utilizado é o *worst-fit*. Cada simulação é repetida 10 vezes para encontrar a média dos valores.

A Figura 6.6 mostra que com o desvio padrão baixo ($\sigma < 1$) o APoS tem dificuldades em formar grandes comunidades diferentes da solução trivial. Acima deste desvio padrão ($\sigma > 1$), o número de comunidades estabiliza independente do desvio padrão. Isso significa que o APoS consegue formar comunidades quando o tráfego trocado entre as máquinas virtuais tem diferenças mais significativas. Quanto existe uma grande concentração de tráfego existem menos máquinas virtuais trocando muito tráfego, dessa forma é mais fácil encontrar comunidades.

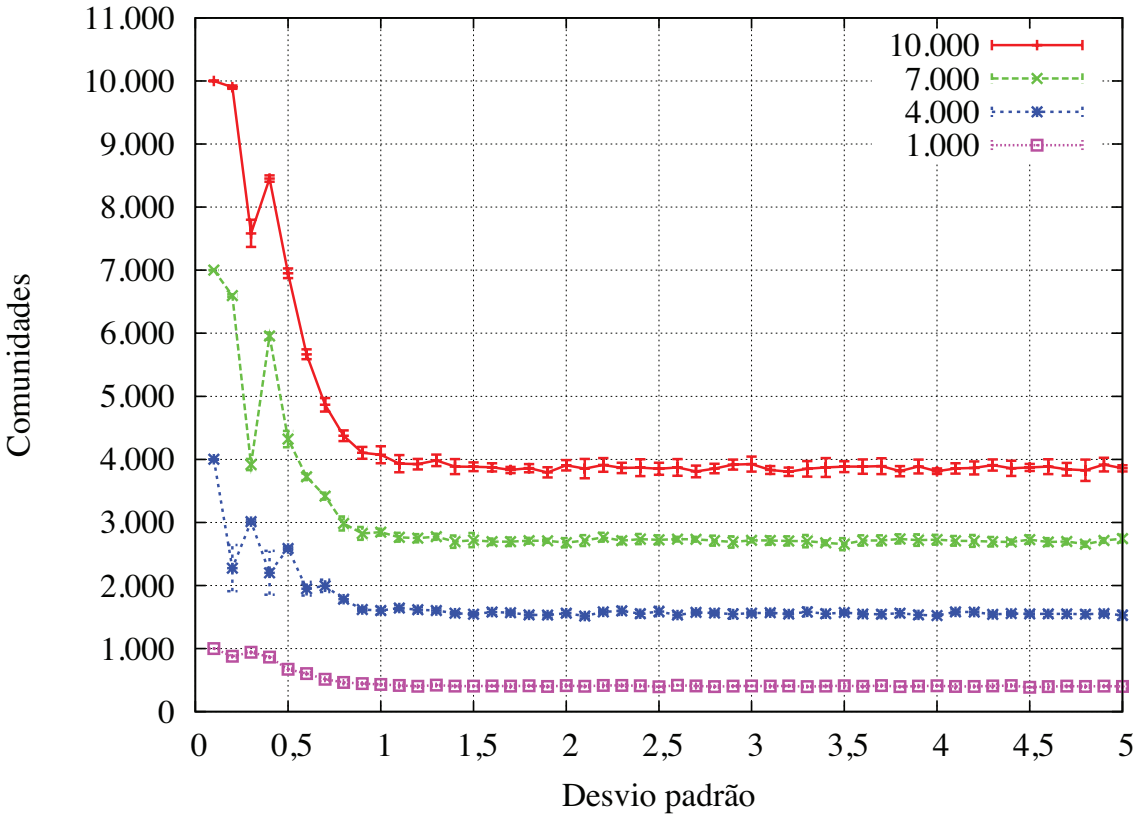


Figura 6.6: Comunidades encontradas por desvio padrão.

Para verificar os benefícios que o algoritmo APoS pode trazer para data centers executou-se uma simulação variando o desvio padrão da matriz de entrada. O objetivo é simular o tráfego em diferentes tipos de centros de processamento de dados, os que tem a característica de poucas mudanças no padrão e pouco tráfego, com níveis baixos de desvio padrão ($\sigma < 2,5$) e os centros de processamento de dados que tem um nível elevado de mudanças de padrão e um volume alto de tráfego, tendo desvio um desvio padrão alto ($\sigma > 2,5$).

Para gerar a Figura 6.7, 10% das arestas foram removidas por rodada, foram

usadas 1.000 máquinas virtuais, 20 partições de 10 servidores em cada uma. Os valores da matriz de tráfego, de CPU e memória são gerados aleatoriamente de forma que a capacidade de processamento do data center fique entre 90% e 100% nas simulações. O algoritmo de empacotamento utilizado é o *worst-fit*. Cada simulação é repetida 10 vezes para encontrar a média dos valores.

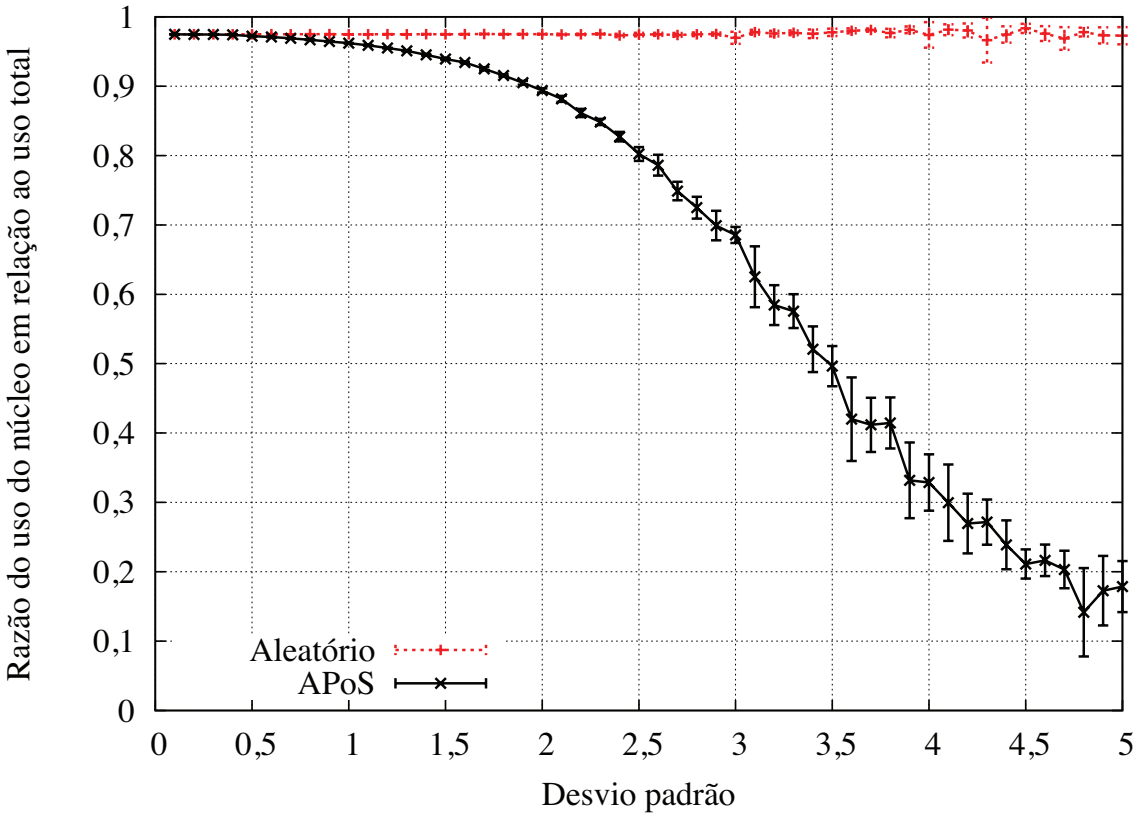


Figura 6.7: Tráfego do núcleo sobre o tráfego total.

A Figura 6.7 mostra a proporção do tráfego que trafega no núcleo sobre o total do tráfego no data center, ou seja, é a razão entre a quantidade de tráfego extra-cluster de todos os clusters sobre o total de tráfego que trafega no data center. Podemos ver uma melhora significantiva no uso do APoS sobre o posicionamento sem levar em consideração o tráfego entre as máquinas virtuais. Quando o volume de tráfego é elevado ($\sigma > 2,5$) o APoS reduz a necessidade do tráfego utilizar caminhos que passam pelo núcleo para até 20% do total que circula no data center, fazendo com que essa troca de dados ocorra dentro de uma partição, ou seja, próxima dos servidores.

Ampliando essa análise para mostrar a diferença que podemos encontrar ao usar números distintos de máquinas virtuais e de desvio padrão apresentamos a Figura 6.8. Essa figura mostra como o número de servidores, máquinas virtuais e a matriz de tráfego do data center podem influenciar o resultado do APoS.

Nessa simulação é variado o número de máquinas virtuais contidas em um data center ao mesmo em que se aumenta o número de servidores para sustentar essas máquinas virtuais. O número de servidores é calculado a partir do número de máquinas virtuais, para que a ocupação fique entre 90% e 100%.

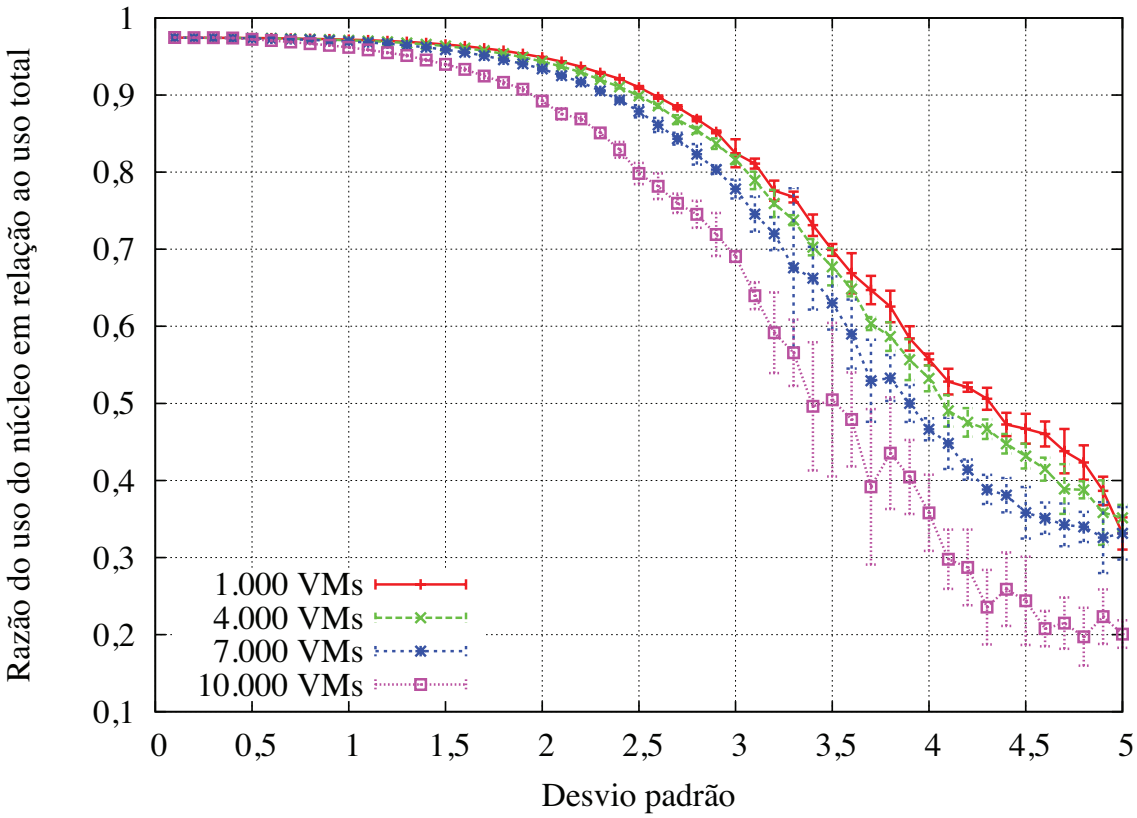


Figura 6.8: Variando o número de máquinas virtuais.

Quanto maior o número de máquinas virtuais em um data center, maior é a possibilidade de melhorar o uso do núcleo. Isso se explica pela topologia, nesse caso uma partição tem mais máquinas virtuais ao crescer o número de servidores, conseguindo agrupar comunidades com um número maior de máquinas virtuais. Todas as curvas têm o mesmo comportamento, mostrando que o tamanho do data center não influencia diretamente no funcionamento do algoritmo.

Para gerar a Figura 6.8, 10% das arestas foram removidas por rodada, foram usadas com 1.000, 4.000, 7.000 e 10.000 máquinas virtuais, 20 partições com 10, 40, 70 e 100 servidores respectivamente. Os valores da matriz de tráfego, de CPU e memória são gerados aleatoriamente de forma que a capacidade de processamento do data center fique entre 90% e 100% nas simulações. O algoritmo de empacotamento utilizado é o *worst-fit*. Cada simulação é repetida 10 vezes para encontrar a média dos valores.

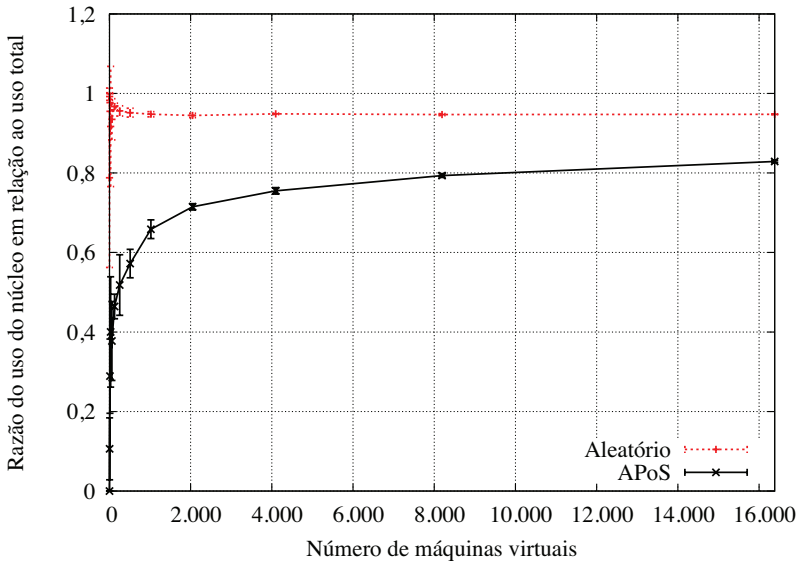


Figura 6.9: Variando o desvio padrão e o número de máquinas virtuais.

Para avaliar a influência da escala do data center, executamos uma simulações variando o número de máquinas virtuais e o número de servidores por partição.

Para gerar a Figura 6.9, 10% das arestas foram removidas por rodada, usamos entre 4 (2^2) e 16384 (2^{14}) máquinas virtuais, 10 partições com o número de servidores variando de 1 (4 máquinas virtuais) até 328 (16384 máquinas virtuais), para conseguir acomodar todas as máquinas virtuais de forma que ocupem entre entre 90% e 100% e um desvio padrão com valor 3. Os valores da matriz de tráfego, de CPU e memória são gerados aleatoriamente de forma que a capacidade de processamento do data center fique entre 90% e 100% nas simulações. O algoritmo de empacotamento utilizado é o *worst-fit*. Cada simulação é repetida 10 vezes para encontrar a média dos valores.

Quando o número de máquinas virtuais é incrementado, o número de servidores necessários para alocá-las também é aumentando. Nesse cenário a utilização do núcleo tende a se estabilizar em torno de 82% do uso do núcleo. Esse valor mostra que no pior caso, quando temos 16.384 máquinas virtuais, podemos esperar uma melhora de 12,5% sobre o algoritmo que não leva em consideração o tráfego da rede para o posicionamento das máquinas virtuais.

Para avaliar o nível de uso do data center, é simulado um caso com número fixo de máquinas virtuais, variando o número de servidores. Ao aumentar o número de servidores sem modificar a demanda das máquinas virtuais, criamos um ambiente em que os recursos são abundantes. A Figura 6.10 mostra essa situação, onde todos os algoritmos de alocação são testados, já que existe uma diferença entre eles nesse caso. Quando a utilização é entre 90% e 100% temos poucas diferenças entre os três algoritmos, com variações em torno de 2% entre os algoritmos de alocação.

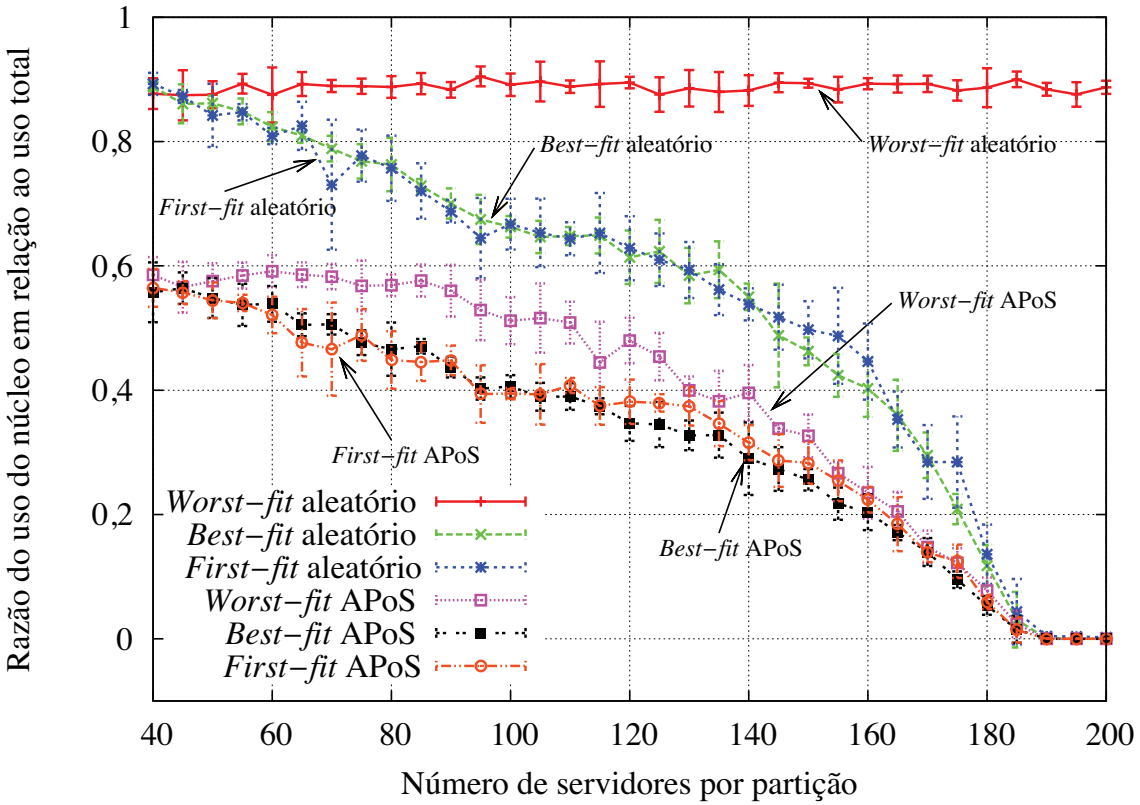


Figura 6.10: Variando o número de servidores por partição.

Na Figura 6.10, retiramos 10% das arestas por rodada, usamos 1000 máquinas virtuais, cinco partições com o número de servidores variando de 40 até 200 e um desvio padrão com valor 3. Os valores da matriz de tráfego, de CPU e memória são gerados aleatoriamente. Com 40 servidores por partição temos perto de 93% de uso do data center, enquanto que com 200 servidores temos menos de 20% de uso do data center. Cada simulação é repetida 10 vezes para encontrar a média dos valores.

Ao chegar a níveis mais baixos de uso, cada heurística de empacotamento das comunidades passa a ter um comportamento distinto, mas, com o uso muito baixo, todas (menos a *worst-fit* aleatória) tendem a não ter tráfego no núcleo da rede.

No algoritmo proposto ao aumentar a quantidade de servidores disponíveis para a mesma carga (adicionando o mesmo número de servidores em cada uma das partições), o uso do núcleo cai para próximo de 0% já que as partições tem a sua capacidade aumentada, permitindo que praticamente todas as máquinas virtuais sejam alocadas a somente uma partição.

A Figura 6.10 também mostra a necessidade de não precisar exibir os resultados de todas as heurísticas de empacotamento nas simulações anteriores. Em ambientes onde o data center está perto do seu limite, acima de 90% de uso, a diferença entre os três algoritmos não influencia o resultado tanto do algoritmo proposto quanto do algoritmo aleatório.

As diferenças entre o tráfego que circula no núcleo ao abaixar o nível de uso do data center podem ser explicadas pela maneira com que as comunidades ou máquinas virtuais são alocadas às partições. Quando o data center está próximo ao limite de sua capacidade, as heurísticas de empacotamento não trazem diferenças significativas entre cada uma delas, pois os servidores estão todos próximos do seu uso máximo. Ao reduzir a lotação do data center, as heurísticas passam a ter um papel importante.

Enquanto as heurísticas *first-fit* e a *best-fit* tendem a concentrar a alocação de máquinas virtuais em poucas partições, a heurística *worst-fit* tende a distribuir o tráfego entre todas as partições.

Para observar essa tendência, a ocupação do *cluster* com 1.000 máquinas virtuais e com 150, 300 e 1.500 servidores na topologia é exibida para as três heurísticas de alocação, com 10 servidores por partição e 95% de uso na Figura 6.11(a), com 20 servidores por partição e 45% de uso na Figura 6.11(b) e com 100 servidores por partição e 10% de uso na Figura 6.11(c). Cada servidor consegue alocar sete máquinas virtuais, por isso podemos alocar sete vezes o número de máquinas virtuais em uma partição.

A mesma conclusão pode ser retirada do algoritmo de alocação aleatória baseado nas heurísticas apresentadas, por isso não são exibidos nos gráficos.

Apresentamos com detalhe de CPU, memória e máquinas virtuais a alocação de máquinas virtuais pelo APoS. Nas Figuras 6.12(a), 6.12(b) e 6.12(c) temos 95% de uso do data center, cada uma com uma heurística de empacotamento das comunidades. Nessa situação não se tem grandes diferenças na distribuição de máquinas virtuais, apesar de ser possível perceber uma partição com menos máquinas virtuais alocadas nas heurísticas *first-fit* e *best-fit*, enquanto que a heurística *worst-fit* distribui entre todas as partições as máquinas virtuais.

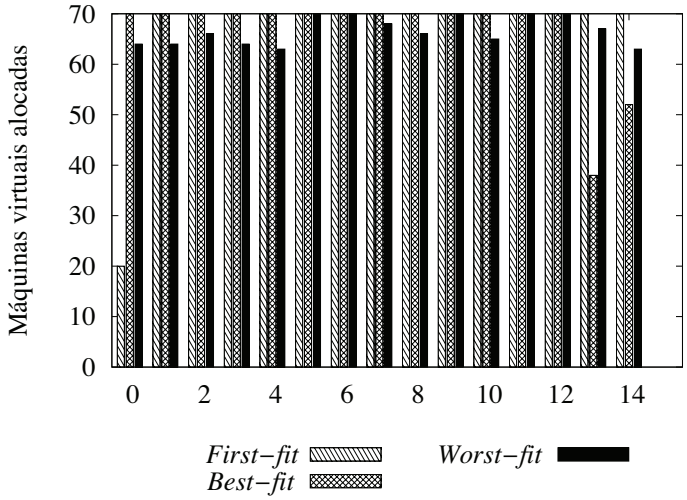
Nas Figuras 6.13(a), 6.13(b) e 6.13(c), temos 45% de uso do data center, cada uma com uma heurística de empacotamento das comunidades. Nessa situação temos grandes diferenças na distribuição de máquinas virtuais, já que nas heurísticas *first-fit* e *best-fit* temos metade das partições com 100% de uso enquanto a outra metade não tem nenhuma máquina virtual alocada, concentrando o tráfego. Essa característica é boa, pois melhora o uso do núcleo, já que as partições sem máquinas não são usadas, mas temos um intenso uso dos enlaces das partições em uso. Essas heurísticas podem ser usadas para diminuir o consumo de energia elétrica do data center, já que temos partições de servidores inteiras que podem ser desligadas sem que se influencie as máquinas virtuais no data center. A heurística *worst-fit* distribui entre todas as partições as máquinas virtuais, deixando todas com ocupação próxima a 50%. Nessa situação temos um melhor uso dos enlaces do núcleo, pois o algoritmo consegue identificar e agrupar comunidades, mantendo grande parte do

tráfego dentro das partições. Como existe banda disponível no núcleo e existem recursos não utilizados em cada uma das partições, podemos concluir que essa distribuição tolera melhor mudanças bruscas nos padrões de tráfego do data center, conseguindo atender de forma mais fácil a alguma demanda crescente de recursos.

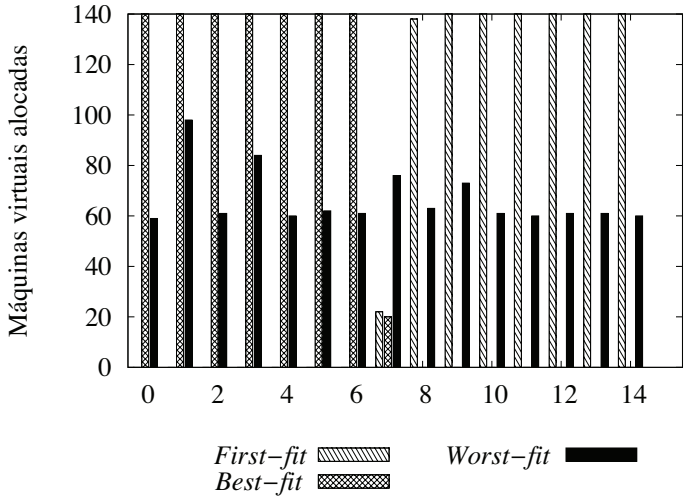
Nas Figuras 6.14(a), 6.14(b) e 6.14(c), temos 10% de uso do data center, cada uma com uma heurística de empacotamento das comunidades. Nessa situação temos grandes diferenças na distribuição de máquinas virtuais, já que nas heurísticas *first-fit* e *best-fit* temos as máquinas virtuais concentradas em duas partições. Temos os mesmos benefícios que o caso anterior, com uma possibilidade ainda maior de economia de energia. A heurística *worst-fit* tem um comportamento interessante nesse caso, agrupando uma grande comunidade em uma única partição e as demais comunidades distribuídas igualmente entre as partições restantes.

No caso do algoritmo aleatório, aplicando-se a heurística *worst-fit*, o algoritmo sempre vai escolher dividir as máquinas virtuais entre todas as partições, fazendo com que o uso do núcleo seja sempre intenso.

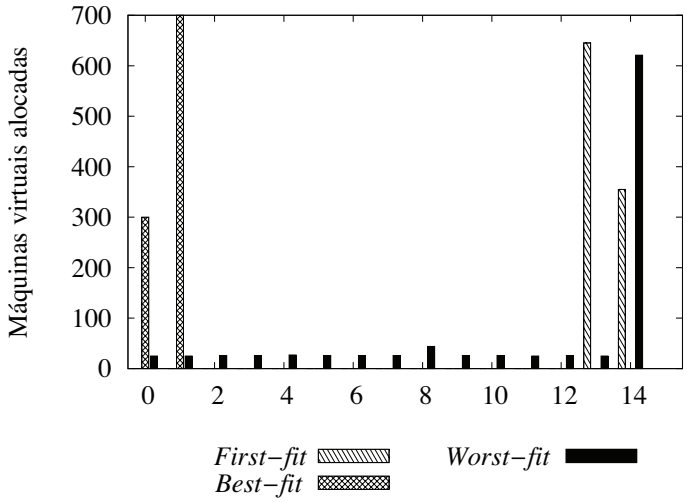
Voltando à Figura 6.10, escolheu-se aplicar o *worst-fit* ao APoS supondo que em um data center em que o padrão da matriz de tráfego ou os requisitos de processamento e de memória sejam bastante mutáveis ao longo das horas ou um que se deseja utilizar melhor os recursos de todas as máquinas virtuais, sem que haja alguma sobrecarga, mesmo que momentânea. Os algoritmos *first-fit* e *best-fit* são praticamente idênticos, somente mudando a ordem dos índices das partições para esse caso, e devem ser usados em data centers onde o padrão de tráfego e os requisitos de processamento e de memória sejam bem definidos ou bem previsíveis com o tempo. Essas heurísticas têm a vantagem de trazer a economia de energia se associadas a um controle de energia dos servidores com nenhuma carga de máquinas virtuais.



(a) 10 Servidores por partição, 95% de uso.

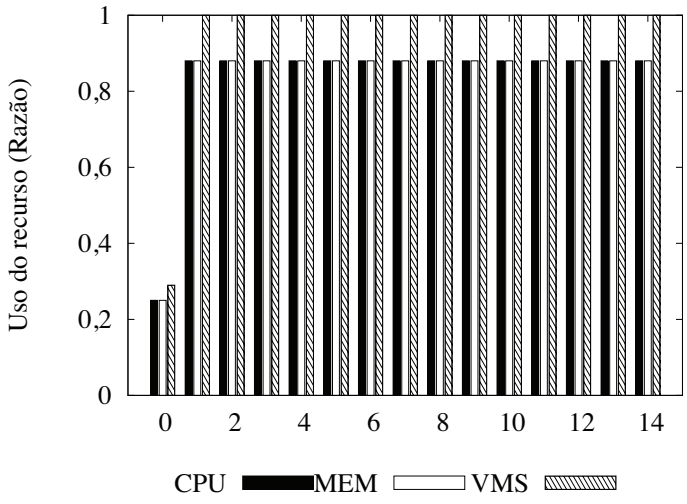


(b) 20 servidores por partição, 45% de uso.

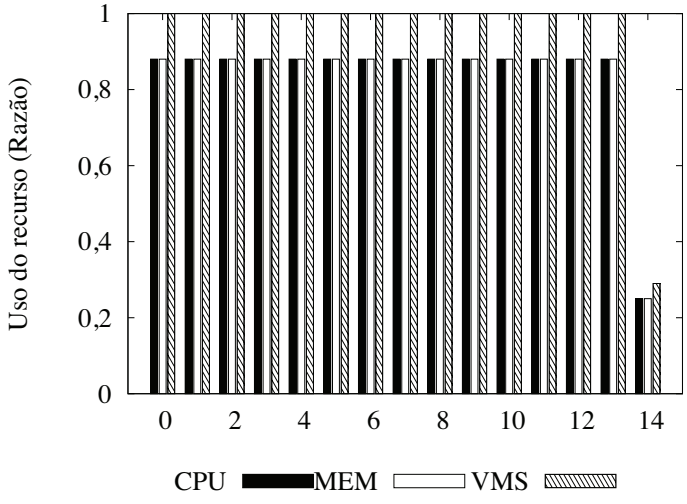


(c) 100 servidores por partição, 10% de uso.

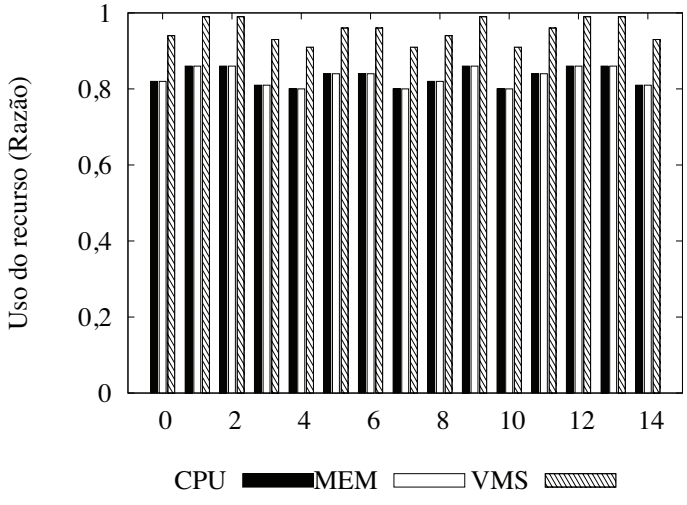
Figura 6.11: Alocação de máquinas virtuais por partição.



(a) *First-fit* para 95%.

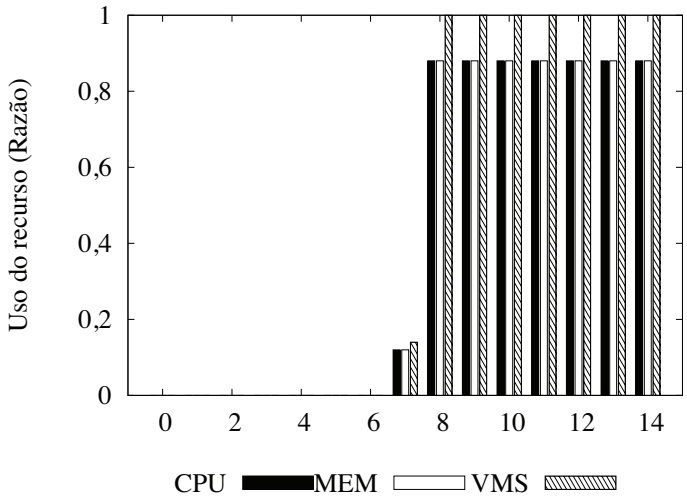


(b) *Best-fit* para 95%.

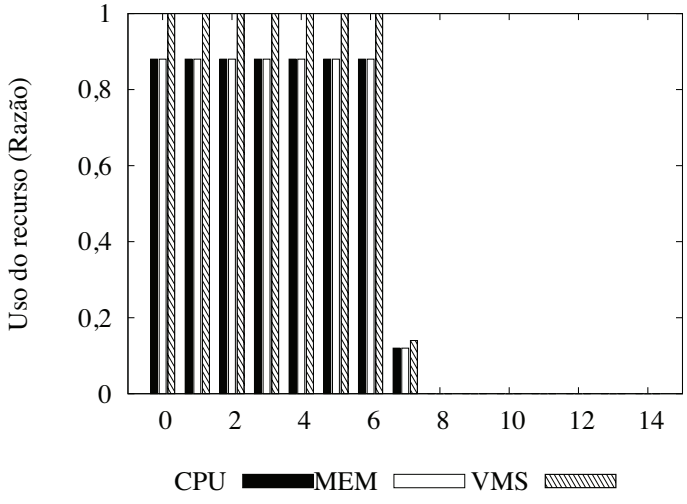


(c) *Worst-fit* para 95%.

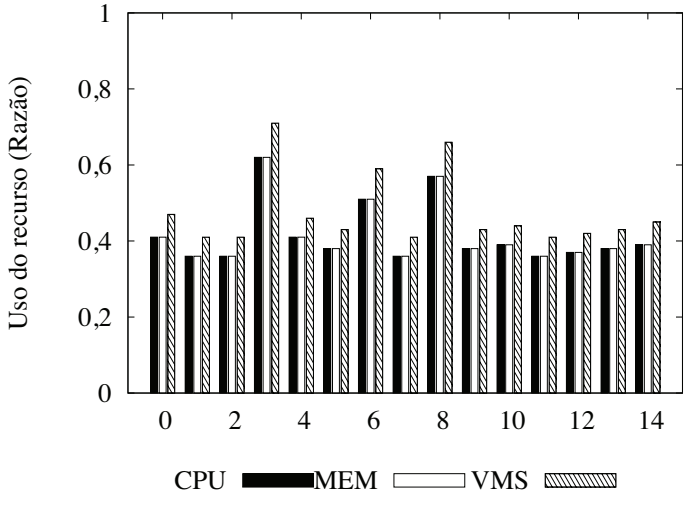
Figura 6.12: Alocação de CPU, memória e máquinas virtuais por *cluster* com APoS, 95% de uso.



(a) *First-fit* para 45%.

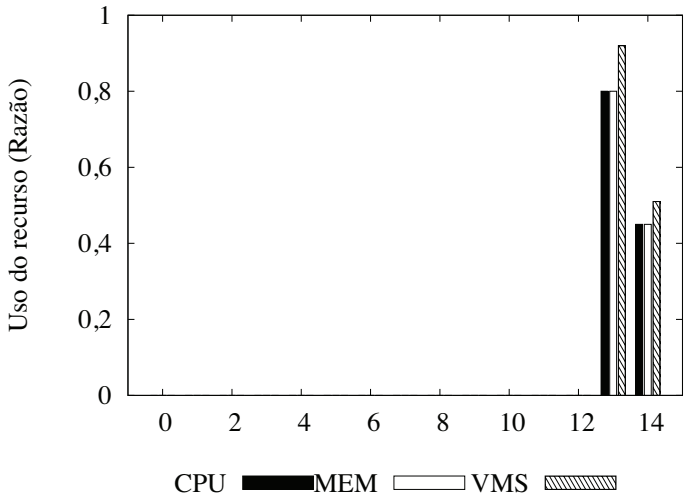


(b) *Best-fit* para 45%.

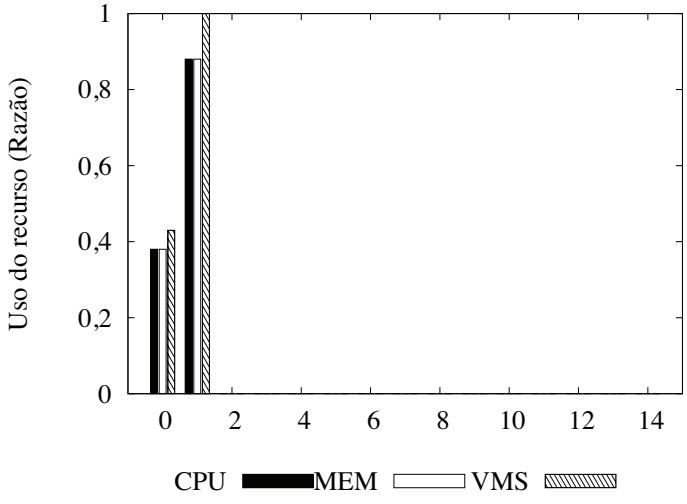


(c) *Worst-fit* para 45%.

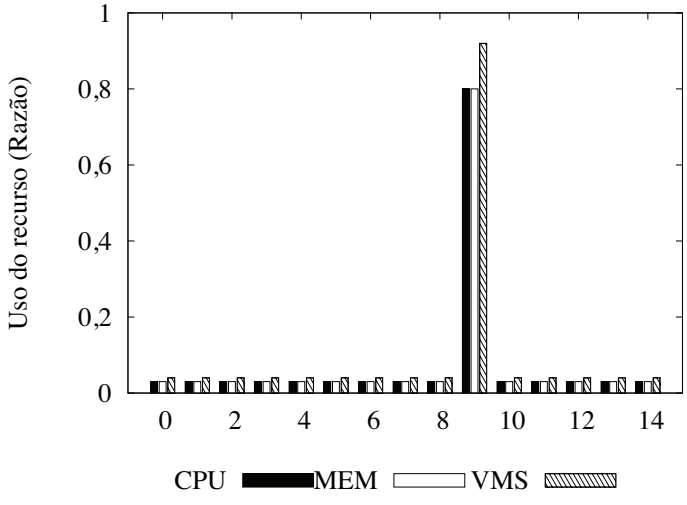
Figura 6.13: Alocação de CPU, memória e máquinas virtuais por *cluster* com APoS, 45% de uso.



(a) *First-fit* para 10%.



(b) *Best-fit* para 10%.



(c) *Worst-fit* para 10%.

Figura 6.14: Alocação de CPU, memória e máquinas virtuais por *cluster* com APoS, 10% de uso.

Capítulo 7

Conclusão

Nesta dissertação foram apresentados os ambientes mais comuns de centros de processamento da dados, identificando o seu funcionamento e as práticas comuns para as ligações em rede. A partir desse estudo foi proposto um algoritmo para posicionamento de máquinas virtuais dependente da matriz de tráfego, que procura direcionar o tráfego que precisaria passar por partes em que a banda de dados é mais escassa para áreas em que o tráfego de dados consegue fluir com uma banda de dados maior.

Foi proposto o Algoritmo de Posicionamento de Serviços (APoS) para atenuar o problema de posicionamento de máquinas virtuais em data centers que não consideram os dados de fluxos de dados trafegados entre as máquinas virtuais do data center, objetivando diminuir a carga do núcleo da rede. A partir da coleta de dados da CPU, memória e do tráfego entre as máquinas virtuais contidas no ambiente e de dados da topologia e dos servidores que compõem o data center, o algoritmo analisa e posiciona as máquinas virtuais, retirando desses dados características para relacionar as máquinas virtuais a comunidades.

O APoS usa o conceito de comunidades para encontrar máquinas virtuais que trocam uma grande quantidade de informação entre si e usa dados da topologia para melhor localizar essas comunidades, tentando gerar a menor quantidade possível de tráfego no núcleo da rede do data center. O APoS, diferente dos algoritmos encontrados na literatura, foi desenvolvido para ser ser executado em ambientes de data center com milhares de máquinas virtuais, sem precisar previamente agrupá-las para diminuir o tempo necessário para a computação das posições, conseguindo usar dados de CPU, memória e tráfego para essa escolha. Dessa forma, o APoS foi projetado para ser um algoritmo *online*, conseguindo ser executado em poucos instantes. Esse diferencial pode ser usado para tratar o posicionamento de forma instantânea, aplicado principalmente a ambientes em que as condições de tráfego e de uso das máquinas virtuais sejam bastante variáveis em um dia, respondendo de maneira rápida a essas mudanças.

Para melhor avaliar o APos foi desenvolvido um simulador do ambiente de data center, onde diversas variáveis foram monitoradas, como o número de comunidades, o tráfego entre as partições da topologia, a quantidade de recursos alocados, entre outras variáveis. Nesse simulador também foi implementado o algoritmo de posicionamento do APoS, que usa a modificação do algoritmo de localização de comunidades de Girvan-Newman e também usa as heurísticas do algoritmo de *bin packing*.

Para complementar o simulador, foi desenvolvido um gerador de tráfego baseados no trabalho de Benson *et al.* [27, 28], onde cada valor para a matriz de tráfego é gerado independentemente a partir de um número aleatório, gerado através de uma distribuição log-normal. Dessa forma, poderíamos simular diversos tipos de tráfego de dados, somente modificando os dados gerados aleatoriamente.

As simulações foram executadas com diversas modificações de parâmetros, desde data centers em que a variação entre o tráfego das máquinas virtuais é intensa até em data centers onde o tráfego é disperso. O número de máquinas virtuais foi variado até mais de 16 mil máquinas, alocadas a mais de três mil servidores, dispersos em diferentes números de partições. Os dados de CPU e memória utilizados por cada máquina virtual foram gerados aleatoriamente a partir de uma distribuição normal.

As simulações mostraram que o algoritmo de posicionamento de serviços proposto é capaz de posicionar máquinas virtuais de forma mais eficiente quanto ao tráfego no núcleo. Quando comparado ao posicionamento que não leva em conta informações do tráfego, somente dados de requisitos de CPU e memória. Nos melhores casos simulados existe uma diminuição de 80% do tráfego no núcleo, enquanto nos piores casos simulados ainda existe uma melhora de 12,5%. Em todos os casos apresentados o tráfego no núcleo usando o APoS superou o algoritmo que não considera o tráfego no posicionamento das máquinas virtuais.

O algoritmo proposto tem a capacidade de melhorar a banda de tráfego disponível, especialmente a banda de dados disponível para as máquinas virtuais no núcleo da rede. Ao melhorar a distribuição do tráfego de dados, movendo o tráfego do núcleo da rede para as bordas da rede, a banda livre resultante pode ser melhor aproveitada para enviar os dados de dentro do data center para a Internet, que necessariamente devem passar pelo núcleo da rede, já que os roteadores normalmente estão ligados nessa camada.

7.1 Trabalhos futuros

A partir da análise do comportamento do APoS podemos definir rumos a serem seguidos para continuar o aprimoramento desta dissertação.

Um primeiro trabalho futuro deve analisar as melhores formas de migrar

múltiplas máquinas virtuais ao mesmo tempo e com pouca interferência na rede do data center, criando métodos eficientes para as migrações. Esses métodos devem observar que uma quantidade elevada de migrações simultâneas pode sobrecarregar a rede. Uma sobrecarga nem sempre pode ser desejável naquele momento ou as migrações, mesmo melhorando o tráfego momentaneamente, podem acabar gerando a longo prazo um efeito indesejado, como a necessidade de outras migrações de máquinas virtuais relacionadas ou futuramente relacionadas àquela máquina.

Outro aspecto que deve ser analisado antes da migração de uma máquina virtual é se essa migração não irá interferir de forma prejudicial no cliente da máquina virtual, já que uma migração sobrecarrega os recursos de processamento e de rede enquanto está sendo executada e muitas vezes existem acordos para a qualidade do serviço prestado. Uma máquina também não deve ser migrada seguidamente, atrapalhando o funcionamento do algoritmo e do data center como um todo.

É possível também explorar a economia de energia no data center. Como mostrado nos resultados, dependendo do algoritmo de *bin packing* utilizado é possível que uma porção do data center seja desligada para economizar energia ou colocada em modo de economia de energia, para conseguir responder rapidamente a alguma variação de demanda mas gastando mais energia.

Análises mais profundas quanto ao tráfego gerado devem ser feitas. De forma ideal é desejável a análise de dados reais de um data center operacional, com um número elevado de máquinas virtuais. Desse modo seria possível, além de analisar o algoritmo com dados reais, validar com dados reais o gerador de dados proposto nesta dissertação.

Com dados reais de diversos data centers é possível ajustar o APoS para ter um melhor desempenho, observando se há possibilidade de melhoria no algoritmo de encontrar comunidades ou há possibilidade de melhoria na técnica utilizada para empacotar as comunidades. Dados reais de topologia podem ser utilizados para melhorar a divisão em partições de servidores proposta nesse trabalho.

Um último passo para a implementação do algoritmo como solução é a criação de um protótipo capaz de obter os dados das máquinas virtuais, escolher e executar de forma autônoma as migrações necessárias para encontrar o melhor posicionamento das máquinas virtuais baseado no tráfego de dados entre as máquinas virtuais.

Referências Bibliográficas

[1] ZHANG, Q., CHENG, L., BOUTABA, R. "Cloud computing: state-of-the-art and research challenges", *Journal of Internet Services and Applications*, v. 1, pp. 7–18, 2010. Disponível em: <http://dx.doi.org/10.1007/s13174-010-0007-6>.

[2] RIMAL, B., CHOI, E., LUMB, I. "A taxonomy and survey of cloud computing systems". In: *INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on*, pp. 44–51. IEEE, 2009.

[3] BALIGA, J., AYRE, R., HINTON, K., et al. "Green cloud computing: Balancing energy in processing, storage, and transport", *Proceedings of the IEEE*, v. 99, n. 1, pp. 149–167, 2011.

[4] ARMBRUST, M., FOX, A., GRIFFITH, R., et al. "A view of cloud computing", *Communications of the ACM*, v. 53, n. 4, pp. 50–58, 2010.

[5] ZHANG, Q., CHENG, L., BOUTABA, R. "Cloud computing: state-of-the-art and research challenges", *Journal of Internet Services and Applications*, v. 1, n. 1, pp. 7–18, 2010.

[6] BARI, M., BOUTABA, R., ESTEVES, R., et al. "Data Center Network Virtualization: A Survey", *Communications Surveys Tutorials, IEEE*, , n. 99, pp. 1–20, set. 2012.

[7] GROUP, M. M. "Internet World Stats Usage and Population Statistics". Website, 2013. Disponível em: <http://www.internetworldstats.com/stats.htm>.

[8] GOOGLE. "Your web, everywhere". Blog, 2012. Disponível em: <http://googleblog.blogspot.se/2012/06/chrome-apps-google-io-your-web.html>.

[9] GOOGLE. "Zeitgeist 2012". Website, 2012. Disponível em: <http://www.google.com/zeitgeist/2012/#the-world>.

[10] LUO, L. "China data center roundup: Big cloud news from big cloud players". DatacenterDynamics, 2012. Disponível em: <http://www.datacenterdynamics.com/focus/archive/2012/08/china-data-center-roundup-big-cloud-news-big-cloud-players>.

[11] MISHRA, M., DAS, A., KULKARNI, P., et al. "Dynamic resource management using virtual machine migrations", *Communications Magazine, IEEE*, v. 50, n. 9, pp. 34 –40, set. 2012.

[12] BODÍK, P., MENACHE, I., CHOWDHURY, M., et al. "Surviving failures in bandwidth-constrained datacenters", *SIGCOMM Comput. Commun. Rev.*, v. 42, n. 4, pp. 431–442, 2012.

[13] BARHAM, P., DRAGOVIC, B., FRASER, K., et al. "Xen and the art of virtualization". In: *ACM SIGOPS Operating Systems Review*, v. 37, pp. 164–177. ACM, 2003.

[14] KOOMEY, J. G. "Worldwide electricity used in data centers", *Environmental Research Letters*, v. 3, n. 3, pp. 034008, jul. 2008.

[15] GREENBERG, A., HAMILTON, J., MALTZ, D. A., et al. "The cost of a cloud: research problems in data center networks", *SIGCOMM Comput. Commun. Rev.*, v. 39, n. 1, pp. 68–73, 2008.

[16] CLARK, C., FRASER, K., HAND, S., et al. "Live migration of virtual machines". In: *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2*, NSDI'05, pp. 273–286, 2005.

[17] PISA, P. S., FERNANDES, N. C., CARVALHO, H. E. T., et al. "OpenFlow and Xen-Based Virtual Network Migration", *Communications: Wireless in Developing Countries and Networks of the Future*, v. 327, pp. 170–181, 2010.

[18] PIAO, J. T., YAN, J. "A Network-aware Virtual Machine Placement and Migration Approach in Cloud Computing". In: *Grid and Cooperative Computing (GCC), 2010 9th International Conference on*, 2010.

[19] WANG, G., NG, T. "The Impact of Virtualization on Network Performance of Amazon EC2 Data Center". In: *INFOCOM, 2010 Proceedings IEEE*, pp. 1 –9, 2010.

[20] *Cisco data center infrastructure 2.5*. Relatório técnico, . Disponível em: <http://www.cisco.com/en/US/docs/solutions/Enterprise/Data_Center/DC_Infra2_5/DCI_SRNDa.pdf>.

[21] AL-FARES, M., LOUKISSAS, A., VAHDAT, A. "A scalable, commodity data center network architecture". In: *Proceedings of the ACM SIGCOMM 2008*, pp. 63–74. ACM, 2008.

[22] GREENBERG, A., HAMILTON, J., JAIN, N. "VL2: a scalable and flexible data center network". In: *Proceedings of the ACM SIGCOMM 2009*, pp. 51–62. ACM, 2009.

[23] NIRANJAN MYSORE, R., PAMBORIS, A., FARRINGTON, N., et al. "PortLand: a scalable fault-tolerant layer 2 data center network fabric". In: *Proceedings of the ACM SIGCOMM 2009*, v. 39, pp. 39–50. ACM, 2009.

[24] GUO, C., WU, H., TAN, K., et al. "Dcell: a scalable and fault-tolerant network structure for data centers", *Proceedings of the ACM SIGCOMM 2008*, pp. 75–86, 2008.

[25] GUO, C., LU, G., LI, D., et al. "BCube: a high performance, server-centric network architecture for modular data centers", *Proceedings of the ACM SIGCOMM 2009*, pp. 63–74, 2009.

[26] KANDULA, S., SENGUPTA, S., GREENBERG, A., et al. "The nature of data center traffic: measurements & analysis". In: *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, IMC '09, pp. 202–208, 2009.

[27] BENSON, T., AKELLA, A., MALTZ, D. A. "Network traffic characteristics of data centers in the wild". In: *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, IMC '10, pp. 267–280, 2010.

[28] BENSON, T., ANAND, A., AKELLA, A., et al. "Understanding data center traffic characteristics", *SIGCOMM Comput. Commun. Rev.*, v. 40, n. 1, pp. 92–99, jan. 2010.

[29] VERMA, A., AHUJA, P., NEOGI, A. "pMapper: power and migration cost aware application placement in virtualized systems". In: *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, Middleware '08, 2008.

[30] MEHTA, S., NEOGI, A. "ReCon: A tool to Recommend dynamic server Consolidation in multi-cluster data centers". In: *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*, 2008.

[31] STAGE, A., SETZER, T. "Network-aware migration control and scheduling of differentiated virtual machine workloads". In: *Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, pp. 9–14, 2009.

[32] WOOD, T., SHENOY, P., VENKATARAMANI, A., et al. "Sandpiper: Black-box and gray-box resource management for virtual machines", *Computer Networks*, v. 53, n. 17, pp. 2923 – 2938, 2009.

[33] SONNEK, J., CHANDRA, A. "Virtual putty: Reshaping the physical footprint of virtual machines". In: *Proc. of Workshop on Hot Topics in Cloud Computing (HotCloud'09)*, 2009.

[34] MENG, X., PAPPAS, V., ZHANG, L. "Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement". In: *INFOCOM, 2010 Proceedings IEEE*, pp. 1 –9, mar. 2010.

[35] FERRETO, T. C., NETTO, M. A., CALHEIROS, R. N., et al. "Server consolidation with migration control for virtualized data centers", *Future Generation Computer Systems*, 2011.

[36] WANG, M., MENG, X., ZHANG, L. "Consolidating virtual machines with dynamic bandwidth demand in data centers". In: *INFOCOM, 2011 Proceedings IEEE*, pp. 71 –75, abr. 2011.

[37] ALICHERRY, M., LAKSHMAN, T. "Network aware resource allocation in distributed clouds". In: *INFOCOM, 2012 Proceedings IEEE*, pp. 963–971. IEEE, 2012.

[38] BIRAN, O., CORRADI, A., FANELLI, M., et al. "A Stable Network-Aware VM Placement for Cloud Systems". In: *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, pp. 498 –506, mai. 2012.

[39] JIANG, J., LAN, T., HA, S., et al. "Joint VM placement and routing for data center traffic engineering". In: *INFOCOM, 2012 Proceedings IEEE*, pp. 2876–2880. IEEE, 2012.

[40] VERDI, F. L., ROTHENBERG, C. E., PASQUINI, R., et al. "Novas Arquiteturas de Data Center para Cloud Computing". In: *SBRC 2010 - Minicursos*, cap. 3, 2010.

[41] GROUP, T. W. B. "Internet users". Website, 2012. Disponível em: <http://data.worldbank.org/indicator/IT.NET.USER>.

[42] DIAS, D. S., COSTA, L. H. M. K. "Online Traffic-aware Virtual Machine Placement in Data Center Networks". In: *Global Information Infrastructure and Networking Symposium 2012 (GIIS'12)*, Choroni, Venezuela, dez. 2012.

[43] LEISERSON, C. E. "Fat-trees: universal networks for hardware-efficient supercomputing", *IEEE Trans. Comput.*, v. 34, n. 10, pp. 892–901, 1985. Disponível em: <http://dl.acm.org/citation.cfm?id=4492.4495>.

[44] POPA, L., RATNASAMY, S., IANNACCONE, G., et al. "A cost comparison of datacenter network architectures". In: *Proceedings of the 6th International COnference*, Co-NEXT '10, pp. 16:1–16:12. ACM, 2010.

[45] COUTO, R. S., CAMPISTA, M. E. M., COSTA, L. H. M. "A Reliability Analysis of Datacenter Topologies", *Global Telecommunications Conference (GLOBECOM 2012)*, dez. 2012.

[46] ZHAO, W., WANG, Z., LUO, Y. "Dynamic memory balancing for virtual machines", *SIGOPS Oper. Syst. Rev.*, v. 43, n. 3, pp. 37–47, jul. 2009.

[47] MORAES, I. M., PISA, P. S., CARVALHO, H. E., et al. "Vnext: Uma ferramenta de controle e gerenciamento para redes virtuais baseadas em xen", *Salão de Ferramentas do XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos*, pp. 981–988, 2011.

[48] CARVALHO, H. E., DUARTE, O. C. "Elastic Allocation and Automatic Migration Scheme for Virtual Machines", *Journal of Emerging Technologies in Web Intelligence - JETWI*, v. 4, n. 4, 2012.

[49] FOSTER, I., ZHAO, Y., RAICU, I., et al. "Cloud computing and grid computing 360-degree compared". In: *Grid Computing Environments Workshop, 2008. GCE'08*, pp. 1–10. IEEE, 2008.

[50] IOSUP, A., OSTERMANN, S., YIGITBASI, M., et al. "Performance analysis of cloud computing services for many-tasks scientific computing", *Parallel and Distributed Systems, IEEE Transactions on*, v. 22, n. 6, pp. 931–945, 2011.

[51] COSTA, L. H. M. K., AMORIM, M. D., CAMPISTA, M. E. M., et al. "Grandes Massas de Dados na Nuvem: Desafios e Técnicas para Inovação", *Minicursos do Simpósio Brasileiro de Redes de Computadores - SBRC'2012*, p. 58, 2012.

[52] APACHE. "The Hadoop Project". Website, 2013. Disponível em: <http://hadoop.apache.org/>.

[53] GOLDMAN, A., KON, F., JUNIOR, F. P., et al. "Apache Hadoop: Conceitos teóricos e práticos, evolução e novas possibilidades", *XXXI Jornadas de atualizações em informatica*, 2012.

[54] DEAN, J., GHEMAWAT, S. "MapReduce: simplified data processing on large clusters", *Communications of the ACM*, v. 51, n. 1, pp. 107–113, 2008.

[55] PINTO, R. J. *Aplicação de Processamento Paralelo ao Problema de Planejamento da Operação de Sistemas Hidrotérmicos Baseado em Cluster de Computadores*. Tese de Doutorado, UFRJ, set. 2011.

[56] BAKER, M., APON, A., BUYYA, R., et al. "Cluster computing and applications", *Encyclopedia of Computer Science and Technology*, v. 45, n. Supplement 30, pp. 87–125, 2002.

[57] CERN. "Worldwide LHC Computing Grid". CERN Website, 2013. Disponível em: <http://public.web.cern.ch/public/en/lhc/Computing-en.html>.

[58] O'LUANAIGH, C. "CERN Data Centre passes 100 petabytes". CERN Website, 2013. Disponível em: <http://home.web.cern.ch/about/updates/2013/02/cern-data-centre-passes-100-petabytes>.

[59] STATEN, J. "Is Cloud Computing Ready For The Enterprise?" *Forrester Research*, v. 7, mar. 2008.

[60] PAPADOPOULOS, P. M. "Extending clusters to Amazon EC2 using the Rocks toolkit", *Int. J. High Perform. Comput. Appl.*, v. 25, n. 3, pp. 317–327, 2011.

[61] MURUGESAN, S. "Understanding Web 2.0", *IT Professional*, 2007. ISSN: 1520-9202. doi: 10.1109/MITP.2007.78.

[62] INCORPORATED, P. A. R. C. "OVERVIEW: content-centric networking". Website, 2013. Disponível em: <http://www.parc.com/services/focus-area/content-centric-networking/>.

[63] JACOBSON, V., BRAYNARD, R. L., DIEBERT, T., et al. "Custodian-based information sharing", *Communications Magazine, IEEE*, v. 50, n. 7, pp. 38–43, 2012.

[64] PORTER, M., ONNELA, J., MUCHA, P. "Communities in networks", *Notices of the American Mathematical Society*, v. 56, n. 9, pp. 1082–1097, 2009.

[65] BLONDEL, V., GUILLAUME, J., LAMBIOTTE, R., et al. "Fast unfolding of communities in large networks", *Journal of Statistical Mechanics: Theory and Experiment*, v. 2008, n. 10, pp. P10008, 2008.

[66] CLAUSET, A., NEWMAN, M., MOORE, C. "Finding community structure in very large networks", *Physical review E*, v. 70, n. 6, 2004.

[67] GIRVAN, M., NEWMAN, M. "Community structure in social and biological networks", *Proceedings of the National Academy of Sciences*, v. 99, n. 12, pp. 7821–7826, 2002.

[68] NEWMAN, M. "Detecting community structure in networks", *The European Physical Journal B-Condensed Matter and Complex Systems*, v. 38, n. 2, pp. 321–330, 2004.

[69] NEWMAN, M., GIRVAN, M. "Finding and evaluating community structure in networks", *Physical review E*, v. 69, n. 2, 2004.

[70] JAIN, A. K., MURTY, M. N., FLYNN, P. J. "Data clustering: a review", *ACM computing surveys (CSUR)*, v. 31, n. 3, pp. 264–323, 1999.

[71] JOHNSON, D. "Fast algorithms for bin packing", *Journal of Computer and System Sciences*, v. 8, n. 3, pp. 272–314, 1974.

[72] MAN JR, E., GAREY, M., JOHNSON, D. "Approximation algorithms for bin packing: A survey". In: *Approximation Algorithms for NP-Hard Problems*, pp. 46–93, 1996.

[73] KANG, J., PARK, S. "Algorithms for the variable sized bin packing problem", *European Journal of Operational Research*, v. 147, n. 2, pp. 365–372, 2003.

[74] BOYAR, J., EPSTEIN, L., FAVRHOLDT, L., et al. "The maximum resource bin packing problem", *Theoretical Computer Science*, v. 362, n. 1, pp. 127–139, 2006.

[75] PANDEY, S., CHOI, M., LEE, S., et al. "IP network topology discovery using SNMP". In: *Information Networking, 2009. ICOIN 2009. International Conference on*, pp. 1–5. IEEE, 2009.

[76] HIMURA, Y., YASUDA, Y. "Discovering configuration templates of virtualized tenant networks in multi-tenancy datacenters via graph-mining", *ACM SIGCOMM Computer Communication Review*, v. 42, n. 3, pp. 13–20, 2012.

[77] *Topology Discovery Methods*. Relatório técnico, . Disponível em: <http://www.cisco.com/en/US/docs/net_mgmt/cisco_network_connectivity_monitor/1.0/user/guide/411_config_and_admin_methods.pdf>.

[78] WILKES, J. "More Google cluster data". Google research blog, nov. 2011. Disponível em: <http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html>.

[79] MISHRA, A., HELLERSTEIN, J., CIRNE, W., et al. "Towards characterizing cloud backend workloads: insights from google compute clusters", *ACM SIGMETRICS Performance Evaluation Review*, v. 37, n. 4, pp. 34–41, 2010.

[80] SHARMA, B., CHUDNOVSKY, V., HELLERSTEIN, J. L., et al. "Modeling and synthesizing task placement constraints in Google compute clusters". In: *Proceedings of the 2nd ACM Symposium on Cloud Computing*, SOCC˜'11, pp. 3:1–3:14. ACM, 2011.

[81] KAVULYA, S., TAN, J., GANDHI, R., et al. "An analysis of traces from a production mapreduce cluster". In: *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, pp. 94–103. IEEE, 2010.

[82] HENDERSON, T., LACAGE, M., RILEY, G., et al. "Network simulations with the ns-3 simulator", *SIGCOMM demonstration*, 2008.

[83] CALHEIROS, R., RANJAN, R., BELOGLAZOV, A., et al. "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", *Software: Practice and Experience*, v. 41, n. 1, pp. 23–50, 2011.

[84] KLIAZOVICH, D., BOUVRY, P., AUDZEVICH, Y., et al. "GreenCloud: A Packet-Level Simulator of Energy-Aware Cloud Computing Data Centers". In: *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pp. 1 –5, dez. 2010.

[85] BUYYA, R., RANJAN, R., CALHEIROS, R. "Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities". In: *High Performance Computing Simulation, 2009. HPCS '09. International Conference on*, pp. 1 –11, jun. 2009.