PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

MAPEAMENTO DE MÁQUINAS VIRTUAIS EM DATACENTERS PRIVADOS VISANDO MINIMIZAR A INTERFERÊNCIA DE DESEMPENHO

LUIS CARLOS JERSAK

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciência da Computação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. Dr. Tiago Coelho Ferreto

Porto Alegre
2014

Dados Internacionais de Catalogação na Publicação (CIP)

J56m Jersak, Luis Carlos

Mapeamento de máquinas virtuais em datacenters privados visando minimizar a interferência de desempenho / Luis Carlos Jersak. – Porto Alegre, 2014.
65 p.

Diss. (Mestrado) – Fac. de Informática, PUCRS.
Orientador: Prof. Dr. Tiago Coelho Ferreto.

1. Informática. 2. Máquinas Virtuais. 3. Redes de Computadores. I. Ferreto, Tiago Coelho. II. Título.

CDD 004.65

Ficha Catalográfica elaborada pelo
Setor de Tratamento da Informação da BC-PUCRS

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada "Mapeamento de Máquinas Virtuais em Datacenters Visando Minimizar a Interferência de Desempenho" apresentada por Luis Carlos Jersak como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, aprovada em 26/02/2014 pela Comissão Examinadora:

Prof. Dr. Tiago Coelho Ferreto – Orientador — PPGCC/PUCRS

Prof. Dr. César Augusto Fonticielha de Rose – PPGCC/PUCRS

Prof. Dr. Rodrigo da Rosa Righi – UNISINOS

Homologada em...../...../....., conforme Ata No. ..... pela Comissão Coordenadora.

Prof. Dr. Luiz Gustavo Leão Fernandes
Coordenador.

PUCRS
Campus Central
Av. Ipiranga, 6681 – P32– sala 507 – CEP: 90619-900
Fone: (51) 3320-3611 – Fax (51) 3320-3621
E-mail: ppgcc@pucrs.br
www.pucrs.br/facin/pos

DEDICATÓRIA

Dedico este trabalho a meus pais, exemplos de caráter e perseverança, e à Débora, por seu apoio incondicional.

"...and that, my liege, is how we know the Earth to be banana shaped."
(Sir Bedevere)

AGRADECIMENTOS

Agradeço a todos que direta ou indiretamente contribuíram para a realização deste trabalho. Em especial ao orientador, Prof. Tiago Ferreto, pelos conselhos que foram muito valiosos. Aos amigos do LAD e do CEPES pela ajuda em diversos temas e pelos momentos de descontração. Agradeço ainda à minha família e à minha companheira Débora pelo amor e por confiarem em mim neste desafio.

MAPEAMENTO DE MÁQUINAS VIRTUAIS EM DATACENTERS PRIVADOS VISANDO MINIMIZAR A INTERFERÊNCIA DE DESEMPENHO

RESUMO

O poder dos computadores aumenta ano após ano e atualmente é comum que as pessoas tenham em suas casas computadores pessoais com capacidade computacional similar a de servidores e mainframes de anos atrás. Naturalmente, os servidores atuais acompanharam este desenvolvimento. No entanto, muitas aplicações que são executadas nestes servidores já não necessitam de todo o poder uma máquina exclusiva e isso levou ao surgimento de soluções para evitar que os recursos computacionais de um servidor sejam desperdiçados.

Uma abordagem bastante difundida é a consolidação de servidores. Através do uso de virtualização é possível compartilhar os recursos de um servidor entre diversas máquinas virtuais, reduzindo o desperdício e aumentando a quantidade de clientes que podem ser atendidos com um único servidor. No entanto, diversos estudos [19, 24, 38] mostram que máquinas virtuais podem interferir no desempenho de outras ao existirem disputas pelo mesmo recurso computacional.

Desta forma, a proposta deste trabalho é desenvolver um algoritmo de mapeamento de máquinas virtuais que minimize o número de servidores necessários ao mesmo tempo em que mantém a interferência de desempenho abaixo de um limiar a ser especificado pelo usuário.

Os resultados obtidos com a avaliação da solução proposta mostram que a mesma consegue realizar o mapeamento de máquinas virtuais sem ultrapassar o limiar estabelecido pelo usuário, bem como reduzir a interferência significativamente sem aumento expressivo na quantidade de servidores necessários e desta forma atingindo o objetivo geral do trabalho.

Palavras Chave: virtualização, consolidação, mapeamento, interferência.

MAPPING OF VIRTUAL MACHINES IN PRIVATE DATACENTERS AIMING TO REDUCE PERFORMANCE INTERFERENCE

ABSTRACT

The power of computers increases year after year and today is common to have at home personal computers with computational power similar to servers and mainframes of years ago. Naturally, todays' servers followed this evolution. However, many applications that run in these servers no longer require the computational power of a single, exclusive, server and this led to the development of solutions to avoid wasting servers' resources.

A common approach is server consolidation. Through virtualization it is possible to share resources from a single server among multiple virtual machines, reducing the waste of resources and increasing the amount of customers that can be served with a single server. However, several studies [19, 24, 38] show that virtual machines can interfer in the performance of other virtual machines when there are disputes over the same resources.

This work proposes an algorithm for mapping virtual machines that minimize the number of servers required while maintaining the performance interference below a threshold specified by the user.

The results obtained after evaluating the proposed solution show that it can map virtual machines without exceeding the threshold set by the user, as well as significantly reduce the interference without an expressive increase in the number of required servers.

Keywords: virtualization, consolidation, mapping, interference.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

O poder dos computadores vem aumentando ano após ano e atualmente já é comum que as pessoas tenham em suas casas computadores pessoais com capacidade computacional similar ao de servidores e *mainframes* de anos atrás. Naturalmente, o desempenho dos servidores atuais também acompanhou esse crescimento. Atualmente, não é raro que aplicações que anteriormente eram executadas em um servidor exclusivo, como por exemplo um servidor web de pequeno porte, já não necessitem de todo o poder computacional de uma máquina de última geração e para evitar o desperdício de recursos computacionais novas alternativas precisaram ser desenvolvidas.

Uma abordagem utilizada para contornar este problema é a consolidação de servidores. Através do uso de virtualização é possível alocar diversas máquinas virtuais (do inglês *virtual machines* ou VMs) em um único servidor físico, aproveitando melhor os recursos computacionais, reduzindo o desperdício e consequentemente os custos. Outra vantagem que pode ser citada é a economia de energia, já que é possível utilizar um menor número de servidores para atender a mesma demanda. No entanto, sabe-se que o isolamento entre as VMs ainda não é perfeito e é possível que ocorram disputas por recursos computacionais entre as VMs, o que pode degradar o desempenho do sistema em geral, conforme é constatado em diversos estudos [19, 24, 38].

A proposta deste trabalho é desenvolver um modelo de interferência e então aplicá-lo a um algoritmo de mapeamento de máquinas virtuais que minimize o número de servidores necessários, ao mesmo tempo em que mantém a interferência de desempenho abaixo de um limiar a ser especificado pelo usuário. Procurou-se atingir este objetivo em três etapas: primeiramente analisaram-se os níveis de interferência produzidos pelo compartilhamento de recursos de um servidor com a finalidade de criar um modelo de interferência. O modelo de interferência, desenvolvido na segunda etapa, procura prever os níveis de interferência que serão gerados pelas diferentes combinações de máquinas virtuais. A previsão gerada pelo modelo de interferência é utilizada pelo algoritmo de mapeamento desenvolvido na terceira etapa. Este algoritmo de mapeamento tem como base heurísticas utilizadas na solução de problemas de *bin packing* devido à similaridade que este problema tem com o problema de consolidação de servidores.

O foco deste trabalho são *datacenters* privados, onde se tem conhecimento prévio dos tipos de aplicações e cargas de trabalho executados por seus usuários. Utiliza-se esse conhecimento para que não seja necessário realizar monitoramento *online* das máquinas virtuais, desta forma simplificando-se o mapeamento.

Os resultados obtidos com a avaliação da solução proposta mostram que a mesma consegue realizar o mapeamento de máquinas virtuais sem ultrapassar o limiar estabelecido pelo usuário. Em diversos casos, a solução proposta reduz a interferência de desempenho sem aumentar o número de servidores necessários, enquanto em outros casos foi possível obter uma redução maior nos níveis de interferência com um aumento baixo no número de servidores necessários.

1.1 Objetivos

O objetivo geral deste trabalho é desenvolver um algoritmo de mapeamento de máquinas virtuais em *datacenters* privados que mantenha a interferência de desempenho entre as máquinas virtuais abaixo de um limiar pré-estabelecido pelo usuário do algoritmo. No escopo deste trabalho, considera-se que o "usuário"seja uma pessoa responsável pela administração ou gerência de um *datacenter* privado. Os objetivos específicos são os seguintes:

- Estudar o estado da arte relacionado aos algoritmos e técnicas de mapeamento e consolidação de servidores.
- Estudar as tecnologias de virtualização existentes atualmente para melhor entender as características e o funcionamento das mesmas.
- Realizar a análise de interferência entre os principais recursos computacionais (CPU, Memória RAM e E/S de disco) a fim de gerar um modelo de predição de interferência.
- Gerar o modelo que será utilizado pelo algoritmo para prever o nível de interferência entre as máquinas virtuais e realizar o mapeamento de acordo.

A proposta de solução é baseada nas seguintes premissas:

- Tem-se conhecimento prévio das características de cada máquina virtual que será mapeada, como por exemplo o tamanho de cada VM, as suas demandas por recursos e se fazem uso intensivo de algum recurso em específico, já que o algoritmo a ser desenvolvido trabalhará de forma *offline*.
- Os recursos de hardware são homogêneos. Isto é necessário pois servidores com configurações de hardware diferentes podem produzir diferentes níveis de interferência para a mesma combinação de VMs nele alocadas.
- Que o usuário defina um limiar de interferência que será passado como entrada para o algoritmo.

1.2 Hipóteses de pesquisa

Este trabalho procura verificar as seguintes hipóteses de pesquisa:

1. É possível desenvolver um modelo de interferência que preveja a interferência gerada pela combinação de máquinas virtuais alocadas em um mesmo servidor físico?
2. Caso seja possível desenvolver tal modelo de interferência, é possível aplicar este modelo a um algoritmo de mapeamento para realizar a consolidação de servidores?

1.3 Organização do trabalho

O restante deste trabalho está organizado da seguinte forma: o Capítulo 2 apresenta a fundamentação teórica deste trabalho, discorrendo sobre virtualização e técnicas envolvidas, bem como sobre consolidação de servidores, seus benefícios e desafios e, finalmente, sobre o problema de interferência de desempenho existente em ambientes virtualizados. O Capítulo 3 apresenta o estado da arte na área de algoritmos de mapeamento de máquinas virtuais. Os detalhes do desenvolvimento do modelo de interferência e do algoritmo de mapeamento são apresentados no Capítulo 4 e o Capítulo 5 apresenta a avaliação do algoritmo proposto. O Capítulo 6 apresenta as considerações finais e trabalhos futuros.

2. FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta o embasamento teórico utilizado para o desenvolvimento deste trabalho. A Seção 2.1 apresenta os conceitos de Virtualização de sistemas computacionais bem como os modelos existentes e suas características. Na Seção 2.2 são apresentados os conceitos de consolidação de servidores, suas vantagens e desafios. A Seção 2.3 discorre sobre o problema de interferência de desempenho apresentando trabalhos relacionados a este tema que motivam a realização deste trabalho.

2.1 Virtualização

Computadores sempre tiveram restrições atribuídas à arquitetura do sistema sendo utilizado. Incompatibilidades entre sistemas de arquiteturas diferentes frequentemente impediam que uma aplicação desenvolvida para uma arquitetura fosse executada em outra, e a solução para tal problema era refazer a aplicação atendendo aos requisitos do novo sistema alvo.

Tais restrições aconteciam em diversos contextos de sistemas computacionais, desde compiladores e sistemas operacionais à arquitetura de hardware em si, como, por exemplo, diferentes conjuntos de instruções presentes nos processadores (eg. Intel IA-32 vs IBM PowerPC) [30]. A virtualização de sistemas computacionais surgiu como uma opção para a solução destes problemas de portabilidade. Através de uma camada de software que abstrai as características do hardware físico é possível simular uma arquitetura diferente da real, facilitando a portabilidade dos sistemas.

Quando um sistema (ou subsistema, como processador ou memória) é virtualizado, as suas interfaces e os recursos visíveis através destas interfaces são mapeados em uma nova interface de um sistema real. Consequentemente, o sistema real é transformado de forma que o mesmo aparenta ser um sistema diferente, virtual, ou até mesmo um conjunto de diversos sistemas virtuais [30].

Virtualização é uma técnica antiga, com algumas de suas primeiras demonstrações apresentadas pela IBM ainda na década de 60 [9]. Na década de 70, o sucesso dos *mainframes*, que eram máquinas muito caras e poderosas na época, fez com que a virtualização se tornasse interessante. Pelo fato de os *mainframes* frequentemente terem sistema operacional próprio, muitas aplicações tinham problemas de portabilidade, não sendo compatíveis com todos os tipos de máquinas presentes no mercado. Porém, através da virtualização era possível executar softwares legados ou de plataformas diferentes nestes *mainframes*. Por exemplo, juntamente com sua linha de *mainframes* 370 e sucessores, a IBM oferecia uma máquina virtual portada para várias de suas plataformas sobre a qual as aplicações executavam. Dessa forma era possível migrar uma aplicação de uma plataforma para outra desde que houvesse uma máquina virtual compatível com a plataforma alvo [4].

No entanto, com a popularização dos computadores domésticos (de arquitetura x86) a quantidade de sistemas operacionais reduziu-se para alguns poucos representantes (basicamente Windows, Linux e MacOS) e seus respectivos conjuntos de aplicativos. Devido a isto, a virtualização

perdeu, por um tempo, a sua importância já que o problema de portabilidade já não era mais tão grave. Porém, o aumento do poder computacional dos processadores e a popularização dos sistemas distribuídos fez com que o interesse em virtualização ressurgisse [4]. A diferença é que desta vez, ao invés da portabilidade, o foco está voltado ao melhor aproveitamento dos recursos computacionais, que resultam em menores gastos com hardware, energia, infraestrutura e equipe.

Para contribuir ainda mais, novas tecnologias para auxílio à virtualização foram desenvolvidas recentemente, melhorando o desempenho geral de sistemas virtualizados. Algumas delas, como as tecnologias Intel VT-x [35] e AMD-V [10], foram integradas diretamente aos processadores na forma de conjuntos de instruções que aprimoram o suporte à virtualização de sistemas. Além disso, o aperfeiçoamento dos softwares envolvidos também contribuiu para o aumento do desempenho de sistemas virtualizados. Atualmente, diversos estudos [19, 24, 38] mostram que o desempenho de uma máquina virtual é muito próximo ao de uma máquina real. Isto aliado ao fato de que computadores modernos têm poder computacional suficiente para, através de virtualização, suportarem diversas máquinas virtuais menores, contribuiu para o ressurgimento do interesse em tecnologias de virtualização [1].

Através do uso de virtualização é possível aproveitar com maior eficiência os recursos de uma máquina física. Onde antigamente era necessário provisionar uma máquina física inteira para uma aplicação, levando ao desperdício de recursos computacionais, atualmente é possível criar diversas máquinas virtuais menores dentro de uma única máquina física. Outro forte atrativo da virtualização é a economia de energia proporcionada pela consolidação das máquinas. Uma vez que é possível atender a um maior número de aplicações/clientes com uma única máquina física, outras podem ser desligadas, resultando na redução do consumo de energia geral de um *datacenter*. Além das vantagens já mencionadas, ainda pode-se citar como outras vantagens proporcionadas pela virtualização a facilidade de gerenciamento da infraestrutura e a segurança [31].

2.1.1 Tipos de Virtualização

Atualmente existem diversas técnicas para a virtualização de sistemas computacionais. Como as mais populares pode-se citar a virtualização total com tradução binária e a paravirtualização, que podem se beneficiar de virtualização assistida por hardware caso haja suporte. Além destas, também pode-se citar a virtualização em nível de sistema operacional, também chamada de *containers*. As duas primeiras técnicas utilizam um *hypervisor* (também chamado de *Virtual Machine Manager* - VMM), que é uma camada de software que atua entre a máquina hospedeira e as máquinas hóspedes, abstraindo o hardware da máquina física e apresentando uma plataforma virtualizada para as máquinas virtuais. Em contraste às duas primeiras técnicas, a virtualização baseada em containers não utiliza um *hypervisor*. Atualmente existem dois tipos principais de *hypervisors*: os "Nativos" (ou Tipo 1) e "Hospedados" (ou Tipo 2). No Tipo 1, o *hypervisor* é executado diretamente sobre o hardware, sem a necessidade de um sistema operacional subjacente. Os sistemas operacionais hóspedes são executados sobre o *hypervisor*, em uma segunda camada. Já

no Tipo 2, existe um sistema operacional convencional entre o hardware físico e o *hypervisor*, e os sistemas operacionais hóspedes são executados em uma terceira camada [4]. A Figura 2.1 mostra as diferenças entre *hypervisors* Tipo 1 e Tipo 2.

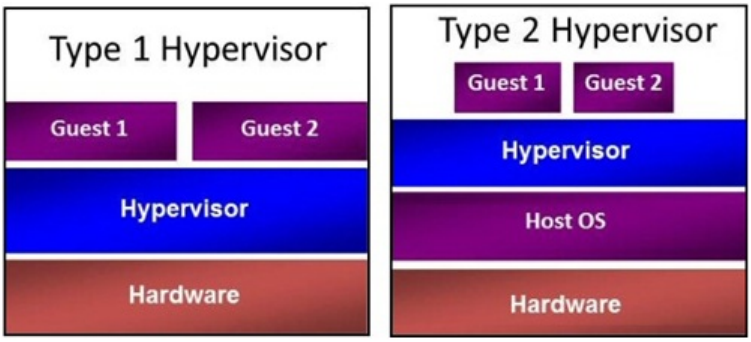

Figura 2.1 – Tipos de *Hypervisors*

No modelo de virtualização total, um sistema operacional sem modificações é executado sobre uma camada de software (*hypervisor*) que abstrai completamente o hardware do servidor físico. Como o sistema operacional não está ciente de que ele está sendo virtualizado, é necessário que o *hypervisor* capture as instruções do sistema operacional e então as traduza para que possam ser executadas no hardware hospedeiro. É importante notar que este processo de virtualização pode ser bastante oneroso do ponto de vista de desempenho já que o *hypervisor* deve emular todos os dispositivos da plataforma com detalhamento suficiente para que o sistema operacional possa manipulá-los em baixo nível. Além disso, o processo de capturar e traduzir as instruções do sistema operacional impõe uma sobrecarga no sistema que causa redução de desempenho. Exemplos de *hypervisors* que implementam este tipo de virtualização são o VMWare ESX/ESXi [36] e o KVM [16]. A Figura 2.2 representa a virtualização total.

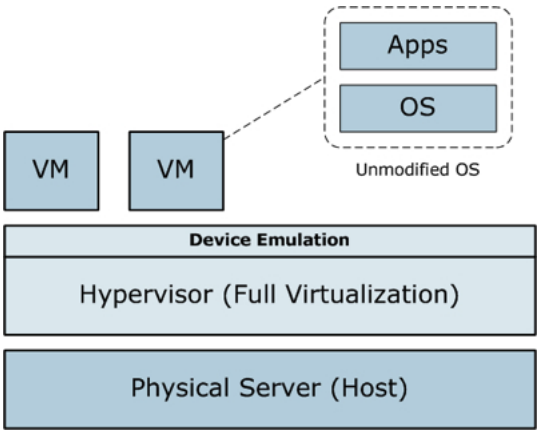

Figura 2.2 – Virtualização Total

A paravirtualização é uma abordagem alternativa que ajuda a contornar as desvantagens da virtualização total. Na paravirtualização o sistema operacional hóspede é modificado, permitindo que este se coordene com o *hypervisor*, reduzindo o uso de instruções que necessitam de tradução e que são tipicamente responsáveis pelas penalidades de desempenho. Na paravirtualização os dispositivos de hardware são acessados por *drivers* do próprio *hypervisor* [4], substituindo a emulação

de dispositivos pela cooperação entre os *drivers* do hóspede e do *hypervisor*. No entanto, a desvantagem da paravirtualização é a necessidade de se modificar o sistema operacional a ser virtualizado para que este se integre com o *hypervisor*. Um exemplo de *hypervisor* que suporta este tipo de virtualização é o Xen [39]. A Figura 2.3 representa a paravirtualização.

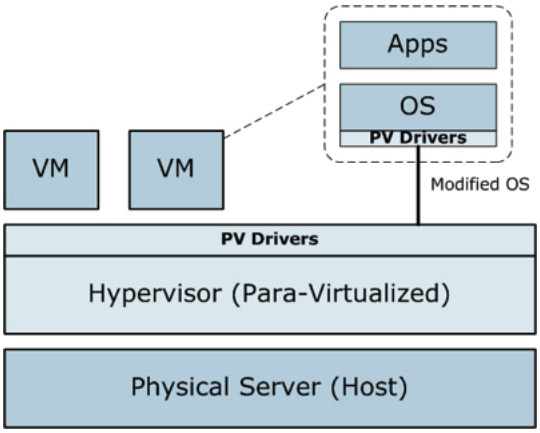

Figura 2.3 – Paravirtualização

Alguns modelos recentes de processadores passaram a integrar conjuntos especiais de instruções para dar suporte à virtualização diretamente em hardware, como as tecnologias Intel VT-x [35] e AMD-V [10]. Com o uso destas tecnologias, a captura e tradução das instruções de sistema operacional são feitas diretamente em hardware ao invés de serem feitas em software, aprimorando o desempenho. Além disso, essas tecnologias também permitem que em alguns casos as máquinas virtuais acessem o hardware diretamente, sem ter de passar pelo *hypervisor*, aumentando o desempenho em operações de entrada e saída.

Na virtualização em nível de sistema operacional, diferentemente dos modelos anteriores, não é utilizado um *hypervisor*. Ao invés disso, o sistema operacional é modificado para que diversas instâncias, ou *user-spaces*, possam ser executadas na mesma máquina física. Cada *user-space* compartilha o mesmo *kernel*, porém é executado isoladamente das outras instâncias, cada um deles tendo sua própria lista de processos, entre outros aspectos. Como neste modelo não existe a necessidade de tradução de instruções, o desempenho é próximo ao da execução nativa [5]. A principal desvantagem deste modelo é que se houver algum problema a nível de *kernel* do sistema, todas as máquinas virtuais (que neste modelo são frequentemente chamadas de *Virtual Private Servers* - *VPS*) hospedadas naquele servidor podem ser comprometidas. No entanto, o fato de se utilizar apenas um *kernel* ao invés de muitos, contribui para que menos recursos sejam necessários. Como alguns exemplos de sistemas para este tipo de virtualização pode-se citar Linux VServer [37], OpenVZ [23] e Linux Containers (LXC) [18]. A Figura 2.4 representa a virtualização em nível de sistema operacional.

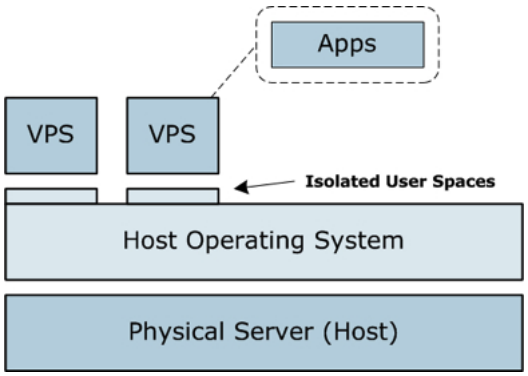

Figura 2.4 – Virtualização em nível de sistema operacional

2.2 Consolidação de Servidores

Das vantagens proporcionadas pela virtualização, talvez uma das mais atrativas seja a possibilidade de se alocar diversas máquinas virtuais em uma única máquina física.

Segundo Indrani et al. [25], consolidação de servidores é o processo de encapsular a carga de trabalho de um único servidor em uma máquina virtual e então executá-lo em uma plataforma de hardware compartilhada. Na maioria dos casos, esse compartilhamento é feito através de virtualização (Figura 2.5).

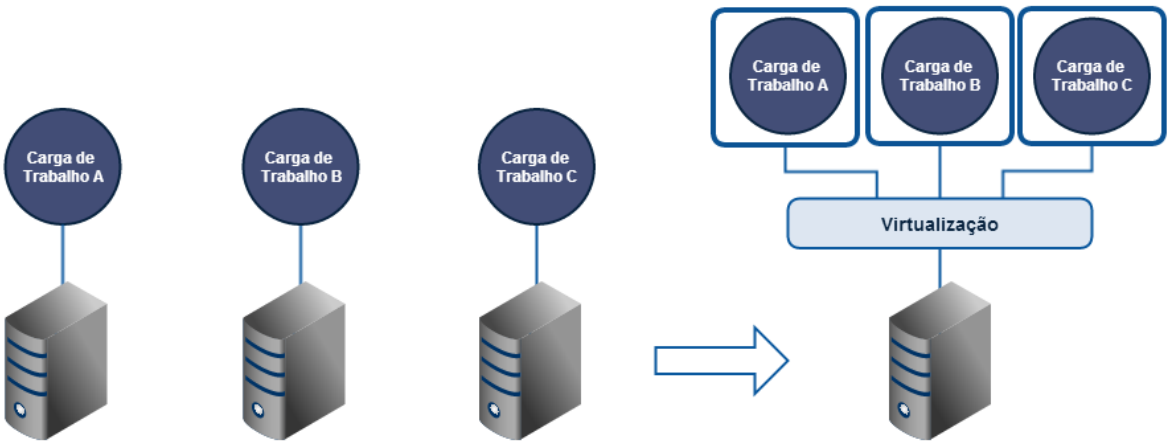

Figura 2.5 – Consolidação de servidores

Como já foi mencionado, aplicações com menor demanda por recursos mas que anteriormente precisavam ser alocadas em uma máquina física exclusiva atualmente podem ser alocadas em um mesmo servidor, compartilhando os recursos deste com outras VMs.

Dos benefícios proporcionados pela consolidação pode-se citar a redução nos custos com hardware em um *datacenter*, pois se pode atender um maior número de clientes com um menor número de servidores. Outro benefício é a redução do consumo energético do *datacenter*, já que com um menor número de máquinas, menos energia é consumida com as próprias máquinas, refrigeração, infraestrutura, etc. Ainda pode-se citar a redução de custos com equipe, já que, teoricamente, com uma infraestrutura mais compacta a complexidade do gerenciamento da mesma diminui.

No entanto, o isolamento entre máquinas virtuais ainda não é perfeito. VMs com uma demanda muito alta por um tipo específico de recurso podem prejudicar o desempenho das outras VMs que compartilham o mesmo servidor. Diversos estudos [19, 24, 38] mostram que o desempenho das VMs pode ser severamente afetado caso ocorram disputas pelo mesmo recurso computacional, em alguns casos fazendo com que VMs até mesmo parem de responder por completo.

Além disso, com a popularização do paradigma de computação em nuvem, que é altamente dependente de virtualização, este problema se tornou ainda mais importante. A virtualização forma a fundação de computação em nuvem uma vez que ela proporciona a capacidade de se criar reservas de recursos computacionais e os atribuir dinamicamente às aplicações que necessitam tais recursos. A consolidação errada ou muito agressiva de servidores pode resultar em congestionamento dos recursos compartilhados, como CPU e memória, degradando o desempenho geral do sistema [40].

Desta forma, políticas de consolidação de servidores que aperfeiçoem o uso do hardware disponível, aproveitando ao máximo os seus recursos sem exceder a sua capacidade, são um fator importante no desempenho de sistemas virtualizados. Além disso, deve-se levar em consideração, além das capacidades da máquina física, as características das VMs que serão alocadas em um mesmo servidor e realizar um arranjo que cause o mínimo possível de degradação de desempenho.

2.3 Interferência de desempenho

Com a popularização dos ambientes virtualizados e, recentemente, de nuvens computacionais, o problema de interferência de desempenho entre máquinas virtuais que compartilham um servidor tem sido um tópico estudado com frequência. Diversos trabalhos recentes avaliam a interferência causada entre os diversos tipos de recursos computacionais em diferentes plataformas de virtualização e constatam a existência do problema de interferência de desempenho.

Em [26] são realizados testes de interferência entre máquinas virtuais contendo cargas de trabalho intensivas em CPU e rede, usando uma plataforma de virtualização com *hypervisor* Xen. Os autores mostram que a degradação de desempenho pode chegar a mais de 50% quando existe o compartilhamento intensivo de CPU e a mais de 95% quando existe o compartilhamento intensivo de rede. Além disso, os autores chegam a conclusão de que o melhor desempenho é obtido quando o servidor é compartilhado por máquinas virtuais contendo cargas de trabalho de tipos diferentes (neste caso uma VM intensiva em CPU e outra intensiva em rede). Neste caso, o desempenho fica muito próximo do desempenho nativo da aplicação. Similar ao estudo anterior, o trabalho de Zhu et al. [41] mostra aumento de até 3,5 vezes no tempo de execução de duas aplicações intensivas em acesso a disco que compartilham o mesmo hardware.

Outra preocupação é levantada em [13]. Neste trabalho os autores sugerem que existe a possibilidade de que um usuário mal intencionado realize ataques maliciosos a máquinas virtuais em uma nuvem, que o autor chama de *cascading performance attacks*. Devido aos problemas de interferência de desempenho existentes em ambientes virtualizados, um ataque poderia ser realizado

instanciando-se uma máquina virtual que consuma os recursos do servidor intensivamente, causando degradação do desempenho das outras VMs hospedadas neste mesmo servidor.

Testes realizados em [15] mostram que duas máquinas virtuais iguais, e portanto com o mesmo tipo de demanda por recursos, alocadas no mesmo servidor resultam em degradação de desempenho em alguns casos superior a 40%. Neste estudo, duas máquinas virtuais são alocadas em um servidor usando Xen como *hypervisor*, e apesar de cada uma delas receber um núcleo exclusivo do processador, a disputa pelos outros recursos ainda existe resultando na perda de desempenho.

Em [38] foram realizados diversos testes para avaliar a sobrecarga causada pela virtualização, bem como testes que avaliam a interferência entre diferentes máquinas virtuais em um mesmo servidor. Os testes foram feitos em quatro plataformas diferentes: LXC, OpenVZ, VServer e Xen. Os resultados mostraram que a sobrecarga causada pela virtualização é muito pequena. Nestes testes foram executados *benchmarks* diretamente em uma máquina física e também em máquinas virtualizadas utilizando os sistemas apresentados anteriormente. Em praticamente todos os casos, o desempenho da execução virtualizada foi muito próximo do nativo. No entanto, nos testes de isolamento percebeu-se perda de desempenho devido à interferência de uma máquina virtual sobre outra. Nestes testes foram executadas diversas máquinas virtuais em um mesmo servidor e utilizou-se um benchmark especialmente desenvolvido para avaliar o isolamento de ambientes virtualizados, chamado *Isolation Benchmark Suite* [14]. Em alguns testes, a interferência foi tão acentuada que algumas máquinas virtuais pararam de responder.

Similar ao estudo anterior, em [19] também são realizados testes de isolamento usando o mesmo *benchmark*. Porém neste estudo são avaliados dois tipos de máquinas virtuais / aplicações: máquinas virtuais "bem comportadas" (*well-behaved*) e "mal comportadas" (*misbehaving*). São consideradas VMs "bem comportadas" as que têm requisitos medianos por recursos computacionais, ou, que causariam pouco impacto em outras VMs no mesmo servidor. Em contrapartida, as VMs "mal comportadas" são máquinas que necessitam grandes quantidades de recursos computacionais, prejudicando o desempenho de outras VMs que compartilham o mesmo servidor. Os resultados mostram que máquinas bem comportadas convivem bem com outras máquinas do mesmo tipo, causando pouca degradação de desempenho.

O estudo de [17] realizou testes de desempenho em diversos provedores de serviços de computação em nuvem. Os resultados mostram que máquinas virtuais iguais, com as mesmas características e quantidade de recursos, instanciadas em momentos diferentes apresentavam desempenhos diferentes, chegando a variações de até 300% em alguns casos. Estes resultados sugerem que o problema pode estar sendo causado pela política de consolidação usada pelos provedores, alocando VMs de maneira descuidada onde uma máquina virtual interfere no desempenho de outra.

Todos estes trabalhos apresentam possíveis problemas existentes com o isolamento entre VMs hospedadas em um mesmo servidor. É importante perceber que tais problemas não são inerentes a um único *hypervisor* ou tipo de virtualização, já que diferentes opções são abordadas pelos autores. Desta forma, uma abordagem que busque aperfeiçoar os *hypervisors* talvez não seja a melhor, já que seria necessário tratar cada um deles exclusivamente. Assim, o foco deste traba-

lho aborda a questão do mapeamento de máquinas virtuais que compartilham um mesmo servidor, buscando contornar o problema de interferência entre elas através da combinação que apresente a menor interferência.

3. TRABALHOS RELACIONADOS

Este capítulo procura apresentar o estado da arte na área de algoritmos de escalonamento e consolidação de servidores. Os trabalhos citados apresentam diferentes focos principais, como a redução de interferência [41, 6, 21], balanceamento de carga entre os servidores [22, 2], ou ainda a redução do consumo de energia da infraestrutura [32, 20]. No entanto, um dos objetivos buscados é comum a todos os trabalhos: minimizar o número de servidores necessários.

Em [41], os autores desenvolveram um algoritmo de consolidação de servidores que é dividido em duas partes: o modelo de interferência e o algoritmo de consolidação *online*. O modelo de interferência é subsequentemente dividido em outras duas partes: o "treinamento do modelo" e a matriz de interferência de recursos. O mapeamento inicial das máquinas virtuais é feita com base no modelo *offline* e mapeamentos subsequentes são feitos pelo algoritmo com base nas características da máquina virtual a ser mapeada e na matriz de interferência de recursos, que é obtida através do treinamento *offline* do modelo de interferência. Ainda, o algoritmo constantemente monitora as máquinas virtuais e atualiza o seu perfil de uso de recursos. O foco deste trabalho é garantir que as máquinas virtuais terminem seu trabalho dentro de um período pré-estabelecido, tentando consolidar os servidores o máximo possível.

Similar ao trabalho anterior, [6] apresenta o TRACON, um *framework* para alocação de tarefas e recursos que é composto de três componentes principais: o modelo de previsão de interferência, o escalonador e o monitor de tarefas e recursos. O monitor de tarefas e recursos monitora as máquinas virtuais e captura informações sobre o uso de recursos destas, e então utiliza estes dados para alimentar o modelo de previsão de interferência. O escalonador então pode usar uma de três estratégias para realizar o mapeamento: a primeira toma decisões rápidas, mapeando as máquinas virtuais no momento em que a requisição é criada, porém sem se preocupar se o mapeamento é o melhor possível. A segunda estratégia coloca as requisições em uma fila antes de realizar o mapeamento, e então toma as decisões de mapeamento com base nas requisições presentes na fila. Desta forma, o algoritmo demora mais para realizar o mapeamento mas consegue realizar combinações melhores. A terceira estratégia é uma combinação das duas primeiras.

Em [21], os autores propõem o conceito de *q-states* (ou estados-q), onde quanto mais alto é um *q-state*, maior é o nível de QoS requerido pela máquina virtual. Primeiramente, uma parte do algoritmo chamada de *staging server* faz um perfil de cada máquina virtual, identificando a quantidade de recursos necessários para que esta VM atinja o nível de QoS desejado, e então cada máquina virtual é mapeada pelo algoritmo de escalonamento que utiliza técnicas de *bin packing*. O escalonador mantém uma reserva de recursos em cada servidor e caso uma máquina virtual sofra interferência que vá reduzir o seu *q-state* abaixo do contratado, os recursos desta reserva são alocados para que a máquina virtual afetada mantenha o seu *q-state*. Ainda, se a reserva de recursos estiver disponível, é possível que um usuário pague um valor mais alto para que as suas máquinas virtuais atinjam um *q-state* mais alto ultilizando parte dos recursos disponíveis na reserva.

No entanto, este modelo tem uma desvantagem clara: se a reserva de recursos nunca for utilizada, seja para manter ou elevar o nível de QoS das máquinas virtuais, essa reserva é desperdiçada.

O trabalho apresentado em [22] apresenta uma política de mapeamento de máquinas virtuais que foca em realizar o balanceamento de carga entre os servidores da infraestrutura. O algoritmo apresentado verifica a carga existente em cada servidor disponível e então gera uma "roleta" que contém a probabilidade de em qual servidor a máquina virtual será mapeada, onde servidores com maior probabilidade tem maiores chances de receberem a máquina virtual. O mapeamento é gerado aleatoriamente com base na roleta de probabilidades e, apesar de a chance ser menor, ainda existe a possibilidade de que a máquina virtual seja mapeada em um servidor sobrecarregado.

Bobroff et al. [2] apresentam um algoritmo iterativo, que verifica a carga dos servidores periodicamente e então remapeia as máquinas virtuais utilizando migração. O mapeamento inicial é feito utilizando a heurística *first fit*, que procura diminuir o número de servidores necessários. Após o estágio inicial, o algoritmo entra em um ciclo de 3 partes: medição de carga, previsão de carga e remapemento para balancear a carga entre os servidores. As desvantagens desta abordagem são o fato de o mapeamento inicial não realizar controle de interferência, o que pode aumentar o número de migrações feitas já na primeira iteração do ciclo. Isso é prejudicial pois sabe-se que migração de máquinas virtuais gera sobrecargas nos servidores e na infraestrutura compartilhada [33, 12].

O trabalho apresentado em [32] foca em reduzir o consumo de energia enquanto mantém o desempenho do servidor acima de um nível estabelecido. Os autores analisam a relação entre o consumo de energia e a utilização do servidor e, a partir desta análise, desenvolvem um algoritmo que realiza o mapeamento procurando minimizar o consumo de energia enquanto mantém um determinado nível de desempenho do servidor. A solução utiliza algoritmos de aproximação (heurísticas) para realizar o mapeamento das máquinas virtuais pois, como é mencionado pelos autores, apesar de um algoritmo que encontra a solução ótima poder alcançar resultados melhores, podem existir casos onde o tempo necessário para realizar o mapeamento seja muito alto.

Similar ao trabalho anterior, em Moreno et al. [20] é proposto um algoritmo que procura otimizar a eficiência energética dos servidores reduzindo a interferência de desempenho resultante da disputa por recursos. Segundo testes realizados pelos autores, em situações de alta disputa por recursos, o consumo energético do servidor aumenta enquanto o desempenho registrado em cada máquina virtual mapeada neste servidor diminui. O algoritmo primeiramente classifica a carga de trabalho a ser mapeada, pré-seleciona um conjunto de servidores que está apto a receber esta requisição e então, com base em um monitor dinâmico que mantém o estado detalhado dos servidores pré-selecionados, realiza o mapeamento final no servidor onde existe a menor interferência.

A maioria destes trabalhos abordam o mapeamento de máquinas virtuais em datacenters similares a ambientes de nuvens públicas, onde não se tem controle sobre o que os clientes irão executar em suas máquinas virtuais e desta forma se faz necessário realizar o monitoramento das máquinas virtuais durante sua execução. Também percebe-se que diversas abordagens são feitas para tratar o problema de interferência, como migração [2] e reserva de recursos [21]. Alguns trabalhos criam diferentes tipos de modelos de interferência [41] [6], porém estes trabalhos desenvolvem modelos

de interferência muito específicos ou que são de difícil adaptação para outros cenários, enquanto outros utilizam o controle da interferência para reduzir o consumo de energia [32] [20]. A Tabela 3.1 apresenta uma comparação das características apresentadas por cada trabalho relacionado.

A coluna "Monitoração online"indica se a solução proposta realiza monitoração em tempo real do estado dos servidores e/ou máquinas virtuais a fim de alterar o modo como o mapeamento é feito. A coluna "Energia"indica se o trabalho foca em aprimorar a eficiência energética dos servidores ou da infraestrutura em geral. A coluna "Modelo de interferência"indica se o trabalho em questão desenvolve um modelo de interferência para ser utilizado em conjunto com a solução proposta. A coluna "Reserva de recursos"indica se a solução reserva parte dos recursos computacionais do servidor para atender VMs que estão sofrendo perda de desempenho por conta de interferência. Finalmente, a coluna "Migração"indica se a solução proposta utiliza migração de máquinas virtuais para reorganizar o mapeamento.

Tabela 3.1 – Características dos trabalhos estudados

Trabalho	**Monitoração online**	**Energia**	**Modelo de interferencia**	**Reserva de Recursos**	**Migração**
Zhang et al. 2011 [41]	Sim	Não	Sim	Não	Não
Chiang et al. 2011 [6]	Sim	Não	Sim	Não	Não
Nathuji et al. 2010 [21]	Sim	Não	Não	Sim	Não
Ni et al. 2011 [22]	Não	Não	Não	Não	Não
Bobroff et al. 2007 [2]	Sim	Não	Não	Não	Sim
Srikantaiah et al. 2008 [32]	Sim	Sim	Sim	Não	Não
Moreno et al. 2013 [20]	Sim	Sim	Sim	Não	Não

Percebe-se que a maioria dos trabalhos realiza monitoração online. Enquanto isto pode ser desejável em ambientes de nuvem pública, onde não se tem conhecimento das características das aplicações dos usuários, neste trabalhou optou-se pelo foco em datacenters privados onde se pode utilizar o conhecimento das cargas de trabalho executadas pelos usuários para simplificar e acelerar o mapeamento das VMs. Também percebe-se que quatro dos sete trabalhos analisados desenvolvem algum tipo de modelo de interferência porém, como mencionado anteriormente, a maioria deles é específico para a solução proposta pelo autor ou de difícil adaptação para outros casos. Este trabalho procura desenvolver um modelo de interferência que seja facilmente portado para outros cenários. Na proposta deste trabalho procurou-se evitar o uso de reserva de recursos e migração devido à sobrecarga que estas abordagens podem impor sobre o sistema em geral.

4. DESCRIÇÃO DO TRABALHO

Neste capítulo é apresentado o procedimento para a criação do modelo de interferência, *benchmarks* utilizados na análise de interferência, bem como as fórmulas matemáticas obtidas a partir desta análise. Também é apresentado em maiores detalhes como o algoritmo de mapeamento foi desenvolvido.

4.1 Modelo de interferência

Para identificar os níveis de interferência gerados pelo compartilhamento de cada recurso computacional foi realizada uma avaliação utilizando *benchmarks* que estressam cada um dos diferentes recursos.

Primeiramente, avaliou-se o desempenho de uma única máquina virtual que foi executada sozinha no servidor e, portanto, sem sofrer interferência de outras máquinas virtuais. Os resultados destes testes foram tomados como caso base para comparação. Após isto, foram realizados testes aumentando-se o número de máquinas virtuais alocadas no mesmo servidor, de forma a produzir interferência de desempenho entre as máquinas virtuais. Os resultados destes testes foram então comparados ao caso base para obter-se o nível de interferência produzido em cada caso onde existia compartilhamento de recursos. O ambiente de testes e a análise de interferência em cada tipo de recurso são apresentados a seguir.

4.1.1 Ambiente de testes

Os testes para a realização da análise de interferência foram feitos utilizando a plataforma de virtualização KVM [16]. Optou-se pelo KVM devido ao fato de o mesmo já vir incluído no *kernel* Linux, facilitando a realização dos testes. Cada uma das máquinas virtuais foi instanciada com 1 núcleo virtual, 4GB de memória RAM e 10GB de espaço em disco. A Tabela 4.1 apresenta a configuração do servidor onde foram realizados os testes.

Tabela 4.1 – Ambiente de Testes

Características do servidor	
CPU	2x Xeon Octa-Core E5-2650, 2 GHz, 20 MB cache L3
Memoria	4 x 16GB DDR3 1333 Mhz
Disco	SAS 300GB - 6Gbps
Hypervisor	KVM
SO	Linux Ubuntu 12.04

4.1.2 Análise de CPU

Para a realização da análise de CPU foram utilizados os *benchmarks* CRAFTY [8] e C-RAY [3]. O *benchmark* CRAFTY resolve partidas de xadrez e realiza operações lógicas e com números inteiros. O *benchmark* C-RAY realiza operações com números de ponto flutuante e foi escolhido de modo a complementar os resultados do primeiro teste.

É importante notar que na análise de CPU, a métrica avaliada é o tempo de execução. Nos resultados da análise de interferência apresentados a seguir, um nível de interferência de, por exemplo, 10%, indica que o teste demorou 10% a mais para ser completado do que o caso base. Da mesma forma, um nível de interferência de 100% indica que o teste demorou o dobro do tempo do caso base para ser finalizado.

Foram feitos testes alocando-se até 16 máquinas virtuais no mesmo servidor, cada uma delas ligada a um núcleo de CPU exclusivo, de modo que duas máquinas virtuais nunca compartilham o mesmo núcleo. A Figura 4.1 apresenta os resultados da análise de interferência de CPU, onde os pontos azuis indicam os valores obtidos através da medição feita com os *benchmarks*, enquanto a linha vermelha apresenta os valores obtidos através de análise de regressão.

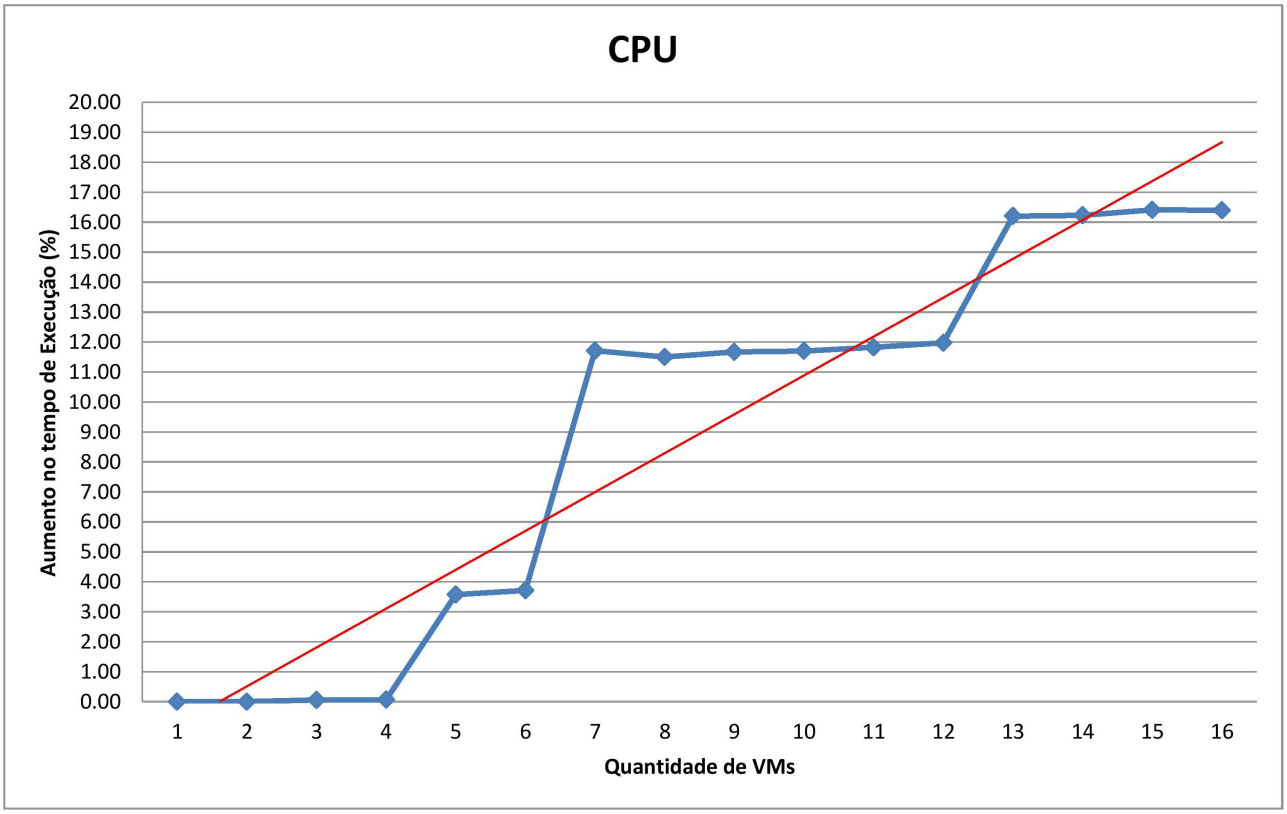

Figura 4.1 – Análise de interferência de CPU

É possível perceber que mesmo sem existir o compartilhamento de núcleos do processador, ainda assim existe perda de desempenho. Esta redução de desempenho é causada por outros fatores como o compartilhamento de memória cache, barramentos, bem como a sobrecarga imposta pelo

hypervisor. Com 16 máquinas virtuais alocadas no mesmo servidor existe um aumento no tempo de execução de em torno de 16%.

4.1.3 Análise de memória RAM

Para a realização da análise de memória RAM utilizou-se o *benchmark* chamado RAMSPEED [27]. Este *benchmark* realiza operações sintéticas de 4 tipos (*copy, scale, add, triad*) para medir a largura de banda alcançada pelo barramento de memória RAM. Nestes testes a métrica avaliada é a vazão dos dados (*throughput*), ou seja, um nível de interferência de 50% indica que a vazão de dados foi igual à metade da alcançada pelo caso base. A Figura 4.2 apresenta os resultados da análise de interferência de memória RAM, onde os pontos azuis indicam os valores obtidos através da medição feita com os *benchmarks*, enquanto que a linha vermelha apresenta os valores obtidos através de análise de regressão

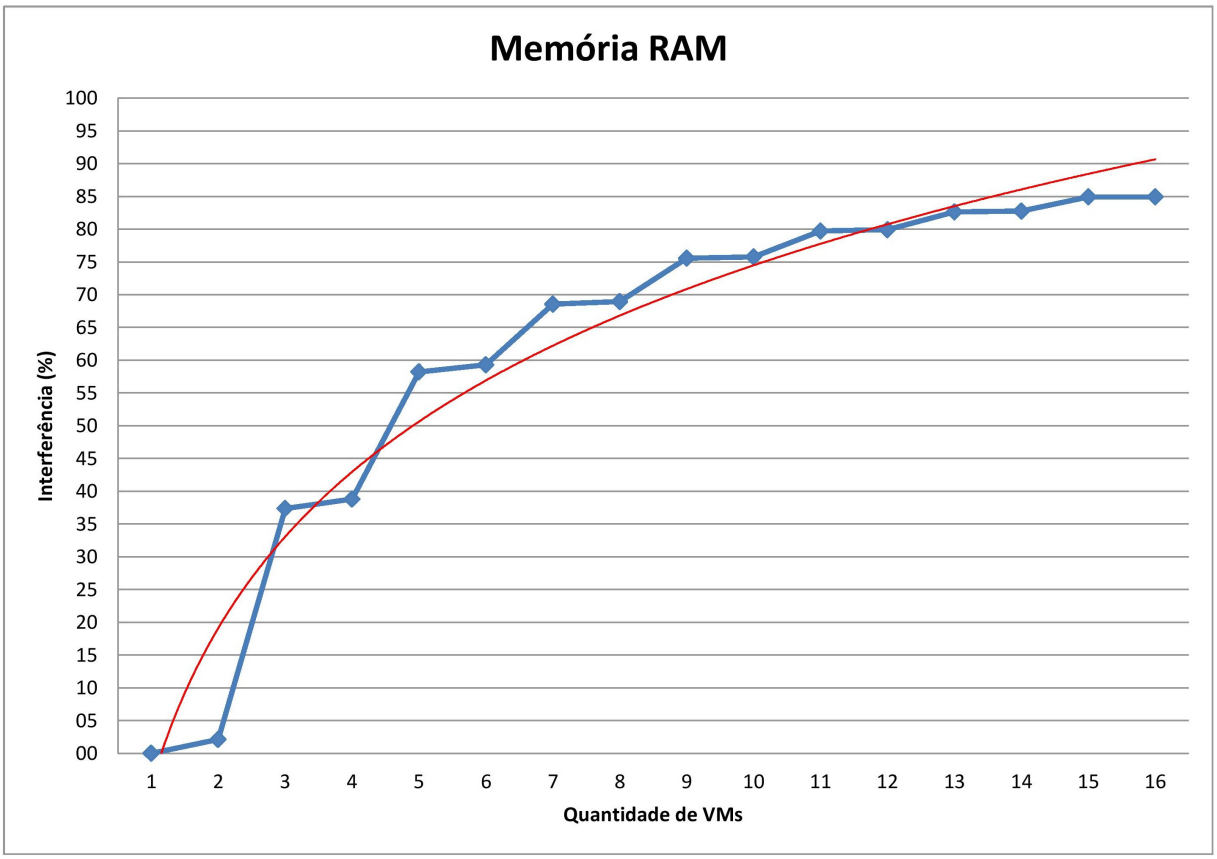

Figura 4.2 – Análise de interferência de memória RAM

Percebe-se que a interferência incidente sobre a memória RAM é consideravelmente mais severa do que a verificada nos testes de CPU. Com 16 máquinas compartilhando o mesmo servidor, a vazão de dados foi reduzida em aproximadamente 85%.

4.1.4 Análise de E/S de disco

Para a realização da análise de disco utilizou-se o *benchmark* TIOBENCH [34]. O TIOBENCH é um *benchmark multithreaded* que realiza 4 operações básicas (leitura sequencial, leitura aleatória, escrita sequencial e escrita aleatória) para avaliar o desempenho de entrada e saída do sistema de arquivos. Similar aos testes de memória RAM, nestes testes a métrica avaliada é a vazão dos dados (*throughput*), ou seja, um nível de interferência de 50% indica que a vazão de dados foi igual à metade da alcançada pelo caso base. A Figura 4.3 apresenta os resultados da análise de disco, onde os pontos azuis indicam os valores obtidos através da medição feita com os *benchmarks*, enquanto que a linha vermelha apresenta os valores obtidos através de análise de regressão.

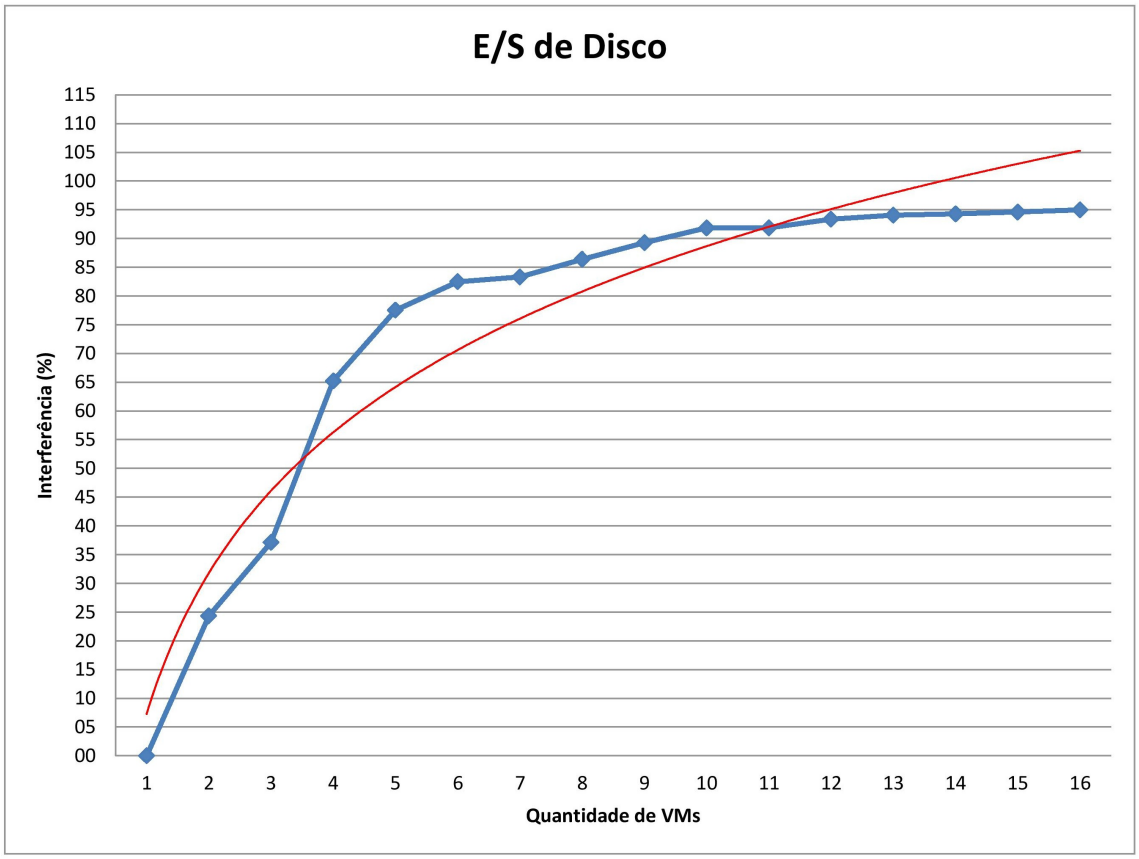

Figura 4.3 – Análise de interferência de E/S de disco

Os resultados da análise de disco mostram que a interferência incidente sobre este recurso é ainda mais severa que a de memória RAM, atingindo aproximadamente 95% quando 16 máquinas virtuais compartilham o mesmo servidor.

4.1.5 Fórmulas

Com os dados obtidos através da análise de interferência foi possível criar um modelo que prevê a interferência gerada pela combinação de máquinas virtuais. Este modelo é usado pelo

algoritmo de mapeamento para garantir que a interferência em cada servidor não ultrapasse um limiar pré-estabelecido. A Tabela 4.2 apresenta as fórmulas matemáticas referentes a cada recurso.

Tabela 4.2 – Fórmulas

Fórmulas matemáticas	
CPU (Linear)	Y = 1,2969x - 2,083
Memória (Logarítmica)	Y = 34,398ln(x) - 4,7183
Disco (Logarítmica)	Y = 35,347ln(x) + 7,2785

Neste modelo, o valor de Y representa o percentual de interferência gerado quando existe a combinação de X servidores que fazem uso intensivo do recurso computacional ao qual a fórmula se refere. É importante perceber que, apesar de as fórmulas não serem portáveis para outras plataformas sem sofrer alteração, a metodologia para obtê-las é suficientemente simples para que se possa gerar novas fórmulas para outras plataformas.

4.2 Algoritmo de mapeamento

Devido à semelhança que a consolidação de servidores tem o com o problema de *bin-packing*, é possível utilizar heurísticas para a construção do algoritmo de mapeamento de máquinas virtuais. O objetivo principal das heurísticas é encontrar boas soluções a um custo computacional aceitável, porém sem garantir que a solução encontrada seja ótima [11].

O algoritmo de mapeamento desenvolvido neste trabalho consiste na modificação das heurísticas chamadas *First Fit Decreasing* (FFD), *Best Fit Decreasing* (BFD) e *Worst Fit Decreasing* (WFD). As três heurísticas realizam pré-ordenação do conjunto de máquinas virtuais com base no tamanho de cada máquina virtual antes de iniciar o mapeamento.

A heurística FFD acomoda a máquina virtual a ser mapeada no primeiro servidor encontrado que possua espaço disponível, sem considerar o restante dos servidores após a primeira opção válida ser encontrada. A heurística BFD avalia todos os servidores disponíveis e aloca a máquina virtual no servidor que possui o menor espaço restante suficientemente amplo para acomodar a máquina virtual. Já a heurística WFD é análoga à heurística BFD, avaliando todos os servidores e então alocando a máquina virtual no servidor que possua o maior espaço restante e suficientemente amplo para acomodar a máquina virtual [7].

As três heurísticas foram modificadas, integrando o modelo de interferência apresentado anteriormente para que, sempre que uma nova máquina virtual é acomodada em um servidor, o algoritmo garanta que o nível de interferência não ultrapasse o limiar pré-estabelecido. O algoritmo baseado na heurística FFD percorre os servidores disponíveis e quando é encontrado um que tenha espaço suficiente para acomodar a máquina virtual a ser alocada, faz-se a verificação dos níveis de interferência para cada recurso em separado. Caso nenhuma delas ultrapasse o limiar estabelecido pelo usuário, a máquina virtual é alocada neste servidor. Caso o nível de interferência de algum dos recursos ultrapasse o limiar estabelecido, o algoritmo continua a percorrer os servidores disponíveis

em busca de um que esteja apto a receber a máquina virtual. Caso nenhum dos servidores existentes possa receber a máquina virtual, um novo servidor é adicionado à lista de servidores e a máquina virtual é alocada neste novo servidor. O Algoritmo 1 apresenta a abordagem FFD.

```
   Data: virtual machines, threshold
   Result: mapping
 1 Add new server;
 2 foreach virtual machine do
 3     index = 0;
 4     aloc = false;
 5     while aloc == false do
 6         if server[index] has space && (serverIntef + VM interf) < threshold
 7             server[index] ⟵ virtualmachine;
 8             aloc = true;
 9             break;
10         elif server is not the last on the server list
11             index++;
12             continue;
13         else
14             add new server;
15             server[last] ⟵ virtualmachine;
16             aloc = true;
17             break;
18     end
19 end
```

Algoritmo 1: Algoritmo FFD

O algoritmo baseado na heurística BFD percorre todos os servidores disponíveis, verificando a quantidade de espaço livre que cada um tem. Quando o primeiro servidor com espaço livre suficiente para armazenar a máquina virtual é encontrado, e se os níveis de interferência deste forem compatíveis com a máquina virtual a ser alocada, o algoritmo armazena o índice deste servidor e continua a verificar os outros servidores disponíveis. Após armazenar o primeiro servidor, o algoritmo continuará percorrendo os servidores disponíveis e atualizará o índice armazenado sempre que um servidor cujo o espaço restante após a alocação da VM for menor que o do servidor atualmente armazenado, e que tenha os níveis de interferência compatíveis com o limiar estabelecido. Caso o algoritmo encontre um servidor com espaço disponível igual ao tamanho da máquina virtual (não haverá espaço restante no servidor após a alocação) e com níveis de interferência que permitam que a máquina virtual seja alocada, a mesma é alocada neste servidor e o restante dos servidores não é verificado. Finalmente, caso não haja servidor apto a receber a máquina virtual, o algoritmo adiciona um novo servidor e mapeia a máquina virtual neste novo servidor. O Algoritmo 2 apresenta a abordagem BFD.

Analogamente à abordagem anterior, o algoritmo baseado na heurística WFD procura pelos servidores que possuam o maior espaço livre e tenham níveis de interferência compatíveis com a máquina virtual a ser alocada. O Algoritmo 3 apresenta esta abordagem.

```
   Data: virtual machines, threshold
   Result: mapping
1  Procedure findBestFitServer();
2  \\Procedure findBestFitServer searches for the server with least amount of free
   space enough to fit the VM;
3  Add new server;
4  foreach virtual machine do
5      foreach Server do
6          if freeSpace == VMSize && (serverIntef + VM interf) < threshold
7              server ⟵virtualmachine;
8              break;
9          elif findBestFitServer()
10             server[BestFit] ⟵virtualmachine;
11             break;
12         else
13             add new server;
14             server[last] ⟵virtualmachine;
15             aloc = true;
16             break;
17     end
18 end
```

Algoritmo 2: Algoritmo BFD

```
   Data: virtual machines, threshold
   Result: mapping
1  Procedure findWorstFitServer();
2  \\Procedure findWorstFitServer searches for the server with highest amount of
   free space enough to fit the VM;
3  Add new server;
4  foreach virtual machine do
5      foreach Server do
6          if freeSpace == 20
7              server ⟵virtualmachine;
8              break;
9          elif findWorstFitServer()
10             server[WorstFit] ⟵virtualmachine;
11             break;
12         else
13             add new server;
14             server[last] ⟵virtualmachine;
15             aloc = true;
16             break;
17     end
18 end
```

Algoritmo 3: Algoritmo WFD

O algoritmo funciona de forma *offline*, ou seja, não faz monitoramento do estado dos servidores ou máquinas virtuais em tempo de execução. Após o desenvolvimento do modelo de interferência e dos algoritmos de mapeamento, o próximo passo foi avaliá-los para verificar a eficácia dos mesmos. O Capítulo 5 apresenta os casos de teste criados para a avaliação dos algoritmos e os resultados destes testes.

5. AVALIAÇÃO DO TRABALHO

Neste capítulo são relatados os testes que foram realizados para avaliar a solução proposta. Foram elaborados quatro casos de teste com diferentes parâmetros para avaliar o desempenho do algoritmo em diferentes situações e o algoritmo foi avaliado utilizando-se simulação. Os parâmetros dos testes e a especificação dos casos de teste são apresentados a seguir.

5.1 Parâmetros

Os testes consistem em mapear conjuntos de 1000 máquinas virtuais no menor número de servidores possível sem ultrapassar o limiar de interferência estabelecido em nenhum dos servidores. Foram geradas máquinas virtuais de seis tamanhos diferentes e que fazem uso intensivo de diferentes combinações de recursos computacionais, sendo estes os recursos analisados anteriormente: CPU, memória RAM e E/S de disco. A Tabela 5.1 apresenta os tamanhos de máquinas virtuais utilizados nos testes.

Tabela 5.1 – Tamanhos das VMs

Tamanho da VM	**% de uso do servidor**
1	5
2	10
3	15
4	20
5	25
6	30

Desta forma, um servidor pode acomodar, por exemplo, cinco máquinas virtuais de tamanho 4, ou 10 máquinas virtuais de tamanho 2, desde que o limiar de interferência de desempenho não seja ultrapassado.

A distribuição do número de máquinas virtuais de cada tamanho foi feita com base em uma pesquisa [28] que mostra quais os tipos de instâncias mais populares no serviço de nuvem pública Amazon EC2 [29]. A Figura 5.1 apresenta os resultados desta pesquisa.

Além disso, outro parâmetro presente nos testes é o limiar (*threshold*) de interferência aceito. Quanto menor o limiar de interferência, mais servidores são necessários para acomodar a mesma quantidade de máquinas virtuais. Avaliou-se o comportamento do algoritmo com o limiar variando de 0% a 90% em incrementos de 10%.

Os resultados das heurísticas modificadas que foram apresentadas na seção 4.2 serão comparadas com os resultados das suas respectivas versões sem modificação e que, portanto, não realizam controle de interferência. Nos casos de testes, as heurísticas serão identificadas pelas abreviaturas apresentadas na Tabela 5.2:

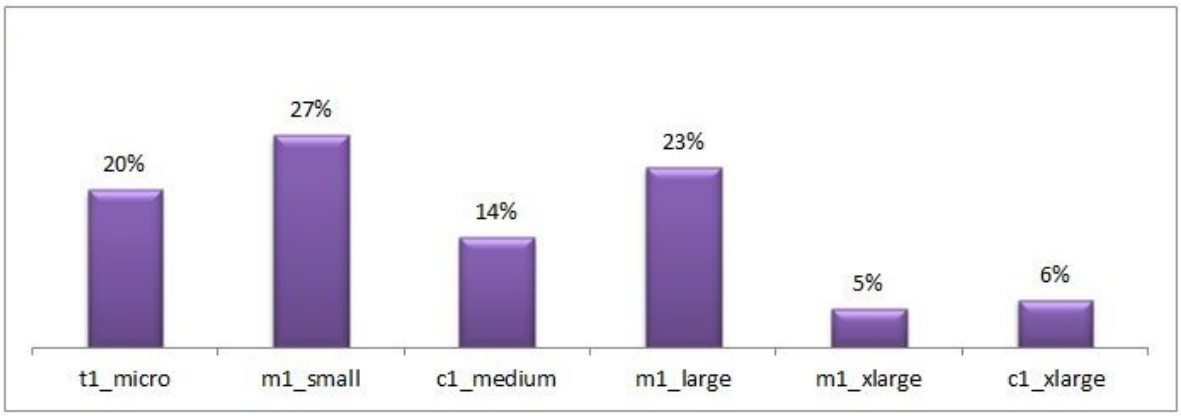

Figura 5.1 – Tipos de instâncias mais populares - Amazon EC2

Tabela 5.2 – Abreviaturas dos algoritmos analisados

Algoritmo	**Abreviaturas**
Best Fit Modificado	BFC
First Fit Modificado	FFC
Worst Fit Modificado	WFC

Como mencionado anteriormente, foram gerados quatro casos de teste onde cada um procura simular diferentes tipos de cargas de trabalho. No primeiro caso de teste, todas as máquinas virtuais fazem uso intensivo dos diferentes tipos de recurso na mesma proporção, ou seja, existe aproximadamente a mesma quantidade de VMs que fazem uso intensivo de CPU, memória RAM ou disco. Em cada um dos outros três casos de teste, produziu-se um conjunto de máquinas virtuais que simula uma carga de trabalho que faz uso intensivo de um recurso em específico. Nestes casos de teste, aproximadamente 80% das máquinas virtuais fazem uso intensivo de CPU (segundo caso de testes), memória RAM (terceiro caso de testes) ou disco (quarto caso de testes). Apesar de nos três ultimos casos de teste a carga de trabalho ser predominantemente intensiva em um tipo de recurso, ainda existe a possibilidade que uma máquina virtual faça uso intensivo de mais de um tipo de recurso computacional ao mesmo tempo. A Tabela 5.3 apresenta resumidamente as características da carga de trabalho simulada em cada caso de teste.

Tabela 5.3 – Características dos casos de testes

Caso de Teste	**Carga de Trabalho**
1	Uniforme
2	Intensiva em CPU
3	Intensiva em Memória RAM
4	Intensiva em E/S de Disco

Foram realizadas 100 iterações de cada teste. Com essa amostra obteve-se um coeficiente de variação de aproximadamente 10%. Os resultados dos casos de testes são apresentados a seguir.

5.2 Caso de teste 1

No primeiro caso de teste realizado, como já foi citado, o conjunto de máquinas virtuais gerado faz uso dos diferentes recursos computacionais na mesma proporção, de forma a simular uma carga de trabalho balanceada. Em todos os testes, a execução do algoritmo demorou menos que um segundo.

A Figura 5.2 apresenta a quantidade de servidores necessários para alocar as 1000 máquinas virtuais para cada limiar diferente.

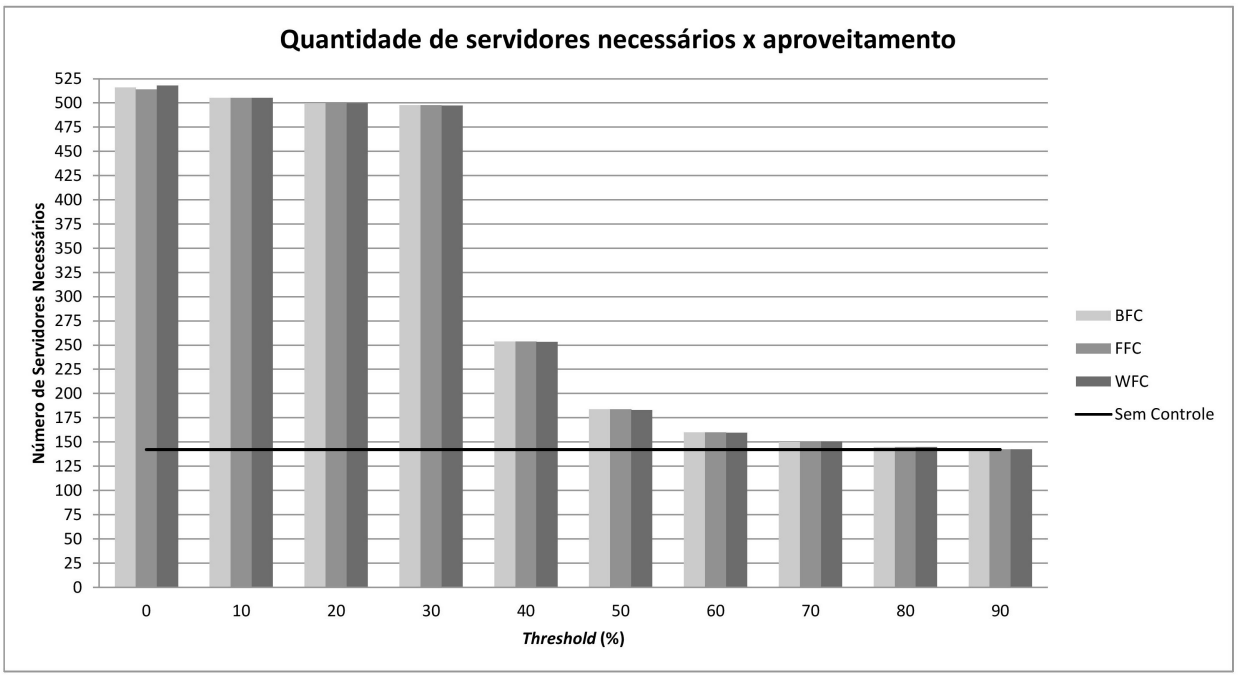

Figura 5.2 – Caso de teste 1 - Quantidade de servidores necessários

Neste caso de teste, o algoritmo que não faz controle dos níveis de interferência utilizou 143 servidores para realizar o mapeamento. Para evitar que se tenha interfência (limiar em 0%), foram utilizados de 514 a 518 servidores, o que representa um acréscimo de aproximadamente 360% no número de servidores necessários em relação ao algoritmo que não realiza controle. Também é possível constatar que o algoritmo conseguiu realizar o mapeamento respeitando o limiar de 60% com um acréscimo de apenas 11% no número de servidores necessários em comparação ao algoritmo que não realiza controle. Ainda é possível perceber que, com basicamente o mesmo número de servidores, os algoritmos que realizam controle de interferência conseguem assegurar que a interferência não irá ultrapassar 80%.

A Figura 5.3 apresenta a interferência média e máxima de CPU alcançada no primeiro caso de testes. A área contornada do gráfico representa a interferência média, enquanto a área sem contorno representa a interferência máxima. Deve-se levar em consideração que no caso dos testes de CPU a escala do gráfico é menor do que a apresentada nos outros recursos computacionais.

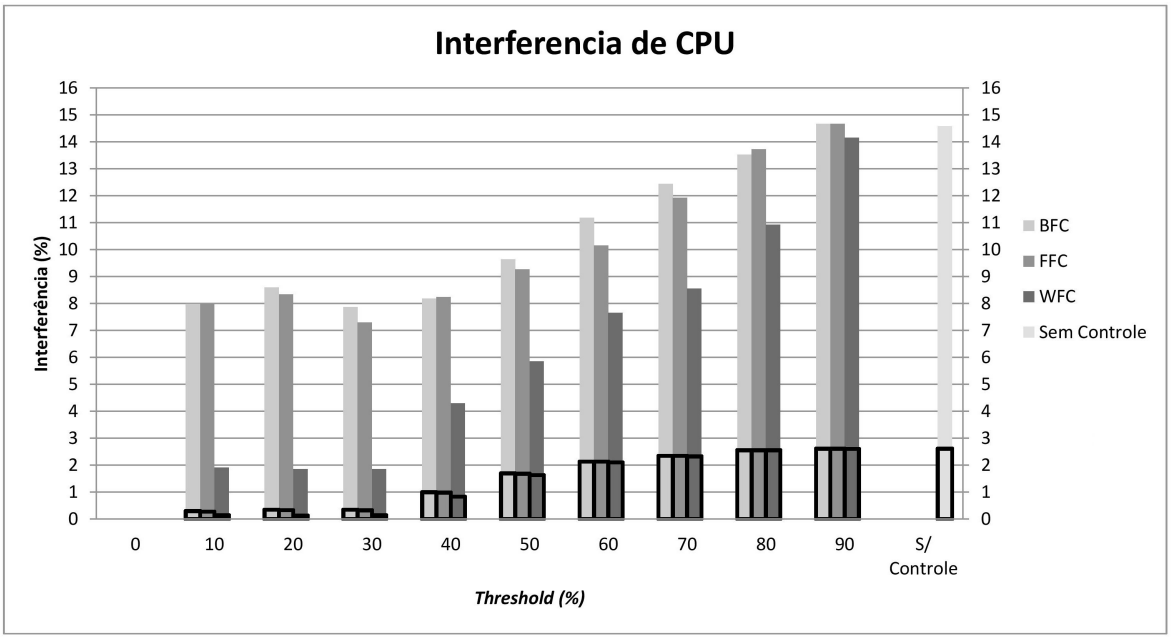

Figura 5.3 – Caso de teste 1 - Interferência de CPU

É possível constatar que, mesmo com limiares elevados, a interferência de CPU nunca ultrapassa 15%, o que já era esperado devido aos resultados da análise de interferência. Além disso, percebe-se que para todos os limiares, o algoritmo WFC tem um desempenho melhor, apresentando interferência máxima menor que os outros algoritmos que fazem controle de interferência.

Já a Figura 5.4 apresenta a interferência média e máxima de memória RAM para o mesmo caso de testes. A área contornada do gráfico representa a interferência média, enquanto a área sem contorno representa a interferência máxima.

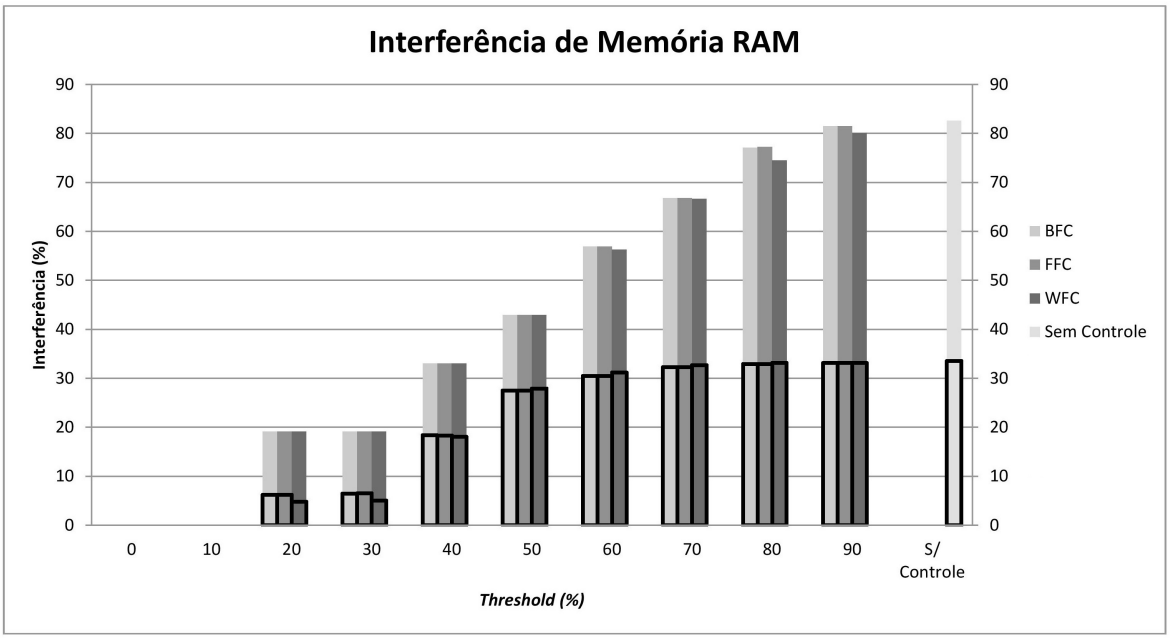

Figura 5.4 – Caso de teste 1 - Interferência de Memória RAM

Como já era esperado, devido ao comportamento constatado na análise de interferência, a interferência de memória RAM atinge níveis mais altos do que a interferência de CPU. Também é possível observar que para o limiar de 10% não existe interferência registrada. Isso acontece devido ao fato de o modelo de interferência prever que duas máquinas virtuais, que fazem uso intensivo de memória RAM, devem gerar interferência maior que 10% e dessa forma nunca mapeia duas VMs deste tipo em um mesmo servidor quando o limiar é estabelecido em 10%.

A figura 5.5 apresenta a interferência média e máxima de E/S de disco para o caso de teste 1. A área contornada do gráfico representa a interferência média, enquanto a área sem contorno representa a interferência máxima.

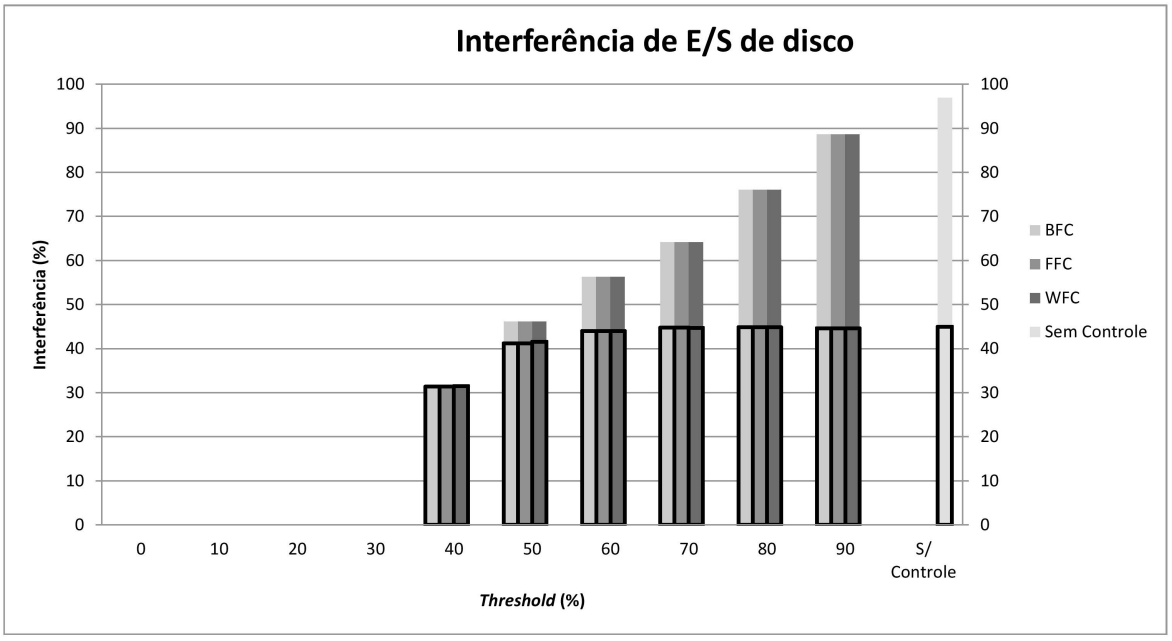

Figura 5.5 – Caso de teste 1 - Interferência de E/S de Disco

Devido a alta interferência gerada pelo compartilhamento de E/S de disco, em limiares menores que 40% o algoritmo nunca faz o mapeamento de duas máquinas virtuais intensivas em E/S de disco no mesmo servidor. Isso explica o fato de haver uma queda acentuada no número de servidores necessários quando se passa do limiar de 30% para o limiar de 40%, como pode ser verificado na Figura 5.2. Também constata-se que, no limiar de 40%, a interferência média é igual à máxima, o que indica que todos os servidores possuem duas máquinas virtuais que fazem uso intensivo de E/S de disco. Também é importante notar que o algoritmo que não faz controle produz uma interferência de mais de 95%. Como foi mencionado anteriormente, os algoritmos que fazem controle conseguem assegurar que a interferência não ultrapassará os 80% sem usar servidores extras, atingindo um mapeamento melhor mesmo quando o número de servidores disponíveis é limitado.

5.3 Caso de teste 2

O segundo caso de teste procura simular uma carga de trabalho que faz uso intensivo de CPU, enquanto os outros recursos são utilizados em menor proporção. Neste caso de testes, aproximadamente 80% das máquinas virtuais fazem uso intensivo de CPU. É importante lembrar que, mesmo que a carga de trabalho faça uso intensivo de CPU, ainda é possível que existam máquinas virtuais que fazem uso intensivo de mais de um recurso ou de nenhum deles.

A Figura 5.6 apresenta a quantidade de servidores necessários para acomodar todas as máquinas virtuais em cada limiar.

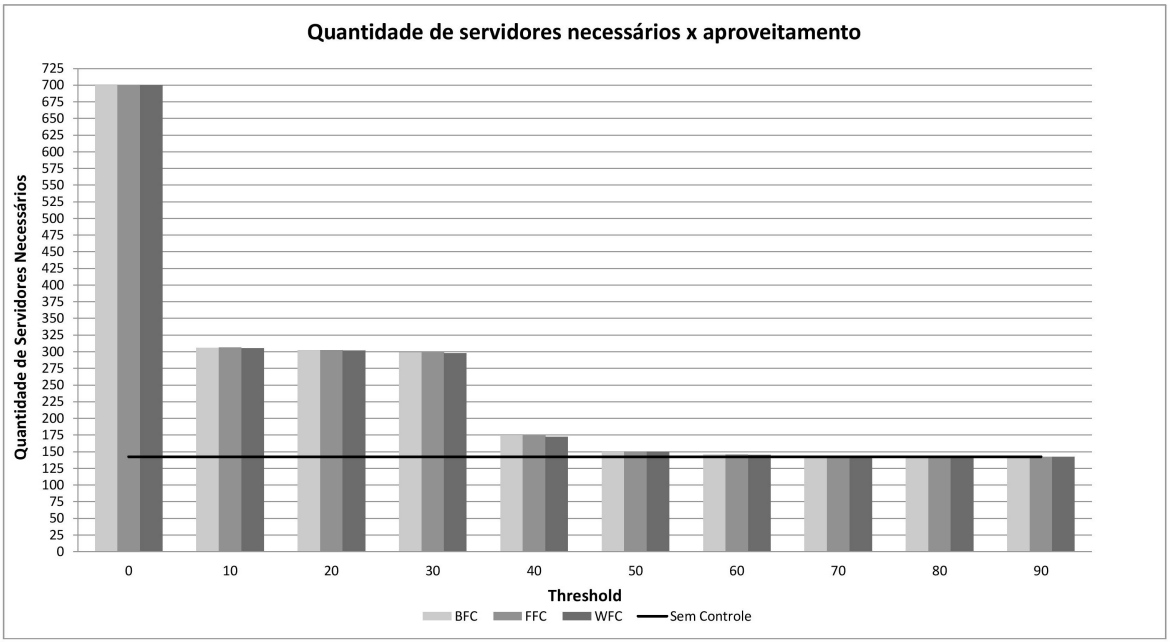

Figura 5.6 – Caso de teste 2 - Quantidade de servidores necessários

É possível perceber que já na mudança de limiar de 0% para 10% acontece uma redução significativa no número de servidores necessários, passando de 700 para pouco mais de 300. No entanto, em comparação ao algoritmo que não faz controle de interferência, isso ainda representa um aumento de aproximadamente 100% no número de servidores necessários para alcançar uma garantia de redução de 90% na interferência. A melhor relação entre acréscimo de servidores necessários em comparação à redução de interferência acontece nos limiares de 40% e 50%. É possível garantir que a interferência não irá ultrapassar 40% com um acréscimo de aproximadamente 21% no número de servidores (de 143 para 173), bem como, é possível garantir que a interferência não ultrapassará 50% com um acréscimo de aproximadamente 4% nos servidores (de 143 para 150).

A figura 5.7 apresenta a interferência média e máxima de CPU atingida em cada limiar. É possível perceber que, similar ao caso de testes 1, mesmo com limiares mais altos, a interferência máxima nunca ultrapassa os 20%. Isto indica que, mesmo quando a carga de trabalho faz uso intensivo de CPU, a interferência gerada pelos outros recursos ainda causa o maior impacto no modo como o mapeamento das máquinas virtuais é realizado.

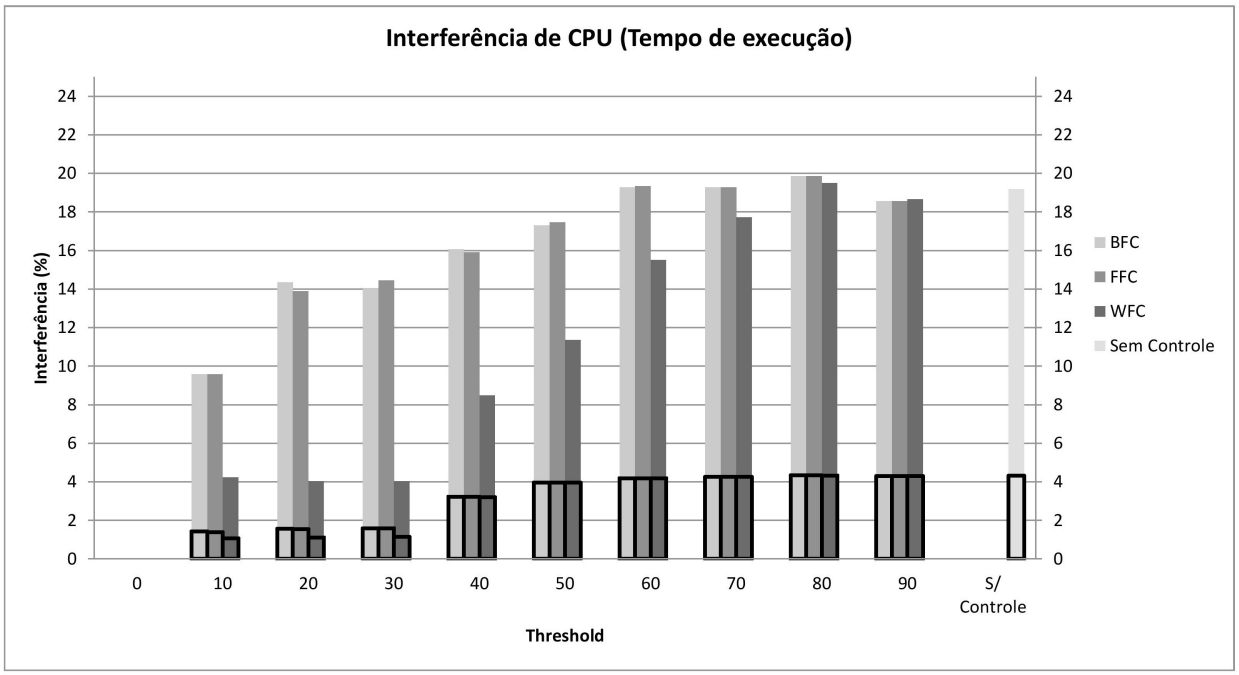

Figura 5.7 – Caso de teste 2 - Interferência de CPU

Pode-se notar também que dentre os algoritmos que realizam controle de interferência, o algoritmo WFC apresentou um resultado melhor, conseguindo manter a interferência máxima abaixo dos outros algoritmos na maioria dos limiares.

A figura 5.8 apresenta a interferência média e máxima de memória RAM.

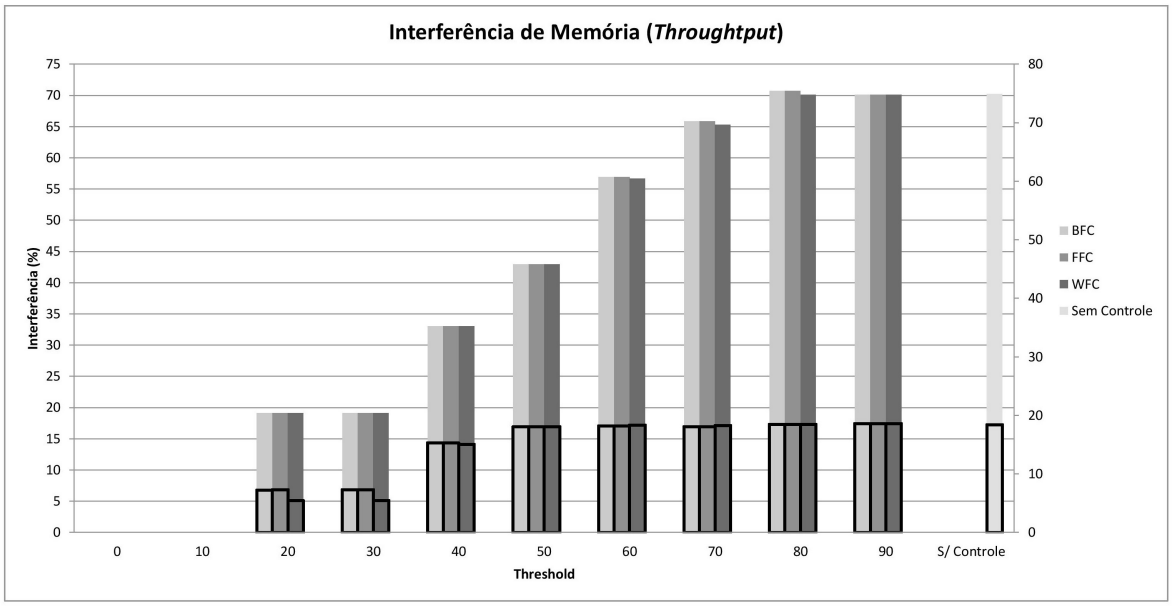

Figura 5.8 – Caso de teste 2 - Interferência de memória RAM

É possível perceber através da imagem que, em comparação ao caso de testes anterior, tanto a interferência média quanto a máxima foram mais baixas devido ao menor número de máquinas virtuais que fazem uso de memória RAM.

A figura 5.9 apresenta a interferência média e máxima de E/S de disco.

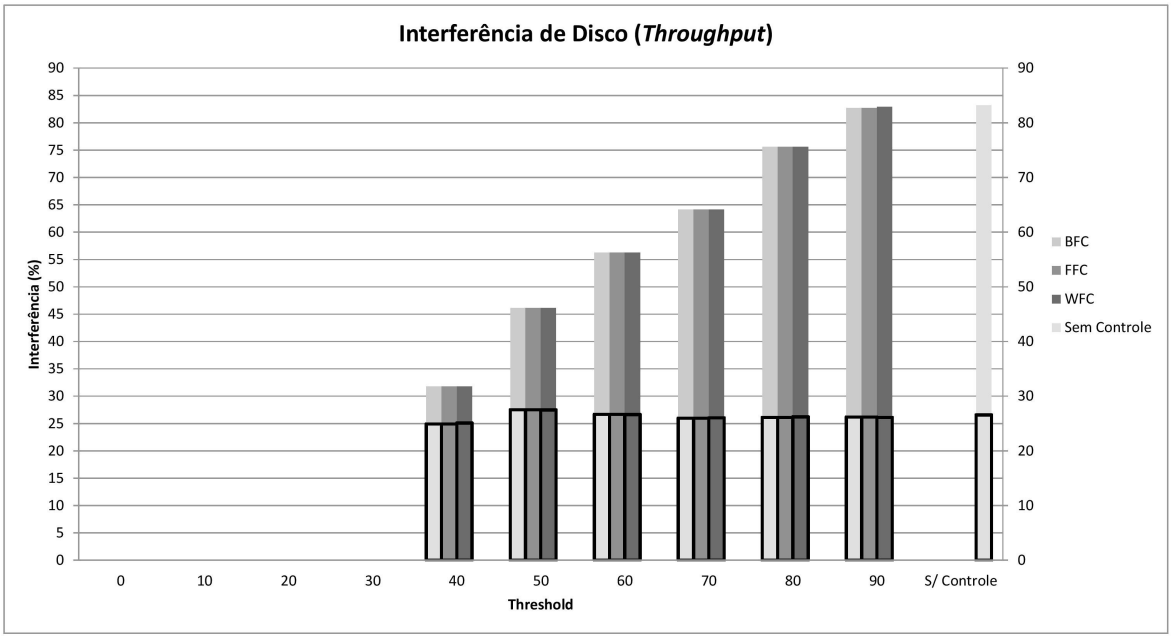

Figura 5.9 – Caso de teste 2 - Interferência de E/S de disco

Similar ao comportamento verificado na análise de memória RAM, em comparação ao caso de testes anterior, a interferência de disco foi menor devido ao menor número de máquinas virtuais que fazem uso intensivo deste recurso. No entanto, ainda assim percebe-se que o maior impacto no mapeamento é causado devido às máquinas virtuais que fazem uso intensivo de E/S de disco, pois mesmo com um menor número deste tipo de máquinas virtuais, ainda percebe-se uma redução considerável no número de servidores necessários a partir do limiar de 40%, que é quando se torna possível mapear mais de uma máquina virtual deste tipo no mesmo servidor.

5.4 Caso de teste 3

Este caso de teste simula uma carga de trabalho que faz uso intensivo de memória RAM. Similar ao caso de testes anterior, neste caso de testes aproximadamente 80% das máquinas virtuais fazem uso intensivo de memória RAM.

A Figura 5.10 apresenta a quantidade de servidores necessários para que seja possível mapear todas as máquinas virtuais em cada limiar estabelecido.

Devido ao fato de duas máquinas virtuais que fazem uso intensivo de memória RAM produzirem interferência maior do que 10%, o número de servidores necessários se mantém igual em limiares menores a 20%, quando se percebe uma redução significativa no número de servidores necessários. A melhor relação entre acréscimo de servidores necessários em comparação à redução de interferência acontece nos limiares de 50% e 60%, onde é possível garantir que a interferência não ultrapasse estes limiares com um aumento no número de servidores de aproximadamente 32% e 10%, respectivamente.

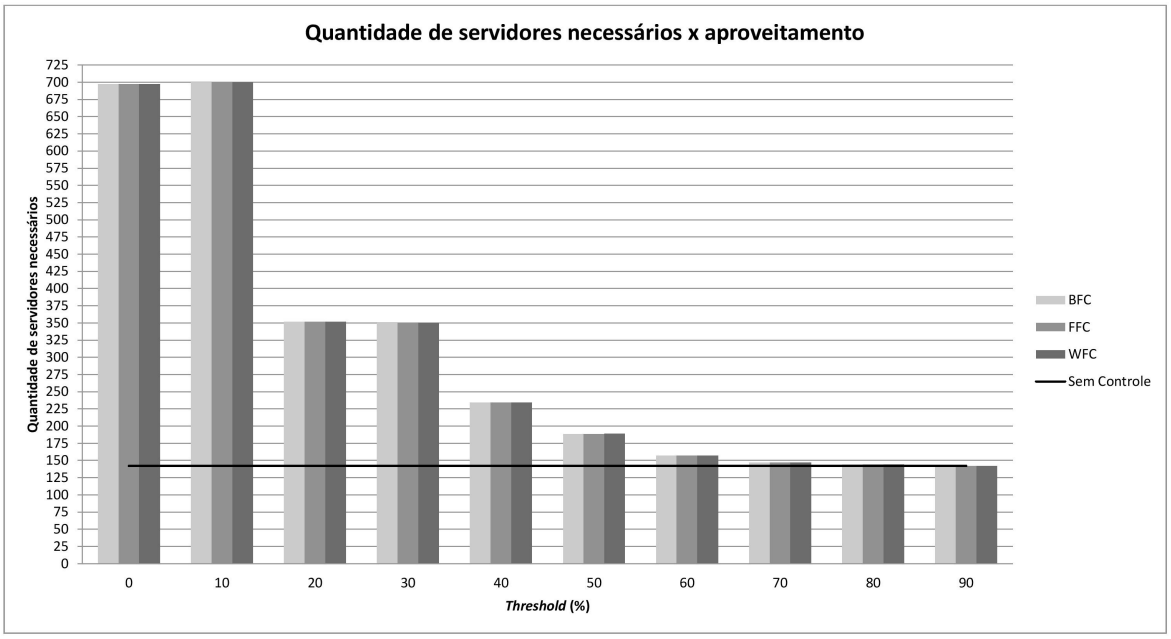

Figura 5.10 – Caso de teste 3 - Quantidade de servidores necessários

A Figura 5.11 apresenta a interferência média e máxima de CPU para este caso de uso.

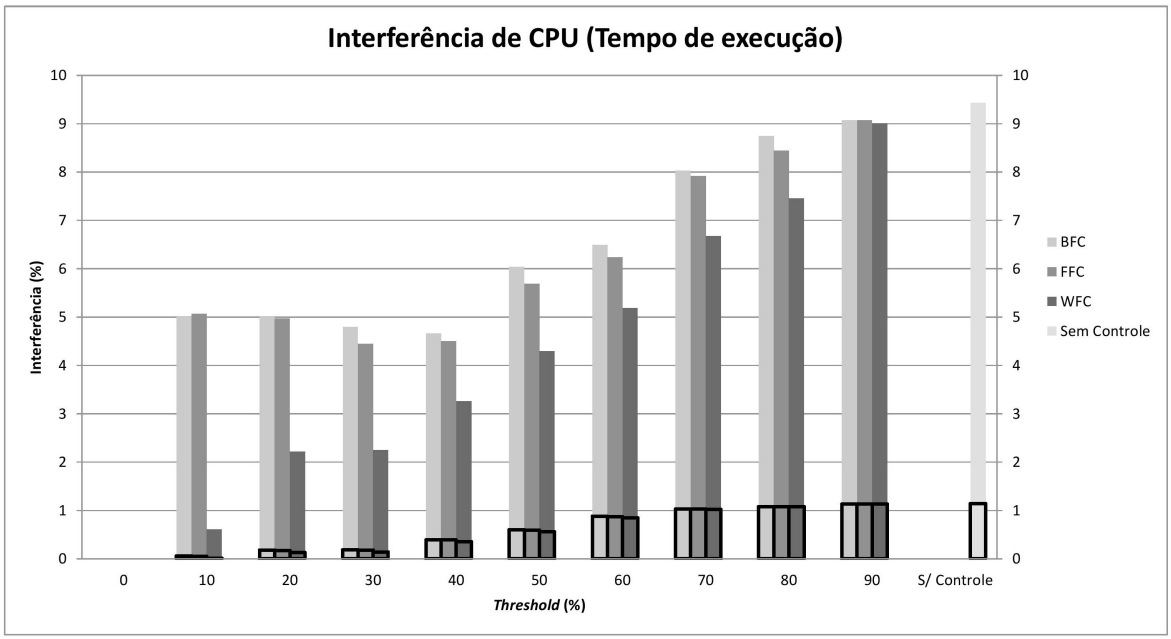

Figura 5.11 – Caso de teste 3 - Interferência de CPU

Neste caso de teste a interferência máxima de CPU foi menor do que a registrada nos casos de teste anteriores, devido ao menor número de máquinas virtuais que fazem uso intensivo de CPU. Ainda, repetindo o comportamento registrado anteriormente, o algoritmo WFC apresentou resultado melhor em comparação aos outros algoritmos que realizam controle, mantendo a interferência máxima mais baixa em todos os limiares.

A figura 5.12 apresenta a interferência média e máxima de memória RAM.

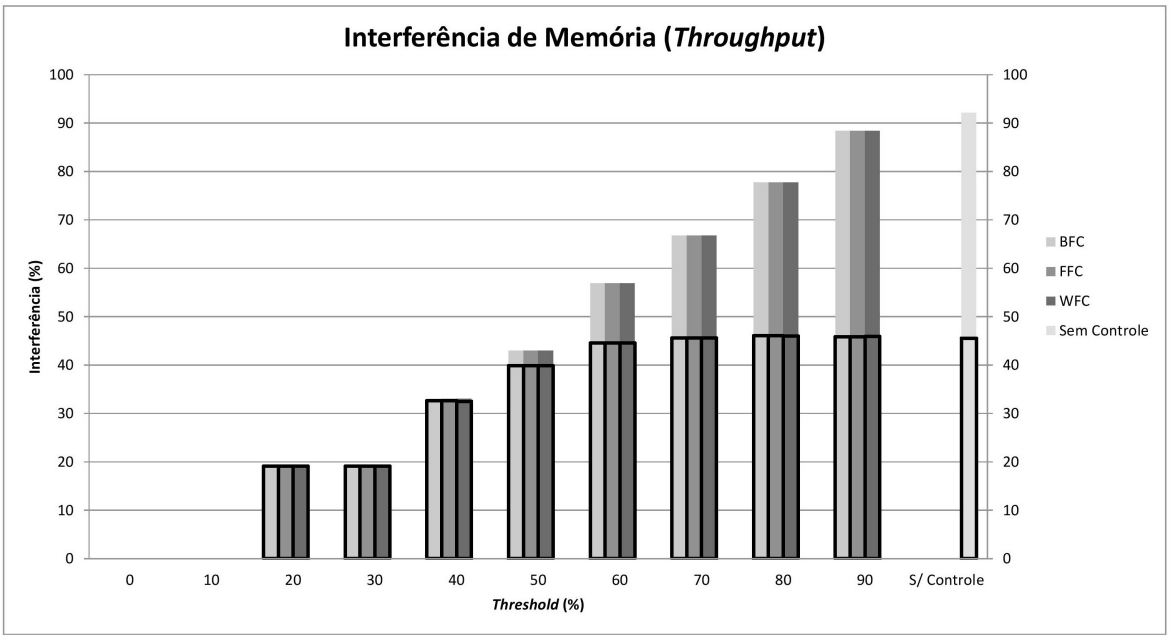

Figura 5.12 – Caso de teste 3 - Interferência de Memória RAM

Devido ao maior número de máquinas virtuais que fazem uso intensivo de memória RAM, neste caso de testes a interferência máxima registrada no mapeamento feito pelos algoritmos que não realizam controle foi acima de 90%, sendo a mais alta dentre os 3 recursos avaliados. Como foi constatado na Figura 5.10, foi possível garantir que a interferência máxima não ultrapasse o limiar de 60% com um aumento de apenas 10% no número de servidores, o que resulta em uma redução de aproximadamente 35% na interferência máxima, já que no limiar de 60%, a interferência máxima registrada foi de aproximadamente 57%.

Como pode ser constatado na Figura 5.13, os níveis de interferência de disco foram similares aos do caso de testes anterior, reforçando a constatação de que, mesmo em número menor, as máquinas virtuais que fazem uso intensivo de E/S de disco ainda criam impacto acentuado nos níveis de interferência.

5.5 Caso de teste 4

No último caso de teste realizado, procurou-se simular uma carga de trabalho que faz uso intensivo de E/S de disco. Seguindo os mesmos padrões dos casos de teste anteriores, neste caso aproximadamente 80% das máquinas virtuais fazem uso intensivo de E/S de disco.

Como pode ser verificado na Figura 5.14, a quantidade de servidores necessários para realizar o mapeamento se mantém a mesma em limiares abaixo de 40%, pois como foi mencionado anteriormente, o mapeamento de duas máquinas virtuais que fazem uso intensivo de E/S de disco em um mesmo servidor gera interferência maior que 30%. Como era esperado, este caso de teste foi o que se resultou no pior mapeamento, requerendo um acréscimo de 30% no número de servidores para manter o limiar de interferência em 60%.

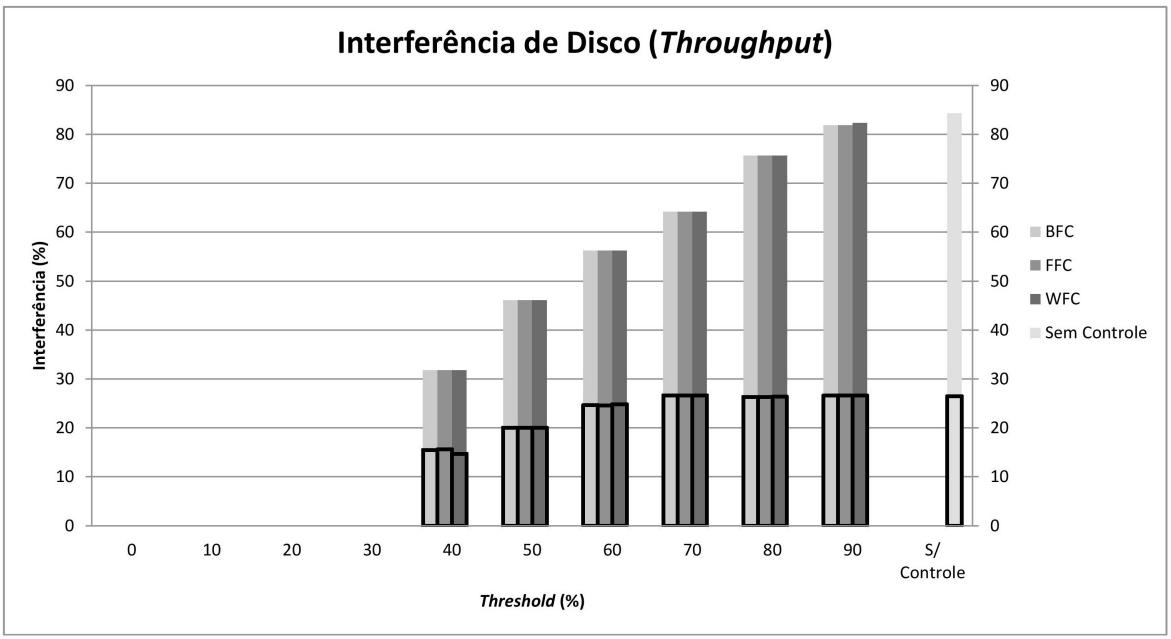

Figura 5.13 – Caso de teste 3 - Interferência de E/S de disco

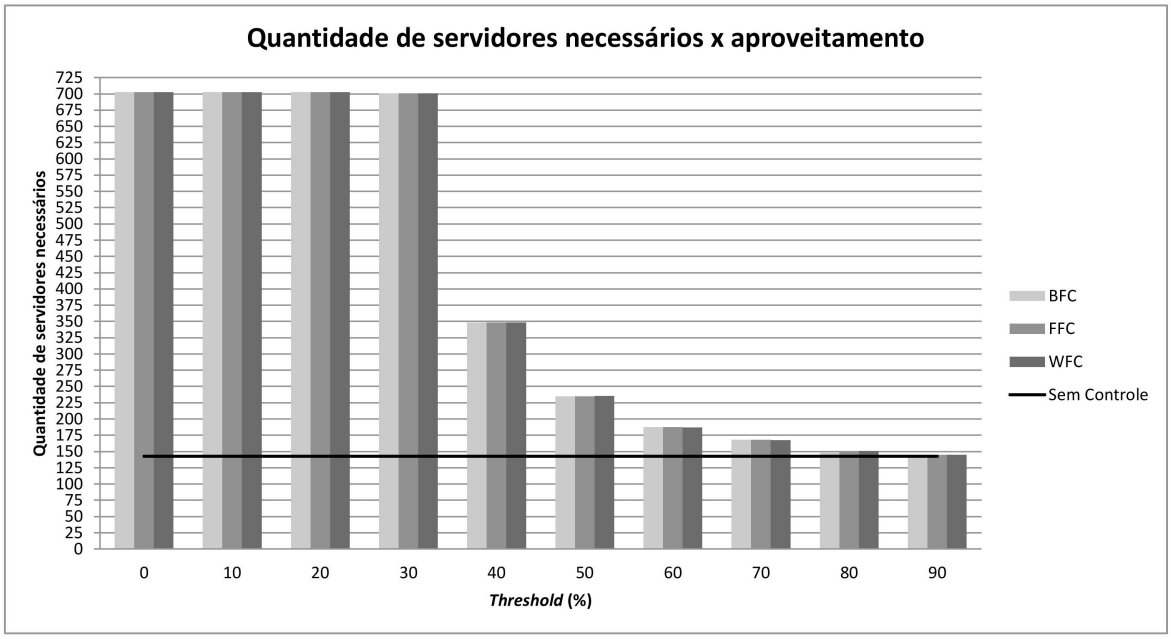

Figura 5.14 – Caso de teste 4 - Quantidade de servidores necessários

No entanto, como pode ser visto na figura 5.15, o nível máximo de interferência alcançado pelo algoritmo sem controle de interferência ultrapassa 100%, indicando que existem máquinas virtuais que pararam de responder devido à alta interferência. Com o mesmo número de servidores, os algoritmos que realizam controle de interferência conseguiram realizar um mapeamento que manteve a interferência abaixo do limiar de 90%, impedindo que houvessem máquinas virtuais que parassem de responder, e mantendo o mesmo nível de interferência média apresentado pelo algoritmo sem controle de interferência.

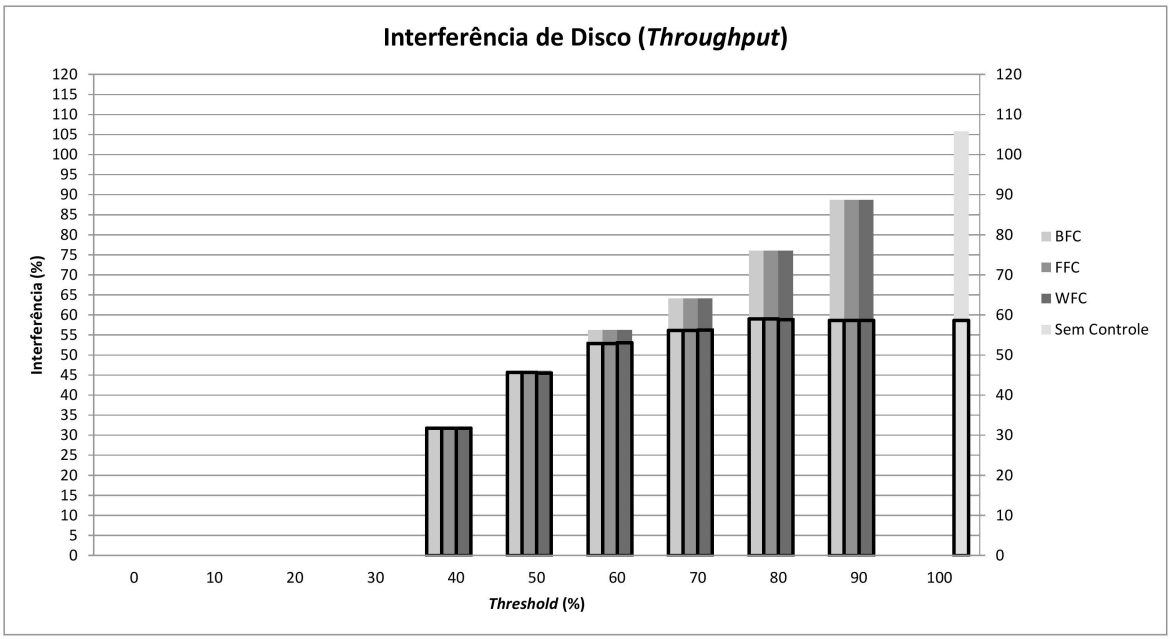

Figura 5.15 – Caso de teste 4 - Interferência de E/S de disco

As figuras 5.16 e 5.17 apresentam os níveis médios e máximos de interferência de CPU e memória RAM, respectivamente. O comportamento de ambos é similar aos que foram registrados nos casos de teste anteriores.

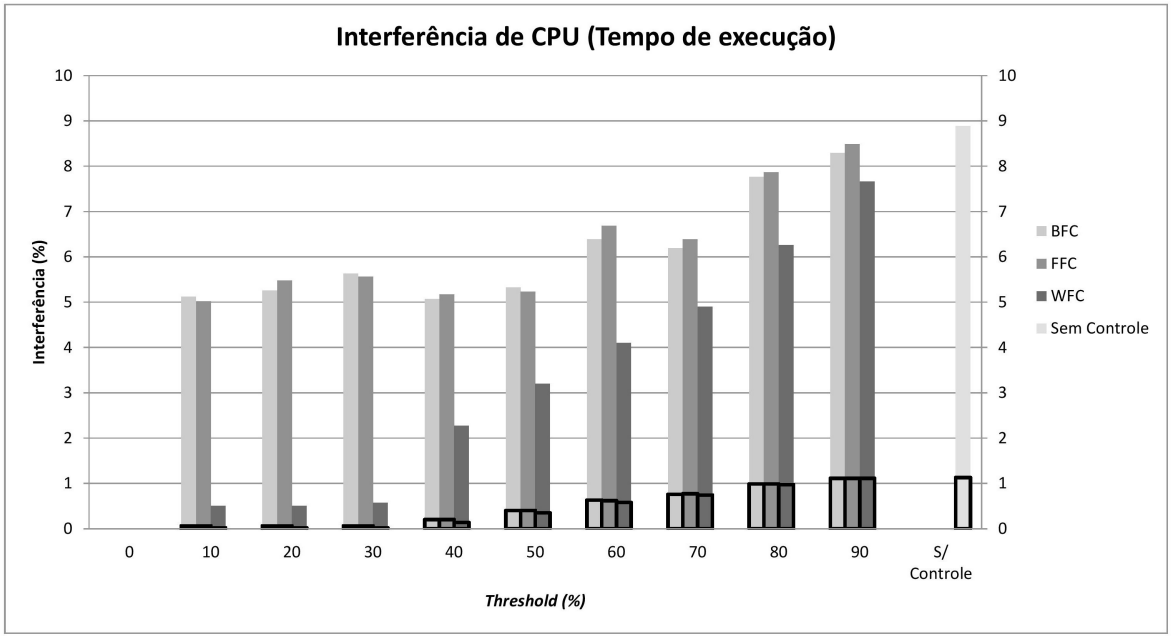

Figura 5.16 – Caso de teste 4 - Interferência de CPU

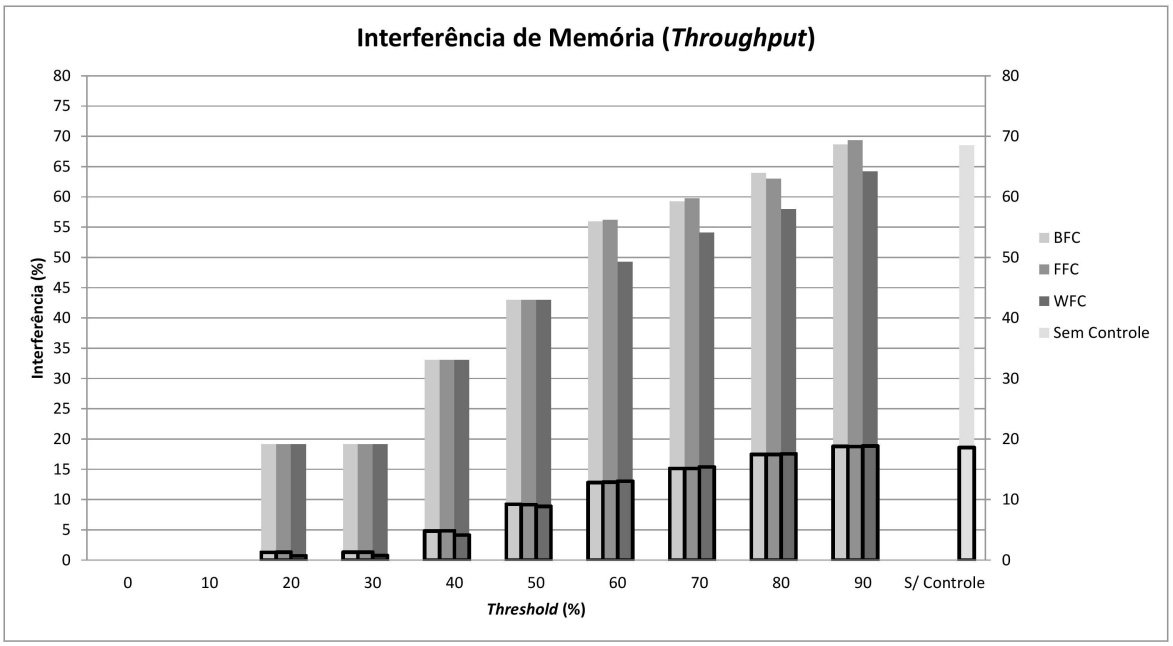

Figura 5.17 – Caso de teste 4 - Interferência de memória RAM

5.6 Conclusões da avaliação

A partir do resultado das avaliações foi constatado que o algoritmo é capaz de realizar o mapeamento mantendo a interferência abaixo do limiar estabelecido. Nos quatro casos de testes realizados, procurou-se simular diferentes cargas de trabalho a fim de verificar o comportamento do algoritmo e do modelo de interferência em diferentes situações e foi possível constatar que, na maioria dos casos, o algoritmo proposto conseguiu realizar um mapeamento mais eficiente que o algoritmo que não realiza controle de interferência sem aumentar expressivamente o número de servidores necessários. Em limiares abaixo dos 40%, devido principalmente ao alto nível de interferência gerado pelas máquinas virtuais que fazem uso intensivo de E/S de disco, a quantidade de servidores necessários foi bastante elevada em comparação ao algoritmo que não realiza controle de interferência. No entanto, em limiares entre 40% e 60% na maioria dos casos é possível alcançar uma boa relação entre servidores utilizados x redução de interferência.

A avaliação também mostrou que as três heurísticas apresentam resultados bastante similares, com resultados praticamente iguais para interferência de memória RAM e E/S de disco. Já no caso de interferência de CPU, o algoritmo *worst fit* apresentou redução de aproximadamente 5% em alguns casos.

6. CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Um dos objetivo deste trabalho foi análisar a interferência gerada pelo compartilhamento de CPU, memória RAM e E/S de disco, a fim de se desenvolver um modelo de interferência a ser utilizado pelo algoritmo para realizar o mapeamento das máquinas virtuais. Foi possível desenvolver o modelo de interferência que prevê os níveis de interferência resultantes do compartilhamento dos recursos computacionais, confirmando-se assim a primeira hipótese de pesquisa.

Outro objetivo foi desenvolver um algoritmo de mapeamento de máquinas virtuais em servidores que minimize o número de servidores necessários enquanto garante que a interferência de desempenho não ultrapasse um limiar pré estabelecido pelo usuário, utilizando o modelo de interferência desenvolvido anteriormente. Este objetivo foi alcançado, desta forma confirmando a segunda hipótese de pesquisa.

O algoritmo foi desenvolvido utilizando-se heurísticas, que apesar de apresentarem resultados sub-ótimos, são capazes de gerar tais resultados rapidamente, uma vez que o tempo necessário para que o algoritmo gere o mapeamento é um fator importante.

A proposta foi avaliada através da simulação de diferentes cargas de trabalho, onde foi possível comprovar que a mesma é capaz de realizar o mapeamento de máquinas virtuais mantendo o nível de interferência abaixo dos limiares pré-estabelecidos. Em diversos casos foi possível reduzir a interferência significativamente sem aumento expressivo no número de servidores necessários.

6.1 Contribuições

As contribuições principais deste trabalho são as seguintes:

- Avaliação do estado da arte em algoritmos e métodos de mapeamento de máquinas virtuais e consolidação de servidores.
- Metodologia para o desenvolvimento do modelo de interferência de desempenho.
- Desenvolvimento do modelo de interferência de desempenho.
- Desenvolvimento do algoritmo de mapeamento de máquinas virtuais.
- Avaliação do modelo de interferência e do algoritmo proposto.

6.2 Trabalhos Futuros

Como trabalhos futuros propõe-se abordar os seguintes tópicos:

- Otimizar o modelo de interferência para aprimorar a sua precisão.

- Estender o algoritmo de mapeamento para que este também utilize métodos mais complexos de otimização, como por exemplo, algoritmos genéticos ou programação linear, e comparar a qualidade dos resultados e tempo de produção do mapeamento em comparação ao algoritmo atual.
- Avaliar os níveis de interferência e desenvolver modelos de interferência para diferentes plataformas de virtualização, uma vez que neste trabalho avaliou-se somente a plataforma KVM.
- Adaptar o modelo de interferência para as plataformas de virtualização existentes, de modo que o mesmo possa ser utilizado em ambientes reais.
- Estender o modelo de interferência para que trabalhe de forma *online*.
- Estender o modelo para que aborde questões como QoS, eficiência energética, entre outras.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Barham, P.; Dragovic, B.; Fraser, K.; Hand, S.; Harris, T.; Ho, A.; Neugebauer, R.; Pratt, I.; Warfield, A. "Xen and the art of virtualization", *ACM SIGOPS Operating Systems Review*, vol. 37–5, 2003, pp. 164–177.

[2] Bobroff, N.; Kochut, A.; Beaty, K. "Dynamic placement of virtual machines for managing sla violations". In: Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on, 2007, pp. 119–128.

[3] C-RAY. "C-ray". Capturado em: http://openbenchmarking.org/test/pts/c-ray, 2013.

[4] Carissimi, A. "Virtualização: da teoria a solucões", *Minicursos do Simpósio Brasileiro de Redes de Computadores–SBRC*, vol. 2008, 2008, pp. 173–207.

[5] Chaudhary, V.; Cha, M.; Walters, J.; Guercio, S.; Gallo, S. "A comparison of virtualization technologies for hpc". In: Advanced Information Networking and Applications, 2008. AINA 2008. 22nd International Conference on, 2008, pp. 861 –868.

[6] Chiang, R. C.; Huang, H. H. "Tracon: Interference-aware scheduling for data-intensive applications in virtualized environments". In: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, 2011, pp. 47.

[7] Coffman Jr, E. G.; Garey, M. R.; Johnson, D. S. "Approximation algorithms for bin packing: A survey". In: Approximation algorithms for NP-hard problems, 1996, pp. 46–93.

[8] Crafty. "Crafty". Capturado em: http://www.spec.org/cpu2000/CINT2000/186.crafty/docs/186.crafty.html, 2013.

[9] Daniels, J. "Server virtualization architecture and implementation", *Crossroads*, vol. 16–1, Set 2009, pp. 8–12.

[10] Devices, A. M. "Amd64 virtualization codenamed pacifica technology". Capturado em: http://www.mimuw.edu.pl/~vincent/lecture6/sources/amd-pacifica-specification.pdf, 2013.

[11] Ferreto, T. C. "Dynamic server consolidation with controlled reconfiguration delays", Doutorado, Pontifícia Universidade Católica do Rio Grande do Sul, 2010.

[12] Ferreto, T. C.; Netto, M. A.; Calheiros, R. N.; De Rose, C. A. "Server consolidation with migration control for virtualized data centers", *Future Generation Computer Systems*, vol. 27–8, 2011, pp. 1027–1034.

[13] Huang, Q.; Lee, P. P. "An experimental study of cascading performance interference in a virtualized environment", *ACM SIGMETRICS Performance Evaluation Review*, vol. 40–4, 2013, pp. 43–52.

[14] IBS. “Isolation benchmark suite”. Capturado em: http://web2.clarkson.edu/class/cs644/isolation, 2013.

[15] Kang, S.; Kim, S.-g.; Eom, H.; Yeom, H. Y. “Towards workload-aware virtual machine consolidation on cloud platforms”. In: Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication, 2012, pp. 45:1–45:4.

[16] KVM. “Kvm”. Capturado em: http://www.linux-kvm.org, 2013.

[17] Lenk, A.; Menzel, M.; Lipsky, J.; Tai, S.; Offermann, P. “What are you paying for? performance benchmarking for infrastructure-as-a-service offerings”. In: Cloud Computing (CLOUD), 2011 IEEE International Conference on, 2011, pp. 484 –491.

[18] LXC. “Linux containers”. Capturado em: http://lxc.sourceforge.net, 2013.

[19] Matthews, J. N.; Hu, W.; Hapuarachchi, M.; Deshane, T.; Dimatos, D.; Hamilton, G.; McCabe, M.; Owens, J. “Quantifying the performance isolation properties of virtualization systems”. In: Proceedings of the 2007 workshop on Experimental computer science, 2007.

[20] Moreno, I. S.; Yang, R.; Xu, J.; Wo, T. “Improved energy-efficiency in cloud datacenters with interference-aware virtual machine placement”. In: Autonomous Decentralized Systems (ISADS), 2013 IEEE Eleventh International Symposium on, 2013, pp. 1–8.

[21] Nathuji, R.; Kansal, A.; Ghaffarkhah, A. “Q-clouds: managing performance interference effects for qos-aware clouds”. In: Proceedings of the 5th European conference on Computer systems, 2010, pp. 237–250.

[22] Ni, J.; Huang, Y.; Luan, Z.; Zhang, J.; Qian, D. “Virtual machine mapping policy based on load balancing in private cloud environment”. In: Cloud and Service Computing (CSC), 2011 International Conference on, 2011, pp. 292–295.

[23] OpenVZ. “Openvz”. Capturado em: http://www.openvz.org, 2013.

[24] Padala, P.; Zhu, X.; Wang, Z.; Singhal, S.; Shin, K. G.; Padala, P.; Zhu, X.; Wang, Z.; Singhal, S.; Shin, K. G. “Performance evaluation of virtualization technologies for server consolidation”, Relatório Técnico, 2007.

[25] Paul, I.; Yalamanchili, S.; John, L. “Performance impact of virtual machine placement in a datacenter”. In: Performance Computing and Communications Conference (IPCCC), 2012 IEEE 31st International, 2012, pp. 424–431.

[26] Pu, X.; Liu, L.; Mei, Y.; Sivathanu, S.; Koh, Y.; Pu, C. “Understanding performance interference of i/o workload in virtualized cloud environments”. In: Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on, 2010, pp. 51–58.

[27] RamSpeed. "Ramspeed". Capturado em: http://openbenchmarking.org/test/pts/ramspeed, 2013.

[28] Reports, D. C. "Aws ec2 instance type survey – what is popular?" Capturado em: http://www.newvem.com/aws-ec2-instance-type-survey-what-is-popular/, 2013.

[29] Services, A. W. "Amazon elastic compute cloud". Capturado em: http://aws.amazon.com/pt/ec2/, 2013.

[30] Smith, J. "Virtual machines : versatile platforms for systems and processes". Amsterdam Boston: Morgan Kaufmann Publishers, 2005.

[31] Somani, G.; Chaudhary, S. "Application performance isolation in virtualization". In: Cloud Computing, 2009. CLOUD '09. IEEE International Conference on, 2009, pp. 41 –48.

[32] Srikantaiah, S.; Kansal, A.; Zhao, F. "Energy aware consolidation for cloud computing". In: Proceedings of the 2008 conference on Power aware computing and systems, 2008.

[33] 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, 2009, pp. 9–14.

[34] TIOBench. "Tiobench". Capturado em: http://openbenchmarking.org/test/pts/tiobench, 2013.

[35] Uhlig, R.; Neiger, G.; Rodgers, D.; Santoni, A.; Martins, F.; Anderson, A.; Bennett, S.; Kagi, A.; Leung, F.; Smith, L. "Intel virtualization technology", *Computer*, vol. 38–5, may 2005, pp. 48 – 56.

[36] VMWare. "Vmware". Capturado em: http://www.vmware.com, 2013.

[37] VServer, L. "Linux vserver". Capturado em: http://linux-vserver.org, 2013.

[38] Xavier, M. G.; Neves, M. V.; Rossi, F. D.; Ferreto, T. C.; Lange, T.; De Rose, C. A. "Performance evaluation of container-based virtualization for high performance computing environments". In: Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on, 2013, pp. 233–240.

[39] Xen. "Xen". Capturado em: http://www.xen.org, 2013.

[40] Zhang, Q.; Cheng, L.; Boutaba, R. "Cloud computing: state-of-the-art and research challenges", *Journal of Internet Services and Applications*, vol. 1–1, Maio 2010, pp. 7–18.

[41] Zhu, Q.; Tung, T. "A performance interference model for managing consolidated workloads in qos-aware clouds". In: Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on, 2012, pp. 170–179.