Alocação de Máquinas Virtuais em Ambientes de Computação em Nuvem Considerando o Compartilhamento de Memória

Fernando José Muchalski[1], **Carlos Alberto Maziero**[1]

[1]Universidade Tecnológica Federal do Paraná (UTFPR)
Curitiba – PR – Brazil

muchalski@gmail.com, maziero@utfpr.edu.br

Resumo. *Nos ambientes de computação em nuvem, é importante manter sob controle a alocação de máquinas virtuais nos servidores físicos. Uma alocação adequada implica na redução de custos com hardware, energia e refrigeração, além da melhora da qualidade de serviço. Hipervisores recentes implementam mecanismos para reduzir o consumo de memória RAM através do compartilhamento de páginas idênticas entre máquinas virtuais. Este artigo apresenta um novo algoritmo de alocação de máquinas virtuais que busca o equilíbrio no uso dos recursos de CPU, memória, disco e rede e, sobretudo, considera o potencial de compartilhamento de memória entre máquinas virtuais. Através de simulações em três cenários distintos, verificou-se que o algoritmo é superior à abordagem padrão na questão do uso equilibrado de recursos e que, considerando o compartilhamento de memória, houve um ganho significativo na disponibilidade deste recurso ao final das alocações.*

Abstract. *In cloud computing environments it is important to keep under control the allocation of virtual machines in physical servers. A good allocation brings benefits such as reduction costs in hardware, power, and cooling, also improving the quality of service. Recent hypervisors implement mechanisms to reduce RAM consumption by sharing identical pages between virtual machines. This paper presents a new algorithm for virtual machines allocation that seeks the balanced use of CPU, memory, disk, and network. In addition, it considers the potential for sharing memory among virtual machines. Simulations on three distinct scenarios demonstrate that it is superior to the standard approach when considering the balanced use of resources. Considering shared memory, there was an appreciable gain in availability of resources.*

1. Introdução

A tecnologia de virtualização emergiu como uma das tecnologias-chave para a computação em nuvem, pois permite a um *datacenter* servir múltiplos usuários de forma segura, flexível e eficiente, principalmente devido à sua capacidade de isolamento, consolidação e migração de cargas de trabalho na forma de máquinas virtuais (VMs). Essas características possibilitam a criação de políticas para o gerenciamento da infraestrutura computacional, visando garantir a qualidade de serviço (QoS) e economia de recursos através do melhor uso da capacidade computacional disponível.

A alocação de máquinas virtuais (VMs) consiste em escolher o servidor físico (*host*) onde alocar cada VM, buscando otimizar alguma métrica como balanceamento

de carga, consumo de energia, etc. Para tal, deve-se conhecer a demanda de recursos de cada VM, o ambiente de execução e as políticas específicas do *datacenter* para possíveis conflitos [Hyser et al. 2007]. De modo geral, a alocação de uma VM é dividida em duas etapas: inicialmente é estimada a demanda de recursos da máquina virtual; em seguida, escolhe-se o servidor onde essa máquina virtual será alocada [Mishra and Sahoo 2011].

A demanda de recursos de uma VM pode ser difícil de estimar, pois as necessidades de recursos podem variar durante sua execução. Uma abordagem comum é de realizar a estimativa com base no histórico de utilização de recursos da VM. Este artigo considera que as necessidades de recursos de uma VM são previamente conhecidas. A alocação em si usa uma estratégia para definir onde alocar a VM de modo a alcançar um nível eficiente de utilização de recursos de um servidor físico. Tal estratégia pode utilizar representações matemáticas ou métricas que representam o grau de utilização dos recursos pelas diversas VMs e servidores físicos. Essas métricas devem levar em consideração as várias dimensões de consumo de recursos, como as capacidades de processador, memória, disco e banda de rede [Hyser et al. 2007], bem como recursos não envolvidos diretamente com as necessidades da máquina virtual, mas que afetam o *datacenter*, como o consumo de energia decorrente da alocação [Buyya et al. 2010].

Nesse artigo é apresentado um algoritmo para alocação de máquinas virtuais, onde são avaliados os recursos de CPU, memória, disco e banda de rede, que busca equilibrar o uso dos recursos nos servidores físicos. Como diferencial, o algoritmo proposto faz uso da capacidade de compartilhamento de páginas de memória presente em alguns hipervisores [Wood et al. 2009, Barker et al. 2012], buscando explorar o potencial de compartilhamento de memória que uma VM terá quando alocada em determinado servidor.

As seções 2 e 3 apresentam os fundamentos necessários à compreensão deste trabalho, a seção 4 apresenta o algoritmo de alocação e suas característica de funcionamento, a seção 5 apresenta o ambiente de simulação utilizado, os cenários testados e os resultados atingidos, e por fim a seção 6 apresenta as considerações finais do trabalho.

2. Máquinas virtuais

A virtualização, técnica surgida na década de 60 no IBM S/370 [Creasy 1981], permite dividir os recursos computacionais de um computador físico em partições autônomas e isoladas chamadas *máquinas virtuais* (VM). Uma máquina virtual funciona como uma cópia exata de uma máquina real, incluindo seus recursos de hardware. A virtualização é construída através de um *hipervisor*, uma camada de software de baixo nível que provê o suporte de execução necessário às máquinas virtuais. O hipervisor abstrai parcialmente os recursos de hardware subjacentes, tornando possível a execução simultânea de várias máquinas virtuais autônomas e isoladas sobre um mesmo computador.

Uma das principais funcionalidades do hipervisor consiste em abstrair os recursos da máquina real e oferecê-los às máquinas virtuais. Assim, cada máquina virtual "enxerga" sua própria interface de rede, seus discos rígidos, seus processadores e suas áreas de memória RAM. Esses recursos virtuais usados pelas máquinas virtuais são mapeados pelo hipervisor nos recursos reais presentes no hardware subjacente, de forma eficiente e controlada. Pode-se afirmar que o hipervisor constitui um "sistema operacional para sistemas operacionais", pela flexibilidade que introduz na gestão dos recursos de hardware. A possibilidade de instanciar máquinas virtuais sob demanda e a relativa independência

do hardware são essenciais para os ambientes de computação em nuvem.

A abstração de recursos construída pelo hipervisor abre a possibilidade de transferir uma máquina virtual em execução de um servidor físico para outro, mantendo seu estado interno. Esse procedimento, denominado *migração de máquinas virtuais*, deu muita flexibilidade à gerência de *datacenters* [Grit et al. 2006]. Migrar máquinas virtuais entre servidores físicos permite balancear a carga de trabalho entre eles ou mesmo concentrar as VMs em poucos servidores, hibernando os restantes e assim diminuindo o consumo de energia, de refrigeração e o ruído no ambiente [Buyya et al. 2010, Lago et al. 2011].

Recentemente, alguns hipervisores têm explorado a possibilidade de reduzir o consumo de memória RAM pelas máquinas virtuais através do compartilhamento de páginas de memória idênticas, como CBPS - *Content-Based Page Sharing*, disponível em hipervisores como VMWare ESX, Xen, e KSM - *Kernel Samepage Merging*, disponível no KVM [Barker et al. 2012]. Nessas técnicas, o hipervisor identifica páginas de memória com conteúdo idêntico e as compartilha usando a estratégia *copy-on-write*, de forma transparente para as máquinas virtuais: caso uma das máquinas virtuais tente alterar o conteúdo de uma página compartilhada, o compartilhamento é desfeito pelo hipervisor e uma cópia da página é alocada à máquina virtual solicitante. Essencialmente, existem duas categorias de compartilhamento: o auto-compartilhamento, quando as páginas duplicadas estão dentro da própria VM e o compartilhamento inter-VM, que ocorre entre máquinas virtuais distintas [Sindelar et al. 2011]. O compartilhamento de memória entre máquinas virtuais permite diminuir o consumo de memória RAM do servidor *host*, influenciando positivamente no desempenho do sistema.

3. Alocação de máquinas virtuais

Considerando um conjunto de servidores físicos em um *datacenter*, a alocação de VMs consiste em, dada uma VM, encontrar um servidor que seja adequado para instanciá-la. O servidor deve possuir recursos suficientes para garantir a execução da VM sem afetar os acordos de nível de serviço. Esse problema representa um desafio algorítmico por sua característica combinatória, onde um conjunto de n máquinas virtuais com requisitos distintos de processamento, memória e outros recursos deve ser alocado em m servidores com diferentes disponibilidades de recursos.

De forma geral, o problema de alocação de VMs pode ser visto como uma generalização do problema de múltiplas mochilas (*multiple knapsack problem - MKP*), onde cada mochila representa um servidor físico, sua capacidade corresponde aos recursos disponíveis no servidor e os itens são as VMs a serem alocadas. Cada VM tem um peso, representado pela quantidade de recursos necessários e um preço, que pode significar o custo que essa alocação representará no ambiente. Esse custo pode ser computacional, energético ou monetário. O objetivo é encontrar a melhor alocação das VMs em cada servidor, de modo a minimizar o custo [Grit et al. 2006]. Os algoritmos de alocação de VMs também podem se apresentar como variações do problema *Bin Packing*: dado um conjunto de n itens, onde cada item possui um peso w associado, e um conjunto de m recipientes (*bins*), cada um com uma capacidade c, encontrar o número mínimo de recipientes necessários para armazenar os itens tal que a soma dos itens em cada recipiente não ultrapasse sua capacidade. Comparando-se com o problema de alocação de VMs, tem-se os recipientes como sendo os servidores físicos, sua capacidade como os recursos

disponíveis no servidor e os itens seriam as VMs a alocar.

Na literatura há vários trabalhos relacionados à alocação de recursos virtuais em servidores físicos. Por se tratar de um problema combinacional do tipo *NP-difícil*, o assunto desperta o interesse de pesquisadores em busca de estratégias algorítmicas eficientes para solução desse tipo de problema [Singh et al. 2008].

O artigo [Grit et al. 2006] propôs estratégias para a chamada orquestração autônoma, onde o *datacenter* se auto-gerencia, independente de interferência humana. Nesse trabalho, os autores estabeleceram que a estratégia para alocação de máquinas virtuais pode estar presente nos processos de provisionamento de VMs, colocação de VMs nos servidores físicos e migração de VMs entre os servidores para consolidação e balanceamento de carga. Para [Hyser et al. 2007] uma boa colocação inicial não é suficiente, já que as cargas de trabalho das máquinas virtuais mudam com o passar do tempo. Consideram então ser necessário alterar dinamicamente a localização das máquinas virtuais de acordo com as condições do *datacenter*. Seu algoritmo considera os recursos de memória, processador, banda de rede e banda de disco *disk bandwidth* para estabelecer as máquinas virtuais que deverão ser migradas e realocadas para estabilizar o ambiente. Em outro trabalho, [Yen Van et al. 2009] criaram um módulo de decisão global para garantir a eficiência do sistema em termos de qualidade de serviço e gerenciamento dos custos dos recursos, desacoplando os processos de provisionamento e de alocação de VMs. Seus algoritmos levam em consideração as capacidades de recursos de processamento e de memória.

Dentro do contexto da computação em nuvem verde [Buyya et al. 2010] tratou do problema de alocação de máquinas virtuais objetivando o uso eficiente de recursos, de modo que a dimensão analisada como critério para alocação se baseia na análise do potencial consumo de energia do servidor físico após a alocação. O trabalho de [Chen et al. 2011] trata especificamente da consolidação de servidores, onde se objetiva liberar os servidores físicos que estejam subutilizados, migrando as máquinas virtuais e estabelecendo uma nova alocação. Sua abordagem, chamada *effective sizing* consiste em determinar as novas alocações através do menor número possível de migrações.

O trabalho [Wood et al. 2009] foi o primeiro a abordar o mecanismo de compartilhamento de páginas de memória entre máquinas virtuais, existente em alguns hipervisores, para estabelecer uma estratégia para alocação de máquinas virtuais. Sua solução baseia-se no cálculo de uma "impressão digital" para cada máquina virtual do sistema e através da comparação com a "impressão digital" gerada para a nova máquina virtual é estabelecido uma probabilidade de compartilhamento de páginas de memórias entre elas. O compartilhamento de páginas de memórias também é abordado no trabalho de [Sindelar et al. 2011], mas sua proposta baseia-se em dois modelos, que buscam estabelecer as maiores probabilidades de compartilhamento de memória de acordo com as características das máquinas virtuais. Essa nova abordagem torna mais eficiente o processo de alocação, tendo em vista que não são necessários o cálculo contínuo das "impressões digitais" de todas as VMs do sistema. No entanto, tais algoritmos consideram apenas a probabilidade de compartilhamento como fator de decisão para alocação, ignorando os demais recursos como processamento, memória, disco e banda de rede.

4. O Algoritmo *VectorAlloc*

Este trabalho propõe um algoritmo para a alocação *online* de máquinas virtuais nos servidores de um *datacenter*, considerando as demandas de recursos das máquinas virtuais, os recursos disponíveis nos servidores e o compartilhamento de memória entre as máquinas virtuais alocadas em um mesmo servidor. Nosso algoritmo, denominado *VectorAlloc*, foi inspirado no *VectorDot* [Singh et al. 2008], um algoritmo de balanceamento de carga (VMs) que busca minimizar o grau de desequilíbrio de carga dos servidores físicos, levando em conta os vários tipos de recursos em uma representação vetorial.

No algoritmo *VectorDot*, os servidores que estão acima de um certo limite de carga são considerados em "desequilíbrio"; calcula-se então o IBScore (*Imbalance Score*) de cada servidor e um *IBScoreTotal*, para determinar o nível de desequilíbrio global do sistema. O algoritmo então busca outros servidores para alocar as máquinas virtuais dos servidores sobrecarregados. O produto escalar entre a carga do servidor e as necessidades da máquina virtual determina a "atratividade" de um servidor. Além disso os vetores são ajustados para refletir os custos de migração das VMs.

Nosso algoritmo também expressa a utilização de recursos de cada servidor e as demandas das VMs como vetores, nos quais cada elemento corresponde a um tipo de recurso (memória, CPU, disco e rede) e seu valor corresponde ao grau de utilização do recurso em relação à sua capacidade total. Contudo, enquanto o *VectorDot* visa o balanceamento de carga, nosso algoritmo trata de alocações *online*, ou seja, sem um préconhecimento do conjunto de máquinas virtuais; a alocação ocorre à medida em que estas chegam. Além disso, nosso algoritmo considera a possibilidade de compartilhamento de memória entre máquinas virtuais.

O algoritmo *VectorAlloc* usa quatro vetores: *Resource Utilization Vector* $RUV(u)$, grau de utilização dos recursos de um servidor físico u; *Resource Requirement Vector* $RRV(u, v)$, recursos requeridos por uma VM v em relação a um servidor físico u; *Resource Threshold Vectors* $RTVmin$ e $RTVmax$, definem os limites mínimos e máximos de uso de cada um dos recursos em um servidor, em percentagem (notação adaptada de [Mishra and Sahoo 2011]). Esses vetores são expressos por:

$$
\begin{aligned}
RUV(u) &= \left[\frac{MemUse(u) - MemShared(u)}{MemCap(u)}, \frac{CPUUse(u)}{CPUCap(u)}, \frac{DiskUse(u)}{DiskCap(u)}, \frac{NetUse(u)}{NetCap(u)}\right] \\
RRV(u,v) &= \left[(1-\alpha(v,u))\frac{MemReq(v)}{MemCap(u)}, \frac{CPUReq(v)}{CPUCap(u)}, \frac{DiskReq(v)}{DiskCap(u)}, \frac{NetReq(v)}{NetCap(u)}\right] \\
RTVmin &= [MemTmin, CPUTmin, DiskTmin, NetTmin] \\
RTVmax &= [MemTmax, CPUTmax, DiskTmax, NetTmax]
\end{aligned}
$$

Cada elemento do vetor representa um tipo de recurso e possui um valor real no intervalo $[0, 1]$. Pela ordem, os recursos analisados são: memória, CPU, disco e rede. O elemento memória tem um tratamento diferenciado dos demais, pois deve considerar o compartilhamento de páginas entre máquinas virtuais.

$RUV(u)$ é calculado realizando-se o quociente entre os valores do recurso já alocado pela capacidade total do recurso. O elemento memória considera a quantidade de memória compartilhada entre as VMs. Por exemplo, um servidor físico que hospede duas máquinas virtuais vm_1 e vm_2, onde vm_1 demanda 100 MB de memória e vm_2 demanda 200

MB, utilizaria 300 MB de memória. Todavia, se 50 MB de memória estiverem compartilhados entre vm_1 e vm_2, o total de memória realmente utilizada será de 250 MB.

O vetor $RRV(u, v)$ representa a razão entre a demanda de recursos de uma VM v e os recursos disponíveis em um servidor físico u. O cálculo do recurso memória usa um *fator de compartilhamento* $\alpha(v, u)$. Este fator representa o potencial de compartilhamento de memória que uma VM v possui em relação ao servidor u. Por exemplo, uma VM que tenha um fator de compartilhamento de 20%, precisará alocar somente 80% de sua demanda de memória. O cálculo de α será discutido na próxima seção.

Os vetores $RTVmin$ e $RTVmax$ indicam os limites mínimos e máximos de uso dos recursos em cada servidor. Seu objetivo é garantir que a alocação de uma nova VM não desequilibre o uso de recursos, sobrecarregando ou subutilizando algum servidor. Assim, uma VM v só poderá ser alocada no servidor u se o uso de seus recursos se mantiver entre os limites inferior e superior, ou seja, se $RTVmin \leq RUV(u) + RRV(u, v) \leq RTVmax$.

Considerando o conjunto de servidores U do *datacenter* e uma máquina virtual v a ser alocada em um servidor $u \in U$, determina-se o conjunto $D(v, \gamma, \delta)$ de servidores disponíveis para alocar v como:

$$D(v, \gamma, \delta) = \{u \in U \mid \gamma \leq RUV(u) + RRV(u, v) \leq \delta\}$$

Com $\gamma = RTVmin$ e $\delta = RTVmax$, $D(v, \gamma, \delta)$ contém os servidores de U onde a alocação de v respeita os limites $RTVmin$ e $RTVmax$. Caso $D(v, \gamma, \delta) = \emptyset$, calcula-se $D(v, 0, \delta)$; caso este seja vazio, calcula-se $D(v, 0, 1)$; caso este também seja vazio, a alocação não é possível. Do conjunto $D(v) \neq \emptyset$ escolhe-se um servidor u_a que satisfaça:

$$u_a = \arg\min_{u \in D(v)} (RUV(u) \cdot RRV(u, v))$$

Em outras palavras, a máquina virtual v será alocada no servidor $u_a \in D(v)$ com o menor produto escalar entre $RUV(u)$ e $RRV(u, v)$, buscando respeitar os limites de alocação. A ideia subjacente é alocar a VM em um servidor para o qual RRV seja complementar a RUV, resultando em um produto escalar mínimo, o que induz um uso balanceado dos recursos. Por exemplo, uma VM com baixa demanda de CPU e alta demanda de memória será alocada em um servidor com alto uso de CPU mas baixo uso de memória.

4.1. O Fator de Compartilhamento $\alpha(v, u)$

O fator de compartilhamento $\alpha(v, u)$ é usado pelo algoritmo proposto para estabelecer o potencial de compartilhamento de memória que uma VM v possui em relação a um servidor u. É um valor que varia entre 0 e 1, sendo 0 quando não há nenhuma chance de v compartilhar memória com as demais VMs presentes em u, e 1 caso toda a memória de v possa ser compartilhada.

O valor de α corresponde a uma estimativa, já que a memória efetivamente compartilhada só poderá ser determinada após a alocação da VM. Este trabalho não se aprofunda no cálculo preciso de α, pois foge do seu escopo. Aqui, inclusive, abre-se uma possibilidade de pesquisa para o estabelecimento de métodos para o cálculo de α, que poderiam fazer uso do histórico de alocações para determinar índices mais próximos da realidade. Alguns trabalhos como [Wood et al. 2009] e [Sindelar et al. 2011] apresentaram formas de

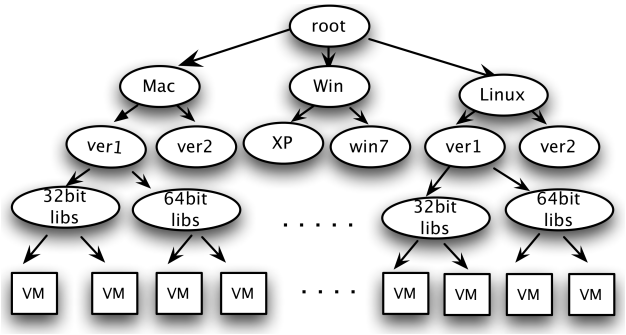


Figura 1. Modelo hierárquico em árvore [Sindelar et al. 2011]

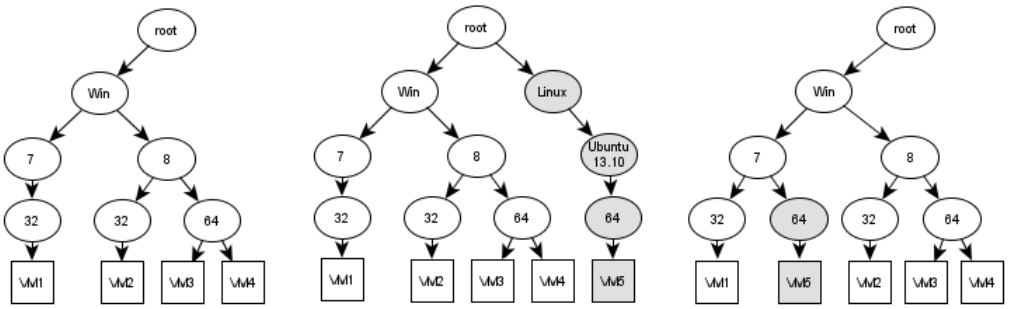
capturar a memória compartilhada entre as máquinas virtuais e utilizaram tal informação para criação de algoritmos de alocação, otimização e balanceamento de carga.

Neste trabalho, o fator de compartilhamento é calculado usando uma representação hierárquica de máquinas virtuais, similar à apresentada em [Sindelar et al. 2011]. Cada servidor referencia as máquinas virtuais nele alocadas usando uma árvore similar à apresentada na Figura 1. Nessa árvore, cada nível representa um novo grau de especialização das informações das VMs. Partindo da raiz, o próximo nível representa o sistema operacional (SO), em seguida vem a versão do SO e então a arquitetura do SO (32 ou 64 bits). As folhas da árvore representam as máquinas virtuais alocadas no servidor.

[Sindelar et al. 2011] utiliza essa árvore para capturar as páginas de memória das VMs alocadas. Em seu modelo, a raiz da árvore contém todas as páginas de memória compartilhadas entre as VMs, os nós no nível do SO contêm as páginas compartilhadas pelo mesmo SO e o mesmo ocorre nos nós para a versão do SO e para a arquitetura do SO. As folhas contêm as páginas de memória que não são compartilhadas, ou seja, são exclusivas de cada VM. Nosso trabalho adotou uma interpretação diversa: cada nível indica o potencial de compartilhamento para uma dada VM. Conforme aumenta o grau de similaridade entre as VMs, também aumenta a probabilidade dessas VMs compartilharem páginas de memória comuns entre si. Dessa forma, ao realizar-se uma busca na árvore por VMs semelhantes a uma dada VM, quanto maior a profundidade alcançada, maior será a probabilidade de compartilhamento de memória.

A Figura 2(a) ilustra um exemplo da árvore de alocação em um servidor físico u. Nesse exemplo existem quatro VMs alocadas: vm_1 é uma máquina Windows 7 32 bits, vm_2 é Windows 8 32 bits e vm_3 e vm_4 são Windows 8 64 bits. Supondo que uma VM Linux Ubuntu 13.10, 64 bits (vm_5) deva ser alocada em u. Como vm_5 não é similar a nenhum nó da árvore, $\alpha(vm_5, u) = 0$. Se mesmo assim vm_5 for alocada em u, uma nova ramificação será criada na árvore (Figura 2(b)). Caso a vm_5 fosse Windows 7 64 bits, seriam encontrados dois níveis de similaridade (SO e versão), portanto $\alpha(vm_5, u) > 0$. Caso vm_5 seja alocada em u, sua árvore ficaria conforme a Figura 2(c).

Para a realização da simulação, foram estabelecidos pesos fixos de 0,1 para cada nível hierárquico da árvore. O fator α é calculado somando-se esse peso até o último nível comum entre as VMs. Assim, uma VM que tenha dois níveis de similaridade com outras terá $\alpha = 0,2$. O valor de 0,1 por nível foi escolhido de forma empírica, apenas para avaliar o algoritmo proposto; em servidores reais o compartilhamento de memória é muito variável, sofrendo oscilações durante o ciclo de vida das VMs.



(a) Cenário original

(b) Alocação de uma VM Ubuntu 13.10 64 bits

(c) Alocação de uma VM Windows 7 64 bits

Figura 2. Exemplo de árvore de alocação em um servidor

5. Experimentos Realizados

Para a validação do algoritmo proposto optou-se pela realização de experimentos em um ambiente de simulação. Alguns fatores que contribuíram para essa escolha incluem a dificuldade de acesso a um ambiente de larga escala real, a dificuldade em generalizar resultados de testes em ambientes de pequena escala para ambientes maiores, a dificuldade de reprodução de testes em ambientes reais decorrente do caráter dinâmico do problema e, não menos importante, a existência de muitos trabalhos de pesquisa que usam o mesmo simulador, atestando a qualidade de seus resultados e os benefícios de sua utilização.

Esta seção apresenta o ambiente de simulação utilizado para os experimentos, os ajustes necessários para adaptá-lo ao algoritmo proposto, os experimentos e cenários testados e os resultados obtidos.

5.1. Ambiente de Simulação

Para a avaliação do algoritmo foi usado o simulador *CloudSim* [Calheiros et al. 2011], um *framework* extensível feito em *Java* que permite a modelagem e simulação de um ambiente de computação em nuvem. O *CloudSim* foi desenvolvido pelo *The Cloud Computing and Distributed Systems (CLOUDS) Laboratory* da Universidade de Melbourne, Austrália. Com o *CloudSim* é possível modelar vários aspectos do funcionamento de uma nuvem, como a configuração de um *datacenter*, nuvens federadas, cargas de trabalho dinâmicas, consumo de energia e políticas de alocação de máquinas virtuais.

O *CloudSim* apresenta uma arquitetura multi-camadas: na base fica a camada de simulação, que oferece suporte para modelagem e simulação de ambientes virtualizados, inclui interfaces para gerenciamento de máquinas virtuais, memória, armazenamento e largura de banda. Dispõe de recursos para o gerenciamento de aplicações e monitoramento dinâmico do estado da nuvem. Nessa camada são estabelecidas as políticas de provisionamento e alocação de VMs. A camada do topo representa o código do usuário, que expõe as entidades básicas para definição dos *hosts* (número de máquinas e suas especificações), aplicações (número de tarefas e seus requisitos), máquinas virtuais, número de usuários e políticas de escalonamento. É nessa camada que são construídos os cenários de simulação e onde os experimentos são realizados.

Para que o *CloudSim* atendesse os requisitos do algoritmo proposto, foram necessárias algumas modificações no *framework*. A Figura 3 apresenta parte do diagrama de

classes do *framework*, sendo visíveis apenas as classes de maior relevância; as classes em fundo cinza foram incluídas ao *framework* neste trabalho, para dar suporte à simulação do compartilhamento de memória.

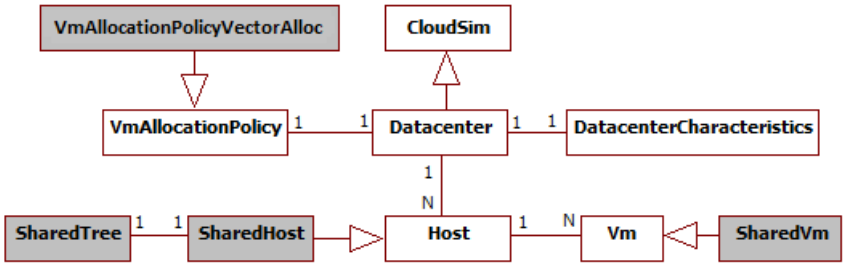


Figura 3. Diagrama simplificado de classes do *CloudSim*, com modificações

As classes *Host* e *Vm*, que modelam, respectivamente, as características de um servidor físico e as características de uma máquina virtual, foram estendidas para *SharedHost* e *SharedVm*, incluindo as informações necessárias para o tratamento da memória compartilhada. Foi criada uma nova classe chamada *SharedTree*, que modela a árvore de alocações de VMs em um *SharedHost*. Através dessa estrutura é possível calcular o fator de compartilhamento α para cada servidor físico. Por fim, a classe *VmAllocationPolicyVectorAlloc* foi estendida de *VmAllocationPolicy*; ela contém a política de alocação e distribuição das máquinas virtuais nos servidores físicos. Ao iniciar a simulação, essa classe é a responsável por receber cada VM e encontrar o melhor servidor para alocá-la.

5.2. Experimentos

Para a realização dos experimentos foram criados três cenários distintos, simulando ambientes de *datacenter* de diferentes portes (inpirado de [Lago et al. 2011]). O *datacenter* de pequeno porte contém 10 servidores onde devem ser alocadas 30 máquinas virtuais. De médio porte contém 100 servidores para alocação de 300 máquinas virtuais e o *datacenter* de grande porte com 1000 servidores para 3000 máquinas virtuais.

As configurações de CPU, banda de rede e armazenamento nos servidores foram fixadas em 10.000 MIPS, 1 Gbps e 1 TB. Para a memória adotou-se uma distribuição circular do conjunto {12, 16, 20 e 24 GB}, simulando servidores com diferentes capacidades. Para as máquinas virtuais, as configurações de banda de rede e de armazenamento ficaram fixas em 100 Mbps e 5 GB, com distribuição circular para a CPU em {1.000, 1.500, 2.000 e 2.500 MIPS e para a memória em 512 MB, 1, 2, 4 e 8 GB. O sistema operacional de cada VM pode ser Windows 7, Windows 8, Ubuntu 10.4 ou Ubuntu 12.4, todos podendo assumir a plataforma de 32 ou 64 bits. Adotou-se $RTVmin = 0,1$ e $RTVmax = 0,9$.

Para cada cenário foram executados três algoritmos de alocação: a) um algoritmo de alocação usando a técnica *First-Fit*, onde uma VM é simplesmente alocada no primeiro servidor que tiver recursos suficientes disponíveis; b) o algoritmo *VectorAlloc* sem o compartilhamento de memória, ou seja, com $\alpha = 0$; e c) o algoritmo *VectorAlloc* considerando o compartilhamento de memória. A alocação das VMs ocorre de forma *online*, ou seja, não há um pré-conhecimento do conjunto a ser instanciado. As VMs são alocadas em sequência, à medida que entram na fila de alocação. Isto se torna interessante para garantir a eficiência de alocação em sistemas onde os usuários chegam continuamente [Zaman and Grosu 2012].

5.3. Resultados

Para medir a eficiência do algoritmo foi coletada uma série de dados das simulações realizadas. Os dados apresentam o estado dos recursos físicos de cada servidor após a realização das alocações. A partir desses dados foi possível realizar algumas análises: pelo *grau de utilização dos recursos*, foi observado se a alocação pelo algoritmo resulta em uma melhor distribuição do uso dos recursos físicos dos servidores, ou seja, se as VMs foram distribuídas de modo a não criarem uma situação de excesso de carga ou de subutilização de recursos. O *gap*[1] entre o uso de memória e de CPU permite avaliar se esses recursos estão sendo alocados de forma equilibrada, não sobrecarregando um recurso do servidor enquanto o outro estaria subutilizado; um *gap* elevado pode fazer com que servidores com grande disponibilidade de um recurso não possam alocar uma nova VM devido à limitação de outro recurso. Finalmente, a *Memória economizada* permite avaliar quanta memória pode ser poupada pelo algoritmo proposto e se essa economia impactou no uso dos outros recursos, em especial a CPU.

A Figura 4 apresenta o grau de utilização da memória após a execução de cada algoritmo nos três cenários. O eixo X apresenta os intervalos do percentual de utilização do recurso e o eixo Y a frequência com que o intervalo ocorre.

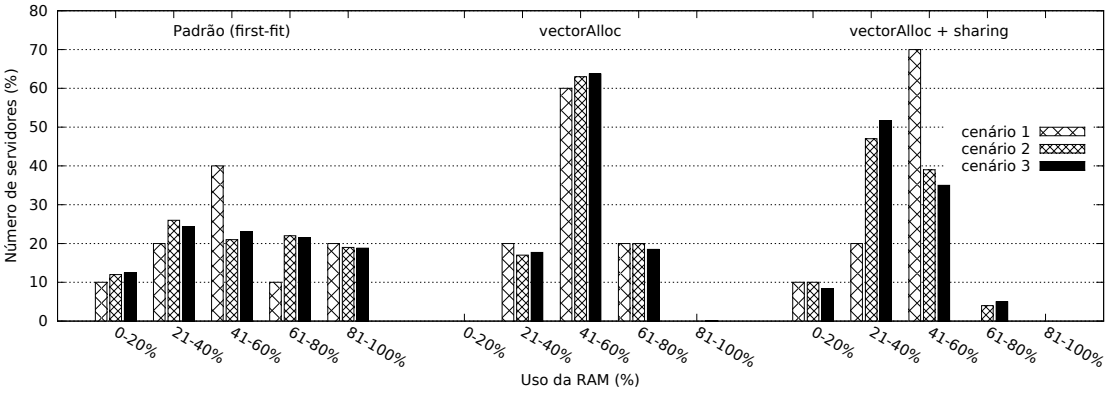


Figura 4. Grau de utilização da memória após alocações

É possível perceber que no algoritmo padrão o uso da memória está desequilibrado, atingindo inclusive extremos de uso: cerca de 10% dos servidores tiveram um uso de memória abaixo de 20%, enquanto outros 20% tiveram seu uso de memória entre 81% e 100%. No *VectorAlloc* tal situação não ocorre e o uso do recurso está equilibrado. Na execução sem compartilhamento de memória, todos os cenários apontaram que 60% dos servidores tiveram um grau de utilização entre 40% e 60%. O restante ficou dividido entre os intervalos vizinhos. Não houveram casos de utilização abaixo de 20% nem acima de 80%. Na execução com compartilhamento de memória os servidores em geral tiveram a utilização dos recursos reduzida. Os cenários apresentaram uma redução dos intervalos mais altos e aumento dos intervalos mais baixos, inclusive com utilização de memória abaixo de 20%, mas sem casos de utilização de recursos acima de 80%.

Em relação ao *gap* existente entre a utilização de recursos de memória e de CPU, novamente o algoritmo *VectorAlloc* mostra-se mais equilibrado. A Figura 5 ilustra essa

[1]Define-se como *gap* o módulo da diferença entre os níveis de uso de dois recursos.

situação. No algoritmo padrão, entre 70% e 80% das alocações criaram um *gap* de até 40%, mas nos ambientes de médio e grande porte, quase 10% dos servidores atingiram uma diferença entre 61% e 80%. Percebe-se que nessas situações um recurso está sendo muito sobrecarregado em relação ao outro. O *VectorAlloc* sem compartilhamento reduziu muito essa diferença, deixando em torno de 90% dos servidores com um *gap* inferior a 20%, o que indica uso equilibrado dos recursos. Com o compartilhamento de memória esse *gap* volta a ter um aumento, mas isso é explicado pelo fato do uso de CPU se manter o mesmo, enquanto o uso de memória foi reduzido pela possibilidade de compartilhamento.

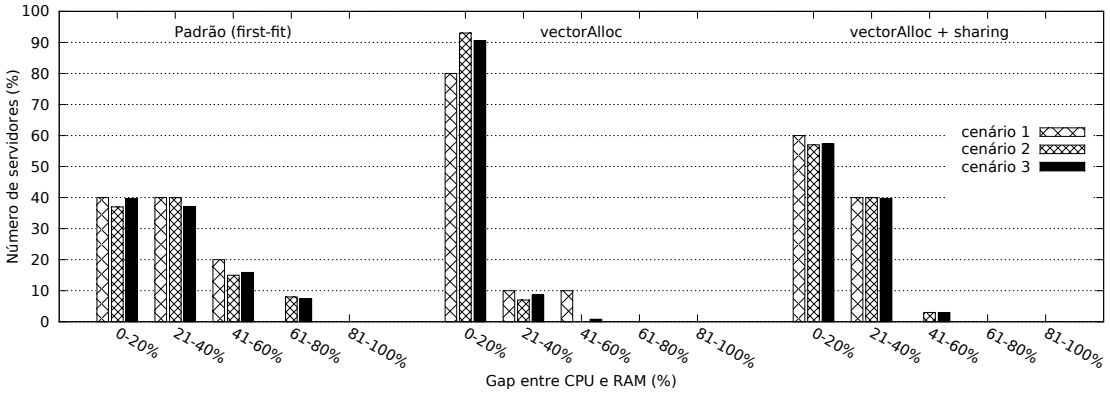


Figura 5. Gap entre uso da memória e de CPU

Os últimos resultados obtidos aparecem na Tabela 1 e mostram o uso médio de memória nos servidores, para cada algoritmo executado. Dos três algoritmos, o *VectorAlloc* com compartilhamento de memória gerou os melhores resultados, resultando numa média de utilização de memória em torno de 38%, cerca de 15% menor que o algoritmo padrão e 12,5% menor que o *VectorAlloc* sem compartilhamento de memória (coluna *ganho*).

Tabela 1. Média e desvio padrão de uso da memória dos servidores

cenário	*First-Fit*		*VectorAlloc*			*VectorAlloc + mem sharing*		
	média	desvio	média	desvio	ganho	média	desvio	ganho
1	58,7%	27,9%	53,5%	12,0%	5.2%	43,9%	12,9%	14.8%
2	53,5%	27,2%	50,7%	12,6%	2.8%	38,2%	13,3%	15.3%
3	53,6%	27,1%	50,8%	12,4%	2.8%	38,0%	13,3%	15.6%

6. Conclusão

Este artigo apresentou o algoritmo *VectorAlloc* para alocação *online* de máquinas virtuais em ambientes de computação em nuvem. O algoritmo faz uso de uma representação vetorial para a análise dos múltiplos recursos físicos e leva em consideração o compartilhamento de memória entre máquinas virtuais. Comparado a uma abordagem padrão de alocação, os resultados dos testes comprovaram que o algoritmo faz um melhor uso dos recursos físicos dos servidores, distribuindo as máquinas virtuais de uma modo mais equilibrado, eliminando cenários de sobrecarga ou subutilização. Nos testes considerando a possibilidade de compartilhamento de memória, o consumo médio de memória dos servidores foi reduzido em cerca de 15% em relação à abordagem padrão.

Referências

Barker, S., Wood, T., Shenoy, P., and Sitaraman, R. (2012). An empirical study of memory sharing in virtual machines. In *Usenix Annual Technical Conference*.

Buyya, R., Beloglazov, A., and Abawajy, J. (2010). Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges. *arXiv preprint arXiv:1006.0308*, pages 1–12.

Calheiros, R., Ranjan, R., Beloglazov, A., De Rose, C., and Buyya, R. (2011). CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50.

Chen, M., Zhang, H., Su, Y.-Y., Wang, X., Jiang, G., and Yoshihira, K. (2011). Effective VM sizing in virtualized data centers. In *IFIP/IEEE International Symposium on Integrated Network Management*, pages 594–601.

Creasy, R. J. (1981). The origin of the VM/370 time-sharing system. *IBM Journal of Research and Development*, 25(5):483–490.

Grit, L., Irwin, D., Yumerefendi, A., and Chase, J. (2006). Virtual machine hosting for networked clusters: Building the foundations for autonomic orchestration. In *1st International Workshop on Virtualization Technology in Distributed Computing*. IEEE.

Hyser, C., Mckee, B., Gardner, R., and Watson, B. (2007). Autonomic virtual machine placement in the data center. Technical Report HPL-2007-189, HP Labs, Palo Alto CA.

Lago, D., Madeira, E., and Bittencourt, L. (2011). Power-aware virtual machine scheduling on clouds using active cooling control and DVFS. In *9th International Workshop on Middleware for Grids, Clouds and e-Science*, New York USA. ACM Press.

Mishra, M. and Sahoo, A. (2011). On theory of VM placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach. In *IEEE 4th International Conference on Cloud Computing*, pages 275–282. IEEE.

Sindelar, M., Sitaraman, R. K., and Shenoy, P. (2011). Sharing-aware algorithms for virtual machine colocation. In *23rd ACM Symposium on Parallelism in Algorithms and Architectures*, page 367, New York USA. ACM Press.

Singh, A., Korupolu, M., and Mohapatra, D. (2008). Server-storage virtualization: Integration and load balancing in data centers. In *ACM/IEEE Conference on Supercomputing*, pages 1–12. IEEE Press.

Wood, T., Tarasuk-Levin, G., Shenoy, P., Desnoyers, P., Cecchet, E., and Corner, M. D. (2009). Memory buddies: Exploiting page sharing for smart colocation in virtualized data centers. *Operating Systems Review*, 43(3):27–36.

Yen Van, H., Dang Tran, F., and Menaud, J.-M. (2009). Autonomic virtual resource management for service hosting platforms. In *ICSE Workshop on Software Engineering Challenges of Cloud Computing*, pages 1–8, Washington DC. IEEE Computer Society.

Zaman, S. and Grosu, D. (2012). An online mechanism for dynamic VM provisioning and allocation in clouds. In *5th IEEE International Conference on Cloud Computing*, pages 253–260. IEEE.