Políticas para Seleção Assistida de Máquinas Virtuais em Nuvem de Computadores

Mario Meireles Teixeira[1], **Azer Bestavros**[2]

[1]Departamento de Informática – Universidade Federal do Maranhão
65080-805 – São Luís, MA – Brasil

[2]Computer Science Department – Boston University
Boston, MA 02215 – USA

mario@deinf.ufma.br, best@bu.edu

Abstract. *In this paper we propose an end-to-end approach to the VM allocation problem using policies based uniquely on round-trip time measurements between VMs. We propose and implement end-to-end algorithms for VM selection that cover desirable profiles of communications between VMs in distributed applications in a cloud setting. The use of informed VM selection allowed to select instances with an average RTT up to 85% lower, in one case, and to find local and global centers of VM clusters in another. The approach described is completely independent of cloud architecture and is adaptable to different types of applications and workloads, as well as light and transparent for cloud providers.*

Resumo. *Este artigo propõe uma abordagem fim a fim para o problema de alocação de VMs em nuvens de computadores, usando políticas baseadas exclusivamente em medições de tempo de ida e volta entre VMs. São propostos algoritmos fim a fim para seleção de VMs que consideram perfis típicos de comunicação entre VMs executando aplicações distribuídas em nuvem. O uso da seleção assistida de VMs permitiu selecionar instâncias com um RTT médio até 85% menor, em um caso, bem como localizar centros locais e globais de aglomerados de VMs, em outro. A abordagem descrita é completamente independente da arquitetura da nuvem e é adaptável a diferentes tipos de aplicações e cargas de trabalho, além de leve e transparente para os provedores de nuvem.*

1. Introdução

Em cenários atuais de utilização de nuvem, na modalidade IaaS, um cliente solicita e recebe do provedor de nuvem um conjunto de máquinas virtuais (VMs) com capacidade determinada, que devem ser organizadas de forma a proporcionar os melhores serviços possíveis com o mínimo consumo de recursos e, portanto, com o menor custo para o cliente.

Quando os clientes de nuvem solicitam um conjunto de VMs de um provedor de nuvem pública, como a Amazon EC2[1], eles geralmente não têm qualquer informação sobre em qual máquina física cada VM está de fato localizada e nem sobre como esses recursos físicos compartilhados são utilizados por outros clientes da nuvem, em termos

[1]http://aws.amazon.com/

de consumo de CPU, dados ou rede. Além disso, os clientes não sabem como as VMs que lhes foram atribuídas relacionam-se quanto ao atraso dentro da rede da nuvem.

Neste contexto, este trabalho propõe um *Serviço de Recomendação* para ser usado por clientes de nuvem que desejam fazer uma melhor utilização dos recursos por eles contratados, assistindo-lhes em suas decisões de alocação de VMs, de modo a não desperdiçar nem abusar desses recursos. Essas VMs seriam selecionadas a partir de um conjunto maior de VMs atribuído pelo provedor de nuvem ao cliente, ou a um agregador ou *broker* operando em nome de um conjunto de clientes [Bestavros and Krieger 2014].

O foco principal não está na alocação de VMs sob a perspectiva do administrador de nuvem, pois este problema já foi bastante estudado recentemente [Alicherry and Lakshman 2012] [Meng et al. 2010] [Piao and Yan 2010]. Em vez disso, nosso interesse particular é na capacitação dos clientes da nuvem, que têm utilizado seus recursos alocados como uma caixa preta, na esperança de que os provedores de nuvem saibam o que é 'melhor' para eles.

Outra importante motivação resulta do fato que nuvens não são mais apenas infraestruturas em que se executam aplicações de *Big Dat*a ou de computação de alto desempenho. Cada vez mais, a nuvem hospeda aplicações que exigem comunicação intensiva, tais como jogos, serviços multimídia e aplicações de tempo real, para as quais problemas de latência (e variação desta) são de suma importância. Problemas de latência de comunicação podem alongar os tempos de conclusão de tarefas na nuvem [Alicherry and Lakshman 2012], prejudicando a qualidade dos serviços prestados. Além disso, pesquisas recentes têm mostrado que os padrões de intercomunicação entre máquinas virtuais podem desempenhar um papel importante e decisivo na eficiência das aplicações [LaCurts et al. De 2013] e, definitivamente, ter um impacto sobre seu desempenho e capacidade de resposta percebidos pelos usuários finais.

Mais especificamente, este trabalho trata sobre algoritmos de alocação de VMs que empregam tempos de ida e volta (RTT) como heurística para auxiliar a alocação de VMs em nuvens IaaS. Medições em tempo real de atrasos entre máquinas virtuais são usadas como entrada para o Serviço de Recomendação, que tem a responsabilidade de selecionar um subconjunto m de VMs, dentre as n possíveis, sujeito a certos requisitos colocados pelo cliente da nuvem.

A abordagem empregada é fim a fim e independente da topologia interna da nuvem. Esta é uma inovação e também um diferencial em relação a pesquisas semelhantes [Meng et al. 2010] [Alicherry and Lakshman 2012], que geralmente empregam algum conhecimento da arquitetura subjacente do *datacenter* com este propósito. Argumenta-se que esta abordagem é consistente com modelos de negócio emergentes em nuvem e inter-nuvens, como no caso da iniciativa *Massachusetts Open Cloud* [MOC 2014].

Este artigo está assim organizado: a Seção 2 discute o cenário atual de utilização de nuvens e descreve o Serviço de Recomendação. A Seção 3 apresenta as políticas de seleção de VMs e os algoritmos propostos. Na seção 4, os algoritmos e seus resultados são avaliados através de simulação, utilizando-se uma arquitetura de referência validada a partir de medições reais na nuvem Amazon EC2. Por fim, a Seção 5 discute as principais conclusões e trabalhos futuros.

2. Arquitetura do Sistema

2.1. Cenário

Clientes de serviços de infraestrutura de computação em nuvem (IaaS) normalmente acessam um provedor de nuvem pública através de uma interface web e requisitam máquinas virtuais genéricas, que pretendem empregar em diferentes tipos de tarefas computacionais. A ideia é que os recursos disponíveis sejam encarados de forma utilitária (como se fossem energia, água, etc.), pois os clientes não têm interesse em um aplicativo em nuvem específico, mas apenas em ter acesso a uma plataforma de computação remota para finalidades as mais diversas.

Muitos provedores de nuvem disponibilizam algum tipo de serviço de monitoramento para seus clientes, através do qual estes podem obter informações de alto nível e/ou refinadas sobre como suas aplicações estão usando as VMs que lhes foram atribuídas, em termos de consumo de CPU, dados e rede. No entanto, estes serviços de monitoramento não fornecem qualquer visibilidade sobre o impacto de aplicativos de terceiros executando paralelamente em outras VMs, seja na mesma máquina física ou em outro lugar da nuvem. A falta desta informação pode conduzir a mapeamentos de aplicações para VMs completamente inadequados. Por exemplo, caso se deseje selecionar uma VM para ser um servidor, é interessante que esta se localize em uma posição equidistante dos seus clientes dentro da nuvem ou, ainda, pode-se querer que o tempo de obtenção da informação seja inferior a um certo limiar. Numa seleção aleatória de VMs, fica difícil senão impossível atender a estes requisitos, o que é perfeitamente viável com a aplicação da seleção assistida proposta.

O objetivo é fornecer um serviço que irá coletar informações de uso da nuvem, de um lado e associá-las com as necessidades do cliente, de outro, a fim de sugerir um mapeamento de VMs em conformidade com os requisitos colocados e adequado à utilização atual dos recursos da nuvem. Esse serviço seria especialmente útil em modelos de negócios emergentes na computação em nuvem, como mostrado na Figura 1, onde se veem corretores (*brokers*) que contratam um lote de recursos do provedor de nuvem e atuam como revendedores (ou intermediários) entre o provedor de nuvem e o cliente de nuvem [Bestavros and Krieger 2014] [Böhm et al. 2010].

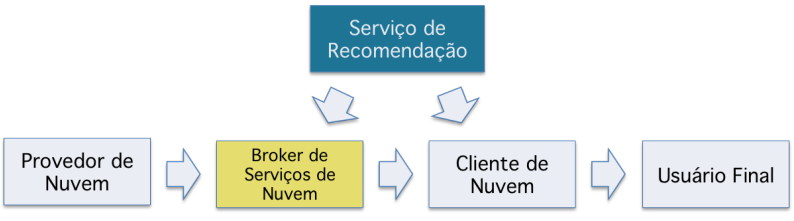

Figura 1. Modelos de negócio emergentes em Computação em Nuvem

2.2. Serviço de Recomendação

Este trabalho propõe um **Serviço de Recomendação**, como mostrado na Figura 2, ao qual os clientes da nuvem fazem uma solicitação de VMs que deve atender a requisitos específicos (1), como um limite mínimo de tempo de ida e volta (*Round-trip time* – RTT) entre as VMs ou uma topologia particular de VMs dentro da nuvem.

O Serviço de Recomendação mantém um conjunto de n VMs anteriormente contratadas a partir de um provedor de nuvem e periodicamente monitora seu desempenho (2). Quando um pedido chega, este serviço irá selecionar um subconjunto de m VMs das n totais que deve satisfazer as demandas do cliente. Em seguida, sugere este arranjo de VMs ao solicitante (3), que por sua vez distribui suas aplicações nas VMs. Consequentemente, o cliente pode escolher entre os recursos disponíveis com mais certeza, realizando a atribuição de suas aplicações às VMs de forma mais esclarecida.

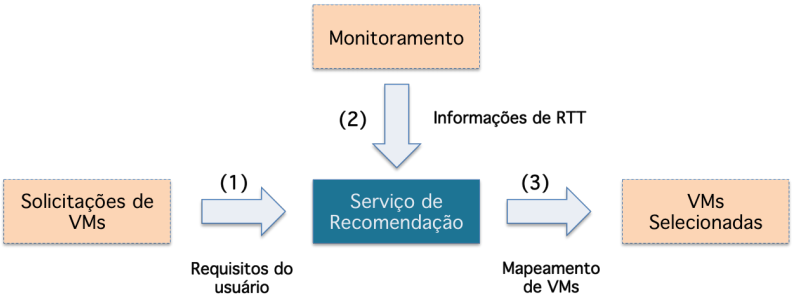

Figura 2. Arquitetura do Serviço de Recomendação

O Recomendador não funciona como um *broker*, uma vez que ele não realiza alocação de VMs. O que ele realmente faz é monitorar o ambiente ao redor de cada VM (por exemplo, a latência de rede ou *throughput*), a fim de ser capaz de melhor balancear as solicitações de seus clientes frente aos recursos disponíveis, proporcionando uma seleção de VMs adequada às suas necessidades.

É nossa percepção que este tipo de serviço pode se tornar uma peça central na gestão de recursos de nuvem em um futuro próximo, na medida em que os aplicativos tornam-se mais e mais complexos e com exigências mais rigorosas, que não podem ser satisfeitas unicamente pelos modelos de negócio atuais do tipo provedor-a-cliente.

Uma parte fundamental do Serviço de Recomendação são os algoritmos de seleção de VMs e também as informações de desempenho disponíveis para eles. O presente estudo detém-se particularmente em algoritmos que procuram tirar proveito das informações de latência fim a fim entre VMs para suas decisões de alocação.

3. Algoritmos de Seleção de VMs

Esta seção detalha a abordagem para monitoramento de latência fim a fim entre VMs e propõe 4 algoritmos para Seleção Assistida de máquinas virtuais, os quais usam algum tipo de informação de atraso para as suas sugestões de alocação.

3.1. Matriz de Latências

A entrada principal para os algoritmos consiste em uma *Matriz de Latências* ($RTT_{i,j}$), que é atualizada periodicamente com informações de tempo de ida e volta (RTT) entre qualquer par (i, j) de VMs conhecido (ou gerenciado) pelo Serviço de Recomendação:

$$RTT_{i,j} = \begin{bmatrix} 0 & t_{1,2} & t_{1,3} & \dots & t_{1,n} \\ t_{2,1} & 0 & t_{2,3} & \dots & t_{2,n} \\ t_{3,1} & t_{3,2} & 0 & \dots & t_{3,n} \\ \dots & \dots & \dots & \dots & \dots \\ t_{n,1} & t_{n,2} & t_{n,3} & \dots & 0 \end{bmatrix} \quad (1)$$

Cada linha ou coluna na matriz representa uma VM dentro da nuvem e o elemento (i, j) contém o tempo de ida e volta entre a VM$_i$ e a VM$_j$. Como esta informação é simétrica, o elemento (j, i) equivalente é ignorado e os valores apresentados na coluna principal da matriz são ajustados em zero. Para fins de implementação, assume-se uma matriz triangular superior.

Optou-se por usar a latência como base para os algoritmos de seleção de VMs porque o RTT é uma métrica confiável para medir a distância de rede dentro de nuvens, como apontado por trabalhos recentes [Alicherry and Lakshman 2012] [Battré et al. 2011] [Jonglez et al. 2014] [LaCurts et al. 2013] [Shea et al. 2014]. Além disso, RTTs são computacionalmente baratos e simples de usar e obter. Neste estudo, medições de latência serão utilizadas para estimar a distância entre VMs e constituir-se-ão em um parâmetro fundamental para as recomendações de alocação de VMs.

Para verificar como isso funcionaria num cenário real, foi realizado um experimento de monitoramento de tempos de ida e volta utilizando-se instâncias Amazon EC2. Primeiramente, colocou-se um aplicativo do tipo PING, cliente-servidor, em cada uma das VMs contratadas. O cliente PING acorda periodicamente e envia uma mensagem a cada uma das outras máquinas virtuais, em *round robin*. Este passo é repetido várias vezes (definido como 10 nos experimentos); quando o cliente recebe as mensagens de retorno, calcula-se o atraso médio para cada par (VM$_i$, VM$_j$) e atualizam-se os valores na Matriz de Latências.

3.2. Algoritmo Colmeia (*Hive*)

Este primeiro algoritmo tem como objetivo encontrar um conjunto de VMs próximas umas das outras por um tempo de ida e volta menor do que r ms, ou seja, o RTT entre qualquer par de VMs pertencente a esse conjunto deve ser inferior a r. Uma variação disto seria exigir que o percentil 0.95 das medições dos RTT_{ij} fosse inferior ao limite r, a fim de evitar que um arranjo adequado de VMs fosse descartado por causa de alguns valores discrepantes.

Esta abordagem é denominada de *Algoritmo Colmeia*, uma vez que tem a intenção de encontrar aglomerados de VMs próximas em uma tentativa de explorar localidade de referência. Na verdade, a maioria dos aplicativos na nuvem pode se beneficiar de tal abordagem, pois ela minimiza o tráfego intra-nuvem e como resultado reduz o uso de banda, além de ajudar a diminuir a fragmentação do *datacenter*, uma vez que as VMs cooperando em uma tarefa tendem a ficar próximas entre si. No entanto, observe que latência, neste caso, está em geral mas não necessariamente relacionada com distância física na rede. Apesar de VMs no mesmo processador ou *rack* tenderem a ter valores de RTT menores entre elas, em alguns casos isso pode não ser verdade devido a questões de virtualização ou de carga de trabalho. Por conseguinte, pode ser que uma VM em outro *rack* venha a ser o vizinho mais próximo.

O Algoritmo Colmeia é muito difícil de se implementar em termos práticos. Ele é de fato um caso especial de um problema NP-difícil em grafos. O que se busca encontrar é um subgrafo onde só é possível desenhar uma aresta de qualquer nó i para qualquer nó j se o RTT entre eles for menor do que o limite especificado. Quanto maior o número m de VMs na colmeia, maior a complexidade do algoritmo. Se a intenção for encontrar a maior colmeia possível dado que $RTT_{ij} < distance$ então ter-se-ia o problema MAXCLIQUE, onde o desafio é encontrar um clique de tamanho máximo, o que é provado NP-difícil.

Uma maneira de resolver este problema é restringir o número de VMs na colmeia para um valor muito pequeno. Na verdade, na Seção 3.5 é apresentado um exemplo de uma abordagem simples para encontrar m pares de VMs dentro da nuvem com um RTT entre elas menor do que um certo limite.

3.3. Algoritmo Estrela (*Star*)

Nossa segunda abordagem, o *Algoritmo Estrela* procura encontrar a resposta para uma pergunta simples: dado um conjunto de VMs dentro da nuvem, deseja-se saber qual VM encontra-se aproximadamente no centro do conjunto, considerando-se a latência. Em outras palavras, o objetivo é organizar este conjunto de VMs de acordo com uma topologia em estrela, cujo centro (a VM selecionada) está mais perto, em média, de todos os seus vizinhos.

O Algoritmo 1 a seguir retorna, para um conjunto de VMs, aquela que possui a menor média de RTTs em relação a todas as outras VMs do conjunto, isto é, o 'centro' da estrela. O algoritmo percorre a diagonal principal da matriz, onde cada iteração se refere a uma VM específica. A matriz considerada é do tipo triangular superior, pois assim aproveita-se a simetria dos tempos de ida e volta (linhas 2 e 3) e a matriz pode, então, ocupar apenas metade da memória requerida por $n \times n$ elementos. A complexidade deste algoritmo é claramente $O(n^2)$.

Algorithm 1 Star

Input: RTT : Matriz de Latências para n VMs

1: **for** $i = 0$ **to** <*num. linhas*> *de* RTT **do**
2: $sumx \leftarrow \sum_{j=i+1}^{n} t_{ij}$
3: $sumy \leftarrow \sum_{j=0}^{i-1} t_{ji}$
4: $avg \leftarrow (sumx + sumy)/(n-1)$
5: **if** avg é a menor média até agora **then**
6: $center \leftarrow i$
7: **end if**
8: **end for**
9: **return** $center$

Este algoritmo pode ser útil para qualquer aplicação distribuída em que um nó deve estar no ponto médio em relação aos outros nós do conjunto. Este poderia ser o caso de uma VM funcionando como um diretório de nomes, um serviço de distribuição de chaves ou transmitido um fluxo de dados multicast para outras VMs. Em um modelo convencional provedor-a-cliente, onde ao último é simplesmente atribuída uma certa quantidade de VMs, é geralmente muito difícil acertar qual das VMs deveria ser escolhida como ponto central (*hub*). Vê-se, portanto, que em todos esses casos o cliente da

nuvem poderia beneficiar-se claramente desta **Seleção Assistida** de VMs, baseada em informações de latência periodicamente coletadas.

3.4. Algoritmo Melhores Centros (*Best Centers)*

Generalizando a abordagem do Algoritmo Estrela, pode-se desejar escolher não apenas uma VM como o centro, mas sim encontrar as m melhores VMs que poderiam funcionar como hubs (ou super-nós), agregando o tráfego de VMs adjacentes dentro da nuvem. Como exemplo de situações em que isso é necessário, considere uma aplicação que exige a manutenção de uma rede *overlay* ou uma tabela hash distribuída com super-nós utilizados para o gerenciamento de clusters próximos.

Propõe-se o Algoritmo 2 que é capaz de encontrar m centros dentre n VMs, todos os quais têm o menor RTT médio em relação a todas as outras VMs no conjunto. O algoritmo *Melhores Centros* tem como objetivo identificar as VMs que são as melhores candidatas a estrela e, portanto, podem potencialmente se tornar centros em torno dos quais grupos de VMs seriam formados. Assim, é possível identificar 'colmeias' de VMs espalhadas por toda a nuvem, a fim de responder a necessidades específicas dos clientes.

Algorithm 2 BestCenters

Input: RTT : Matriz de Latências para n VMs; m : número de 'best centers'

1: **for** $i = 0$ **to** <*num. linhas*> *de* RTT **do**
2: $sumx \leftarrow \sum_{j=i+1}^{n} t_{ij}$
3: $sumy \leftarrow \sum_{j=0}^{i-1} t_{ji}$
4: $avg \leftarrow (sumx + sumy)/(n-1)$
5: $centers\{\} \leftarrow [i, avg]$
6: **end for**
7: $best\{\} \leftarrow centers\{\}$ sorted by key avg ascending
8: **return** $best\{0..(m-1)\}$ {Retorna os m melhores centros}

Observe que este algoritmo pode ser usado como uma abordagem heurística para resolver o problema de agrupamento introduzido na Seção 3.2, porém utilizando uma técnica com muito menos complexidade computacional. Como pode ser visto, o Algoritmo 2 tem uma complexidade que é, pelo menos, igual à do algoritmo Estrela, ou seja, $O(n^2)$, além de passos adicionais para classificar o array $centers$, o que para uma rotina do tipo Quicksort seria $O(n \log n)$, em média.

3.5. Algoritmo Vizinhos *(Neighbors)*

Considere agora o caso em que se pretende encontrar m pares de VMs dentro da nuvem muito próximos uns dos outros, tipicamente com um tempo de ida e volta entre eles abaixo de um determinado limiar. Este é um caso especial do algoritmo Colmeia, porém limitando o número de VMs na colmeia para apenas duas, a fim de evitar uma explosão no número de iterações.

Esta abordagem é descrita no Algoritmo 3, que recebe uma matriz de latências como entrada e devolve todos os pares de VMs cujos RTTs sejam inferiores a um determinado limiar. Como a informação de tempo de ida e volta é simétrica, só é preciso verificar os elementos do lado direito da diagonal principal ($i < j$, como na Linha 2), uma

vez que os à sua esquerda serão testados à medida que o loop avança ao longo da matriz. Isto ainda proporciona uma complexidade $O(n^2)$ para o algoritmo, no entanto permite-lhe trabalhar com apenas metade dos elementos.

Algorithm 3 Neighbors

Input: RTT : Matriz de Latências para n VMs; $thres$: RTT máximo entre VMs

```
1: for i = 0 to <num. linhas> de RTT() do
2:    for j = i + 1 to <num. colunas> de RTT() do {somente i < j}
3:       if RTT(i, j) < thres then
4:          pairs{} ← RTT(i, j)
5:       end if
6:    end for
7: end for
8: return pairs{}
```

4. Avaliação das Políticas Fim a Fim

4.1. Arquitetura de Referência

A fim de avaliar a eficácia das políticas de seleção assistida de VMs e dos algoritmos propostos, foram realizados extensos estudos de simulação usando um modelo para latências entre VMs, desenvolvido no escopo deste trabalho a partir de medições reais entre instâncias EC2 da nuvem *Amazon Web Services* (AWS).

O primeiro passo foi definir uma arquitetura de referência da nuvem a ser utilizada para validar os algoritmos. Em *datacenters* atuais, a arquitetura de rede interna da nuvem é normalmente organizada de forma hierárquica, como mostrado na Figura 3. Na camada inferior, veem-se os *racks*, cada um com vários nós computacionais (referidos como CPUs, por simplicidade) que servem como hosts para um certo número de máquinas virtuais. As VMs dentro da mesma CPU comunicam-se diretamente, sem qualquer interferência da rede. Já as máquinas virtuais localizadas no mesmo rack, mas em diferentes processadores precisam se comunicar através de um switch *Top-of-Rack* (ToR). As VMs em racks diferentes têm que utilizar um agregador para alcançar uma a outra, por exemplo, se uma VM no rack 1 deseja enviar uma mensagem para outra VM no rack 2, esta mensagem terá de passar pelo switch ToR do primeiro rack, em seguida pelo switch agregador de segundo nível, o switch ToR do rack de destino, e assim por diante. Se os racks pertencerem a diferentes agregados, então um agregador de primeiro nível será também envolvido na comunicação. Assim, quanto mais longe estiverem as VMs fisicamente, maior tende a ser a latência de comunicação entre elas. Apesar de outros fatores como a sobrecarga da CPU hospedeira poderem influenciar na latência de rede, aplicações de nuvem em geral podem beneficiar-se de políticas de seleção de VMs que visem assegurar que a comunicação ocorra entre VMs próximas umas das outras.

Para efeito de validação dos algoritmos de seleção de VM, cientes de localização (*locality-aware*), foi utilizada a arquitetura de nuvem descrita acima. Nos experimentos, esta foi configurada com um agregador de nível superior ao qual 2 outros agregadores foram conectados, cada um com 2 racks ligados a ele, com 4 CPUs e 10 máquinas virtuais em cada uma. Isto dá um total de 160 VMs na nuvem (10 VMs por CPU, 4 CPUs por rack, 4 racks).

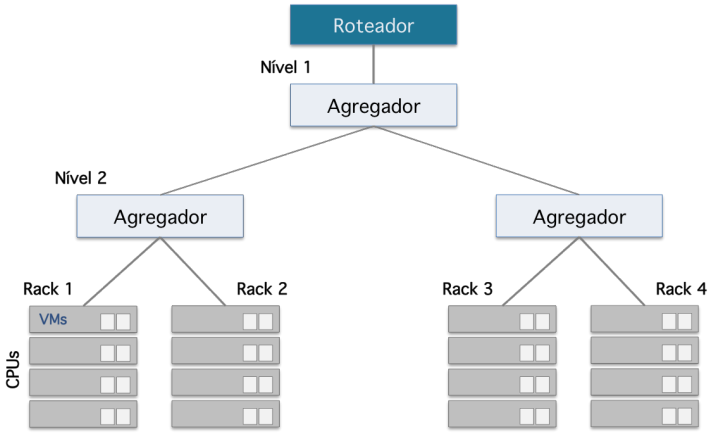

Figura 3. Arquitetura de nuvem usada como referência

Tabela 1. Diferentes níveis para os tempos de ida e volta

Nível	Localização da VM	Peso	%
1	Mesma CPU	0,1	5,6%
2	Mesmo Rack	1	18,8%
3	Mesmo Agregado	10	25,1%
4	Agregados distintos	50	50,3%

4.2. Amostragem dos Tempos de Ida e Volta (RTT)

Como discutido na Seção 3.1, os elementos da *Matriz de Latências* informam a distância, ou seja, os tempos de ida e volta entre quaisquer duas VMs dentro da nuvem. Para fins de implementação, é irrelevante se os dados na matriz referem-se a toda a nuvem ou a apenas um subconjunto de VMs alocadas a um cliente pelo provedor de nuvem. No contexto do Serviço de Recomendação (Seção 2.2), é mais realista assumir que se está lidando com um subconjunto de VMs sob controle de um *broker* de serviços de nuvem (Figura 1), que irá utilizar os algoritmos propostos para fazer uma seleção informada de VMs e então repassá-las ao cliente interessado.

A fim de preencher a Matriz de Latências, foi empregada uma distribuição uniforme no intervalo $[0, 1)$ com o objetivo de gerar aleatoriamente os valores de tempo de ida e volta. Estes serviram, então, como entrada para uma função de distribuição de probabilidade (FDP) acumulada a fim de produzir valores de RTT mais condizentes com os de uma nuvem típica. Os valores resultam, portanto, estratificados em níveis distintos, conforme as VMs comunicantes estejam na mesma CPU, rack, agregado, e assim por diante.

A Tabela 1 enumera os quatro níveis diferentes de RTT considerados, dependendo de onde as VMs estão localizadas. Os pesos atribuídos a cada nível foram consistentes com os experimentos realizados em máquinas Amazon EC2, como relatado na Seção 3.1. A última coluna indica a percentagem de VMs presentes em cada categoria, derivada da arquitetura mostrada na Figura 3. Esta tabela é empregada para preencher a Matriz de Latências, usada nas simulações.

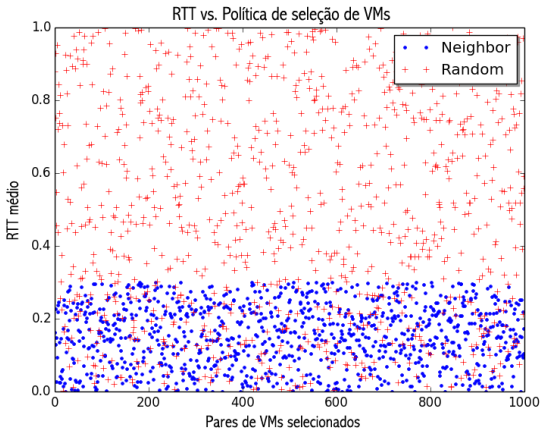

Figura 4. Comparação da seleção de VMs: algoritmo Vizinhos vs. Aleatório

4.3. Resultados

Nesta seção, serão relatados alguns resultados obtidos com o emprego das políticas propostas com o intuito de realizar uma seleção de m VMs a partir de uma população de n máquinas. A principal entrada para os algoritmos de seleção é a Matriz de Latências conforme descrito nas seções 3.1 e 4.2. Cada algoritmo é comparado com o caso aleatório, ou seja, quando um cliente simplesmente atribui aplicações a VMs ao acaso, como acontece normalmente em cenários atuais de utilização de nuvem. As simulações foram realizadas em um MacBook com processador Intel i5, 4GB de memória e sistema operacional Mac OSX 10.10.

No primeiro experimento, foi comparado o desempenho do **Algoritmo Vizinhos** com uma seleção aleatória de VMs. A Matriz de Latências é preenchida usando-se uma distribuição uniforme no intervalo $[0, 1)$ e o limiar de RTT é fixado em 0,3 unidades de tempo. O algoritmo foi executado para 100 VMs (ou seja, uma matriz de valores de RTT 100×100) com o objetivo de encontrar 1.000 pares de VMs dentro do limiar determinado. Os resultados na Figura 4 mostram que os valores de RTT para todos os pares de VMs selecionados pelo Algoritmo Vizinhos caem rigorosamente dentro da faixa de 0,0 a 0,3, como esperado, com um tempo médio de 0,1553. Para o caso aleatório, os pares selecionados encontram-se espalhados por todo o espaço de solução, uma vez que qualquer um deles poderia ser selecionado, exibindo um RTT médio de 0,5073. A percentagem de pares rejeitados pela abordagem Vizinhos, a fim de alcançar a seleção de VMs desejada é de 70,88%, em média, o que é consistente com uma distribuição uniforme entre 0 e 1 com um ponto de corte fixado em 0,3.

Para uma visão mais realista deste algoritmo em ação, foi então gerada uma matriz de latências de acordo com a FDP acumulada detalhada na Tabela 1. O limiar de RTT é fixado em 40, de modo que apenas os pares de VMs localizados no mesmo agregado, rack ou CPU sejam selecionados pelo algoritmo. O objetivo é novamente encontrar 1.000 pares de máquinas cujo RTT satisfaça os critérios especificados. Neste experimento, como pode ser visto através do extrato da Figura 5, os RTTs estão agrupados em torno dos limiares especificados pela FDP, no caso {0,1; 2; 30; 200 } ($nivel \times peso$). O RTT médio para os pares de VMs selecionados pelo algoritmo Vizinhos é 16,9566, em comparação com uma média de 112,6947 quando uma seleção aleatória é usada. A percentagem de descartes

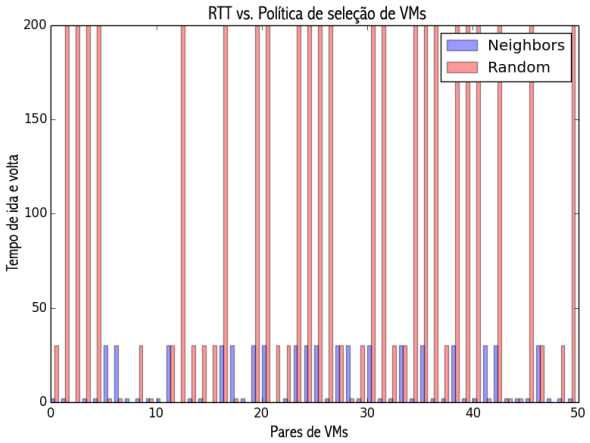

Figura 5. Comparação da seleção de VMs: Vizinhos vs. Aleatório (mesmo agregado)

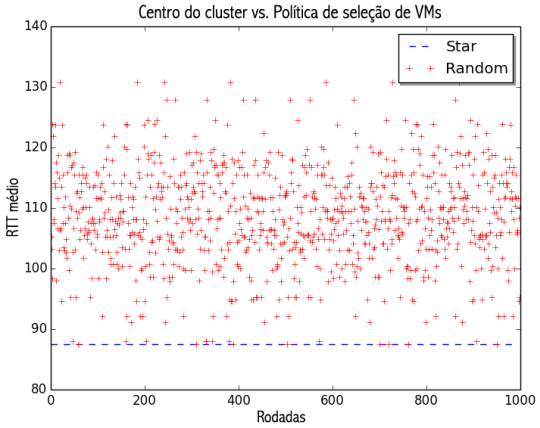

Figura 6. Comparação da seleção de VMs: Estrela vs. Aleatório

pela abordagem Vizinhos é de 49,44%, em média, o que corresponde a cerca de 50% de VMs localizadas em agregados distintos, conforme definido na Tabela 1. Para todos os casos relatados, o tempo de execução do algoritmo foi inferior a 1 segundo.

A segunda política de seleção a ser analisada é o **Algoritmo Estrela**. Neste caso, os valores para a matriz de latências foram gerados de acordo com a FDP descrita anteriormente. Consideram-se 100 VMs e executa-se o algoritmo para encontrar a VM que tem o menor RTT médio em relação a todos as outras VMs. Na Figura 6, há uma comparação entre o centro real encontrado pelo algoritmo Estrela e 1.000 escolhas aleatórias de centros. Como pode ser visto, o algoritmo proposto aponta sempre para o mesmo centro para uma dada configuração da matriz de RTTs, obtendo-se neste caso um RTT médio de 87,4828. No entanto, no caso aleatório, a maioria dos "centros" cai longe da meta, escapando-a por mais de 20%.

Finalmente, foi realizado um experimento para estudar o Algoritmo **Melhores Centros**. Foi gerada uma matriz de latências com 100 VMs usando uma FDP, como anteriormente. Este algoritmo atende a uma classe especial de aplicações que requerem um conjunto de *hubs* que de alguma forma condensam o tráfego ao seu redor, funcionando

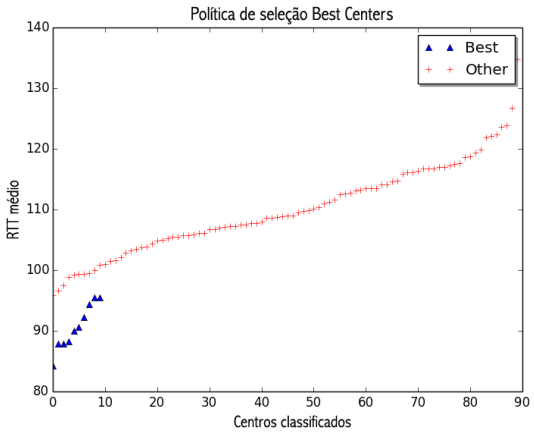

Figura 7. Seleção de VMs: algoritmo Melhores Centros

como pontos de atração. Neste experimento, pretende-se encontrar os 10% melhores centros pertencentes ao conjunto de VMs fornecido. Cada VM é escolhida, individualmente, como um centro candidato e seu RTT médio em relação às outras VMs é calculado. Finalmente, todos os RTT médios são classificados do menor para o maior.

Os resultados estão representados na Figura 7, onde os triângulos azuis indicam com precisão quais das VMs melhor servem ao propósito de funcionar como hubs locais. Ao contrário do que pode parecer à primeira vista, esses melhores centros não estão agrupados em uma região específica da matriz, mas sim espalhados por ela; por exemplo, em uma rodada particular do algoritmo a lista encontrada de candidatos a centro foi {36, 57, 25, 5, 82, 77, 50, 45, 75, 56}, sendo estes os índices das VMs na matriz. O tempo de execução do algoritmo foi inferior a um segundo, o que faz com que seja suficientemente rápido para ser utilizado em um cenário real de nuvem.

4.4. Discussão

Como mostrado pelos resultados acima, o emprego de políticas de seleção assistida de VMs permite harmonizar melhor o perfil dos aplicativos com a utilização atual das VMs e da nuvem. Isso resulta em um melhor mapeamento de aplicativos sobre VMs e pode ter um impacto decisivo no desempenho das aplicações executando na nuvem. Todos os algoritmos propostos neste trabalho atingiram o seu objetivo, com tempos de execução não significativos. Além disso, foram capazes de realizar escolhas de VMs muito mais adequadas do que aquelas obtidas por meio de uma seleção cega (aleatória) de recursos, como normalmente é feito.

Note-se que mesmo uma política de seleção simples e direta como o algoritmo Vizinhos pode realmente produzir resultados muito bons em termos de extrair um subconjunto de VMs que atenda a alguns requisitos anteriores especificados pelo cliente da nuvem. Este algoritmo pode ainda ser calibrado de modo a permitir apenas o retorno de VMs com uma determinada latência uma da outra, como foi o caso do experimento de 'mesmo agregado' relatado acima. No entanto, apesar do atraso estar fortemente correlacionado com a distância física dentro da nuvem, este nem sempre é o caso, porque às vezes uma latência mais alta resulta da sobrecarga da CPU causada pelo processamento da parte de rede da VM. De qualquer maneira, o que é requerido do ponto de vista do cli-

ente são pares de VMs com um atraso dentro de um determinado intervalo de tempo e é precisamente isso que o algoritmo fornece. Além disso, seu tempo de execução diminuto permite rodadas frequentes, a fim de responder rapidamente a mudanças nos padrões de comunicação entre aplicações dentro da nuvem.

Da mesma forma, os resultados do algoritmo Estrela são um excelente exemplo de como uma seleção assistida de VMs pode fazer uma grande diferença para os clientes da nuvem. Suponha que, por algum motivo, eles precisem escolher uma de suas VMs contratadas como centro de um dado subconjunto de VMs, talvez porque essa máquina deva executar um serviço de diretório. Muito provavelmente, o cliente iria terminar, por infortúnio, alocando o servidor de diretório longe de seus clientes, com um impacto negativo no desempenho do sistema. Por outro lado, este algoritmo pode indicar com precisão qual a VM mais apropriada para tal função. Melhor ainda, ele atinge essa conclusão por meio de medidas de RTT apenas, sem nenhum conhecimento adicional da arquitetura interna da nuvem, ou de quaisquer aplicações de terceiros executando simultaneamente nessas VMs.

O algoritmo Melhores Centros realiza uma tarefa ainda mais difícil, a de encontrar um conjunto de VMs, dentre as contratadas, que sejam mais adequadas para desempenhar o papel de centros de tráfego locais. Tal característica seria especialmente importante numa aplicação *peer-to-peer* dentro da nuvem, onde cada nó poderia assumir o papel de um supernó para as suas máquinas vizinhas. Como os resultados demonstraram, este algoritmo é capaz de encontrar as VMs mais adequadas para este fim e repassá-las ao cliente da nuvem, o que representa uma enorme vantagem quando comparado com uma seleção cega de VMs.

As políticas de seleção e algoritmos analisados distinguem-se de abordagens semelhantes porque são capazes de realizar melhores escolhas de VMs com um conhecimento mínimo da infraestrutura de nuvem subjacente. Eles simplesmente inferem as distâncias entre VMs a partir de medições de tempo de ida e volta e então utilizam esta informação como heurística para fazer seleções mais inteligentes de VMs.

5. Conclusão

Pesquisas recentes apontaram que os padrões de comunicação podem desempenhar um papel importante no desempenho geral dos aplicativos na nuvem, o que tem atraído muita atenção da comunidade para esse tema. Neste trabalho, propõe-se um Serviço de Recomendação ao qual um cliente da nuvem faz uma solicitação de VMs, sujeita a critérios específicos e recebe, como resposta, um conjunto de máquinas que satisfaça essas exigências. Este serviço permite que sejam tomadas decisões de alocação de VMs mais bem informadas e adequadas, em contraste com a alocação cega de recursos frequentemente utilizada na computação em nuvem.

Em particular, este estudo teve seu foco em políticas e algoritmos de seleção de VMs, cientes de localização, que utilizam a latência de rede entre VMs como heurística para apoiar decisões de alocação. Medições de tempo de ida e volta foram usadas como a principal entrada para os algoritmos e quatro diferentes políticas de seleção de VMs foram propostas e avaliadas. Todas elas empregam uma abordagem fim a fim e são independentes de detalhes de topologia da nuvem. O tempo de execução dos algoritmos foi não significativo e todos retornaram recomendações de VMs muito melhores do que as

obtidas a partir de uma seleção não-informada de máquinas.

Como trabalho futuro, pretende-se implantar o Serviço de Recomendação e os algoritmos desenvolvidos em uma infraestrutura de nuvem real, assim como avaliar a utilidade desta abordagem para aplicações sensíveis à latência. Isto irá permitir uma melhor compreensão do desempenho dos algoritmos numa situação real, bem como indicará novas formas de aplicá-los.

Agradecimentos

O primeiro autor gostaria de agradecer à CAPES pelo suporte financeiro concedido a este trabalho por intermédio do programa Ciência Sem Fronteiras, bem como ao Hariri Institute da Boston University por tê-lo acolhido como Professor Visitante no ano de 2014. Esta pesquisa foi ainda apoiada parcialmente por várias concessões do NSF (*National Science Foundation*), nos EUA, dentre elas PFI:BIC award #1430145, SaTC awards #1414119 e #1012798, CNS awards #1347522 e #1239021.

Referências

Alicherry, M. and Lakshman, T. (2012). Network aware resource allocation in distributed clouds. In *INFOCOM, 2012 Proceedings IEEE*, pages 963–971. IEEE.

Battré, D., Frejnik, N., Goel, S., Kao, O., and Warneke, D. (2011). Evaluation of network topology inference in opaque compute clouds through end-to-end measurements. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 17–24. IEEE.

Bestavros, A. and Krieger, O. (2014). Toward an open cloud marketplace: Vision and first steps. *Internet Computing, IEEE*, 18(1):72–77.

Böhm, M., Koleva, G., Leimeister, S., Riedl, C., and Krcmar, H. (2010). Towards a generic value network for cloud computing. In *Economics of Grids, Clouds, Systems, and Services (GECON)*, pages 129–140. Springer.

Jonglez, B., Boutier, M., and Chroboczek, J. (2014). A delay-based routing metric. *arXiv:1403.3488*.

LaCurts, K., Deng, S., Goyal, A., and Balakrishnan, H. (2013). Choreo: network-aware task placement for cloud applications. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 191–204. ACM.

Meng, X., Pappas, V., and Zhang, L. (2010). Improving the scalability of data center networks with traffic-aware virtual machine placement. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE.

MOC (2014). Massachusetts Open Cloud (MOC) . `http://www.bu.edu/moc`. [Online; accessed 3-Nov-2014].

Piao, J. T. and Yan, J. (2010). A network-aware virtual machine placement and migration approach in cloud computing. In *9th International Conference on Grid and Cooperative Computing (GCC), 2010*, pages 87–92. IEEE.

Shea, R., Wang, F., Wang, H., and Liu, J. (2014). A deep investigation into network performance in virtual machine based cloud environments. In *INFOCOM, 2014 Proceedings IEEE*, pages 1285–93. IEEE.