Detecção divergente de carga de CPU em máquinas virtuais

Alexandre Heideker Carlos Kamienski
Universidade Federal do ABC, Santo André, São Paulo
{alexander.heideker, cak}@ufabc.edu.br

Abstract: A virtualização exerce hoje um papel fundamental na infraestrutura dos sistemas de informação, sendo a base de ambientes de nuvem computacional. Detectar o estado atual das máquinas virtuais permite o gerenciamento de grandes aplicações, determinando os recursos necessários para garantir a qualidade de serviço sem desperdício de recursos. A porcentagem de uso da CPU é amplamente adotada para detectar a sobrecarga de aplicações, porém, em infraestruturas virtualizadas esta avaliação pode produzir discrepâncias entre o valor reportado e o real estado do sistema. Este trabalho apresenta um experimento onde esta divergência é observada.

1. Introdução

A virtualização exerce hoje um papel fundamental na infraestrutura dos sistemas de informação e micro serviços, exercendo um papel central no conceito de nuvem computacional. Inicialmente, esse conceito surge como uma ferramenta para compatibilizar diferentes gerações de arquitetura e atualmente apresenta-se como técnica para um melhor aproveitamento do hardware disponível, além do consequente impacto energético positivo [1].

A detecção do estado atual das máquinas virtuais (VM) permite o gerenciamento de grandes aplicações, determinando os recursos necessários para garantir a qualidade de serviço sem desperdício de recursos. Quando VMs estão sobrecarregadas, podem ser substituídas por configurações superiores (*scale-up*), ou paralelizando o trabalho em novas VMs (*scale-out*) [2]. O desafio de equacionar estas demandas, gerenciando dinamicamente os recursos computacionais disponíveis e possibilitando não só absorver este crescimento da infraestrutura, como também garantir a qualidade de serviço mostra-se cada vez mais relevante [3].

O uso da carga de CPU é amplamente adotado para detectar a sobrecarga de aplicações, já que é uma métrica presente em todas as implementações de VMs, porém, esta avaliação pode produzir discrepâncias entre o valor reportado e o real estado do sistema. Em [4] os autores tratam da criação de uma API para reportar o estado da VM utilizada em ambientes de Virtualização de Funções de Rede (NFV), considerando aspectos da aplicação. Já em [5] fica clara a discrepância entre a carga de CPU reportada pela aplicação virtualizada e a qualidade de serviço ofertada.

Para demonstrar esta discrepância, este trabalho apresenta um experimento que reporta a carga de CPU utilizada por uma VM e de sua máquina real hospedeira, que é sobrecarregada através da criação de novas instâncias, sem que esta sobrecarga seja reportada pela VM avaliada. A avaliação da qualidade de serviço ofertada pela VM avaliada corroboram as conclusões do estudo.

2. Metodologia

O processo de virtualização se divide em três modelos: virtualização total, para-virtualização e virtualização leve [6]. A virtualização total fornece ao sistema operacional (SO) hóspede uma completa abstração da máquina real, sem que o mesmo consiga detectar que a máquina está virtualizada. Na para-virtualização, parte do SO hóspede é modificado para otimizar o seu desempenho. O hipervisor exerce papel fundamental no processo de virtualização total e para-virtualização por ser responsável por fornecer e gerenciar o hardware virtual, o uso do hardware real e promover as modificações no SO hóspede no caso da para-virtualização. O Xen, KVM, VirtualBox e o VMware são exemplos de hipervisores.

A virtualização leve utiliza as características de compartimentalização do SO hospedeiro para tratar o SO hóspede como um processo, compartilhando estruturas do kernel do SO hospedeiro – ganha-se em performance, mas perde-se em isolamento. O LXC e o Docker são exemplos de hipervisores para virtualização leve. Apesar da adoção da virtualização leve estar em crescimento, o experimento proposto avalia a discrepância entre a carga de CPU reportada pelo hipervisor e o real estado do serviço limitando-se ao uso de virtualização total utilizando o hipervisor KVM.

Apesar das limitações do experimento, esta discrepância pode ser observada em todos os hipervisores, não como uma falha, mas sim como uma forma de equilibrar a distribuição dos recursos de hardware real disponíveis entre as instâncias hospedadas na máquina. A Figura 1(a) apresenta do esquema do experimento. Utilizou-se um servidor equipado com processador Intel Xeon X3333, 2.66Ghz com 6 Gbytes de memória RAM, 2 núcleos e 4 threads, SO

Ubuntu 18.04 e KVM ver 2.11. Cada instância foi dimensionada com 1vCPU, 512Mbytes de RAM e SO Ubuntu 18.04. Para gerar uma carga de processamento homogênea, cada instância executa em *looping* um programa para cálculo do "pi" que gera uma carga aproximada de 50% de uso de CPU. O gerador de tráfego faz requisições a um servidor web Apache 2 instalado na VM-01, com distribuição exponencial de intervalos e o tamanho da requisição por uma distribuição Log-normal.

O experimento começa com apenas a VM-01 instanciada e a cada três minutos uma nova instância é criada na máquina real, como mostra a Figura 1(b). Durante o experimento é registrada a carga de CPU na máquina real, na VM-01 e a qualidade de serviço (QoS) observada pelo gerador de tráfego, dada pelo tempo necessário para realizar o download de 1Mbyte do servidor web.

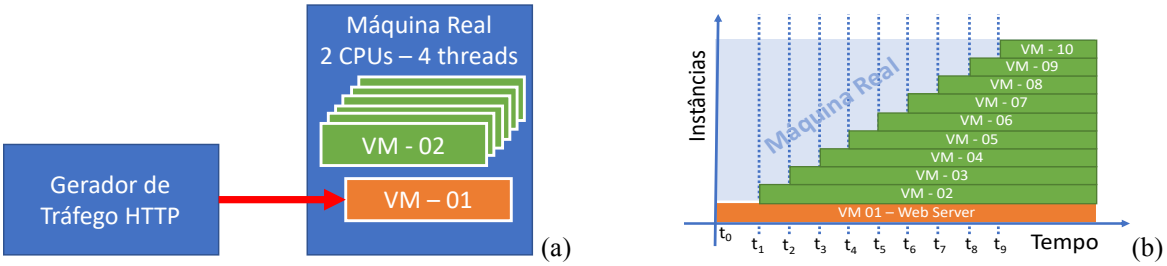

Figura 1: (a) Modelagem do experimento. (b) Criação das instâncias ao longo do experimento.

3. Resultados

Na Figura 2(a) é possível observar a carga de CPU na VM-01 com um comportamento uniforme ao longo do experimento, insensível ao crescimento da carga de CPU da máquina real – os picos observados referem-se ao momento em que uma nova instância é criada. Confirmando a discrepância, podemos observar na Figura 2(b) uma queda de aproximadamente 30% no desempenho de QoS, sem qualquer correlação com a carga de CPU reportada pela VM.

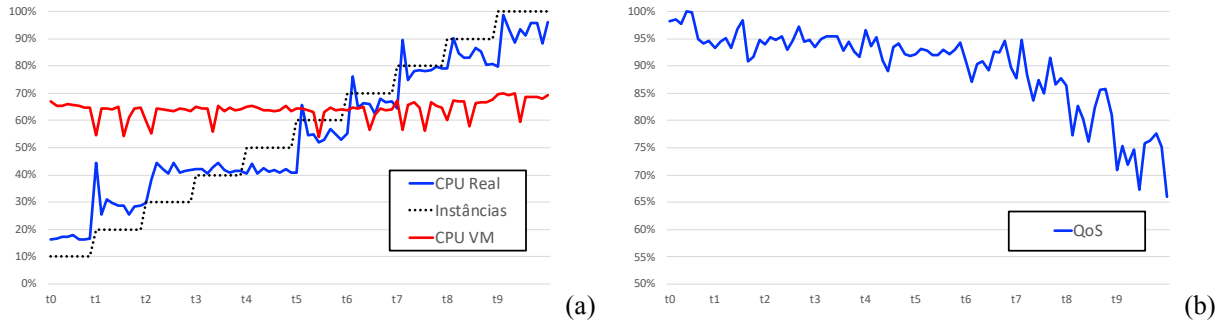

Figura 2: (a) Comparação entre a carga de CPU observada na máquina real e na VM-01. (b) Qualidade de serviço obtida pelo gerador de tráfego.

4. Conclusão

Este trabalho apresentou uma avaliação de desempenho da carga de CPU em máquina real e virtual que demonstra a discrepância observadas em sistemas reais na detecção de sobrecarga ou subutilização de recursos virtualizados. A queda na qualidade de serviço observada não apresentou correlação ao comportamento reportado pela CPU. Como trabalhos futuros, é possível realizar um estudo mais amplo com outros tipos de virtualização, diferentes hipervisores e soluções comercias, assim como diferentes cargas de trabalho concorrentes.

5. Referências

[1] L. Lefèvre and A. C. Orgerie, "Designing and evaluating an energy efficient Cloud," Journal of Supercomputing, 2010.

[2] G. Galante and L. C. E. de Bona, "A Survey on Cloud Computing Elasticity," IEEE Inter. Conf. on Utility and Cloud Computing, 2012.

[3] A. Heideker and C. A. Kamienski, "ElasticNFV: An elasticity manager for NFV using SDN," IEEE Lat. Am. Trans., 2019.

[4] J. Khalid, M. Coatsworth, A. Gember-Jacobson, and A. Akella, "A Standardized Southbound API for VNF Management," SIGCOM - HotMIddlebox, 2016.

[5] A. Heideker and C. Kamienski, "Towards a Network Queuing Assessment for Elasticity Management of Virtualized Services," (aceito para publicação) Consumer Communications and Networking Conf. (CCNC), 2021.

[6] A. Binu and G. S. Kumar. "Virtualization techniques: a methodical review of XEN and KVM," In: ICACC. Springer, Berlin, Heidelberg, 2011.