Aprimorando a elasticidade em ambientes de virtualização em nível de Sistema Operacional

Israel Campos de Oliveira, César A. F. De Rose

[1]Programa de Pós-Graduação em Ciência da Computação
Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Av. Ipiranga, 6681 – Porto Alegre – RS – Brazil

israel.oliveira@acad.pucrs.br, cesar.derose@pucrs.br

Resumo. *Este trabalho propõe uma avaliação do impacto da estratégia de alocação dinâmica de recursos proposta por Dawoud et al. [Dawoud et al. 2012] sobre virtualização em nível de Sistema Operacional, com foco no uso de aplicações de banco de dados e com a finalidade de manter um determinado SLO.*

1. Introdução

Para manter um serviço em execução dentro de níveis de qualidade de serviço (QoS) aceitáveis, os administradores de sistemas devem provisionar recursos suficientes para lidar com as flutuações de carga de trabalho, considerando que os recursos provisionados em excesso aumentam os custos operacionais das empresas, enquanto a provisão subestimada de recursos degrada o desempenho do serviço. No entanto, o mapeamento dinâmico de métricas de QoS em recursos de infraestrutura visando economia de recursos, como CPU e memória, consiste em um grande desafio. A abordagem mais comum propõe estabelecer limites (*thresholds*) superiores e inferiores (fixos ou dinâmicos) que são comumente usados para determinar ações de elasticidade. Embora amplamente utilizados, limites que apresentam valores fixos permitem a quebra de *Service Level Objectives* (SLOs) quando o comportamento da aplicação se mostra irregular. Já limites dinâmicos podem onerar métricas de desempenho devido às seguidas mudanças na quantidade de recursos alocados.

2. Motivação e Proposta

A fim de superar o desafio de suavizar *trade-offs* ocasionados pela utilização de métricas fixas, dinâmicas ou híbridas, diversos trabalhos têm proposto diferentes abordagens. Zhu et al. [Zhu et al. 2006] desenvolveu um modelo matemático com o objetivo de assegurar que uma determinada máquina virtual receba recursos suficientes para manter um serviço dentro dos níveis de eficiência desejados. Este modelo foi utilizado por Dawoud et al. [Dawoud et al. 2012] na construção de sua arquitetura elástica. Dawoud et al. utilizou o Xen como solução de virtualização, o que torna nosso trabalho diferenciado devido à utilização de técnicas de virtualização em nível de Sistema Operacional. Essa nova abordagem de virtualização apresenta vantagens de desempenho frente ao Xen [Xavier et al. 2013]. A arquitetura desenvolvida por *Dawound et al.* se destaca entre as demais estudas, por trazer uma série de benefícios para o usuário como: (i) controle dinâmico da fatia de recursos necessária; (ii) melhor aproveitamento dos recursos alocados; (iii) manutenção do desempenho do serviço. Os principais componentes da arquitetura de *Dawound et al.* são os atuadores de CPU e Mémória ambos projetados por *Zhu et*

al. [Zhu et al. 2006]. O atuador de CPU é expresso pelas seguintes equações:

$$a_{cpu}(k+1) = a_{cpu}(k) - K_1(k)(u_{ref}^{cpu} - u_{cpu}(k)) \tag{1}$$

onde

$$k_1 = \alpha.c_{cpu}(k)/a_{cpu}(k) \tag{2}$$

tem a função de calcular a próxima alocação de CPU $a_{cpu}(k+1)$, baseando-se na última alocação de CPU $a_{cpu}(k)$, além do consumo $c_{cpu}(k)$. A variável u_{cpu} consiste na última utilização de CPU e é representado por $u_{cpu}(k) = c_{cpu}(k)/a_{cpu}(k)$. O parâmetro α em K_1 é o ganho constante, que determina a agressividade do atuador. Com a finalidade de ajustar o valor de utilização de CPU u_{ref} dinamicamente, para assegurar que um SLO seja atendido, outro atuador também foi projetado por [Zhu et al. 2006]:

$$u_{ref}^{cpu}(i+1) = u_{ref}^{cpu}(i) + \beta(SLO_{ref}^{cpu} - SLO_{cpu}(i))/SLO_{ref}^{cpu} \tag{3}$$

onde $u_{ref}^{cpu}(i+1)$ é a utilização de CPU desejada, $SLO_{cpu}(i)$ é o SLO medido, e SLO_{ref}^{cpu} é o SLO desejado. Este atuador garante que o valor de utilizacão CPU esteja sempre dentro de um intervalo aceitável de utilização da CPU.

Já o atuador de memória é expresso pela seguinte equação:

$$a_{mem}(i+1) = a_{mem}(i) + k_2(i)(u_{ref}^{mem} - u_{mem}(i)) \tag{4}$$

onde

$$K_2(i) = \lambda.u_{mem}(i)/u_{ref}^{mem} \tag{5}$$

com o objetivo de manter o sistema longe de gargalos de memória. O controlador aloca agressivamente mais memória, quando a memória alocada anteriormente está próximo da saturação (por exemplo, 90%), e diminui lentamente a alocação de memória quando a alocação está numa região longe de gargalos.

3. Perspectivas

A utilização de *containers* permite uma melhor manipulação dos recursos disponíveis, além de não depender de um monitor de máquinas virtuais. Isso permite que os *containers* tenham desempenho de CPU, memória, disco e rede similares ao de um ambiente nativo [Xavier et al. 2013]. Devido a esses motivos, acreditamos que ambientes de *containers* podem se beneficiar com a utilização do modelo de Dawoud et al., atenuando problemas de intrusividade e I/O.

Referências

Dawoud, W., Takouna, I., and Meinel, C. (2012). Elastic virtual machine for fine-grained cloud resource provisioning. In Krishna, P., Babu, M., and Ariwa, E., editors, *Global Trends in Computing and Communication Systems*, volume 269 of *Communications in Computer and Information Science*, pages 11–25. Springer Berlin Heidelberg.

Xavier, M., Neves, M., Rossi, F., Ferreto, T., Lange, T., and De Rose, C. (2013). 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*, pages 233–240.

Zhu, X., Wang, Z., and Singhal, S. (2006). Utility-driven workload management using nested control design. In *American Control Conference, 2006*, pages 6 pp.–.