Elasticidade Automática Baseada em Thresholds para Aplicações Paralelas Iterativas em Nuvem Computacional

Vinicius Facco Rodrigues[1], **Gustavo Rostirolla**[1], **Rodrigo da Rosa Righi**[1]

[1]Prog. Interdisciplinar de Pós-Graduação em Computação Aplicada, Unisinos - Brasil
Email: viniciusfacco@live.com, grostirolla1@gmail.com, rrrighi@unisinos.br

Resumo. *Na área de computação de alto desempenho, as iniciativas normalmente trabalham com aplicações paralelas iterativas com necessidade de alterações no código para o tratamento da elasticidade em ambientes de Computação em Nuvem. Nesse contexto, esse artigo apresenta o modelo de elasticidade para aplicações paralelas chamado AutoElastic, oferecendo provisionamento automático de recursos com base em análise de thresholds.*

1. Introdução

A elasticidade de recursos em ambientes de computação em nuvem explora o fato que a alocação deles é um procedimento que pode ser efetuado automaticamente de acordo com a demanda do serviço ou do usuário [Righi 2013]. Esforços recentes mostram a exploração da elasticidade em nuvem para serviços com alta demanda de entrada/saída [Righi 2013, Al-Haidari et al. 2013]. Em adição, múltiplos provedores de computação em nuvem estão focando em aplicações que demandam alto desempenho (HPC). A abordagem mais comum em tais serviços é a elasticidade reativa baseada na replicação de máquinas virtuais (VMs) quando um determinado índice de observação (*threshold*) de uma métrica for atingido [Al-Haidari et al. 2013]. Contudo, as obras não são uníssonas sobre a utilização de *thresholds* [Al-Haidari et al. 2013]. Nesse contexto, esse artigo apresenta o modelo **AutoElastic**, que gerencia a elasticidade em aplicações HPC iterativas através da análise de utilização de recursos e violação de *thresholds*.

2. Tratamento de Elasticidade em Aplicações Paralelas Iterativas

O modelo trabalha com elasticidade automática e reativa, proporcionando a alocação e consolidação de nós computacionais e VMs. Uma vez que normalmente aplicações HPC são intensivas quanto ao uso de CPU, optou-se por criar um processo por VM e n VMs por nó, sendo n o número de núcleos de processamento que o nó possui. A técnica de elasticidade usada é a de replicação [Kouki et al. 2014]. Na atividade de aumento da infraestrutura, AutoElastic aloca um novo nó e lança nele novas VMs. O *boot* de uma VM é finalizado com a execução de um processo escravo, que requisita comunicação com o mestre. Quanto a consolidação, o grão de trabalho é sempre um nó e suas VMs buscando um uso eficiente dos recursos e melhor gerência no consumo de energia elétrica.

O monitoramento para as ações de elasticidade é dado de forma periódica através da captura da métrica CPU para análise temporal e comparação com os *thresholds* máximo e mínimo. As ações de elasticidade são disparadas em situações nas quais algum dos *thresholds* é violado aplicando-se a ideia de média móvel sob um determinado número de observações. Para tal, AutoElastic coleta dados de CPU das VMs com base nas seguintes equações: $PC(i) = (\sum_{j=1}^{n} MM(i,j))/n$ e $MM(i,j) =$

$(\sum_{k=i-x+1}^{i} Cjk)/x \;\; em\ que \;\; i \geq x$. Nesse contexto, $MM(i,j)$ é responsável por informar a carga da máquina virtual j na observação i. Para tal, MM faz uso de média móvel levando em consideraçãs as x últimas observações de carga C a partir de i. Usando esse valor, faz-se a média aritmética e estabelece-se a carga média do sistema na observação i pela função $PC(i)$, na qual n representa o número de máquinas virtuais em execução. Por fim, recursos são alocados caso PC for maior que o *threshold* máximo, enquanto que recursos são consolidados quando PC for menor que o *threshold* mínimo.

A aplicação usada nos testes calcula a aproximação para a integral do polinômio $f(x)$ num intervalo fechado $[a, b]$. Para simular variações de carga durante a execução do cálculo, foram modelados 4 padrões de carga: Constante, Crescente, Decrescente e Onda. Em cada cenário a aplicação realiza o cálculo de uma mesma função n vezes, sendo que para cada execução a quantidade de subintervalos entre $[a, b]$ é recalculada, gerando variações de carga de processamento. Cada teste foi realizado com uma combinação diferente de *threshold* máximo e mínimo. Em cada execução, a configuração inicial do ambiente consistia em 2 nós, cada um deles com 2 VMs executando processos escravos e um dos nós com 1 VM executando um processo mestre. Para os testes o valor de x foi definido como 3 e o intervalo de monitoramento em 30 segundos. A Figura 1 ilustra os resultados obtidos nos diferentes cenários. É possível identificar que, em todos as cargas, as variações de desempenho ocorrem em sintonia com a variação do *threshold* máximo.

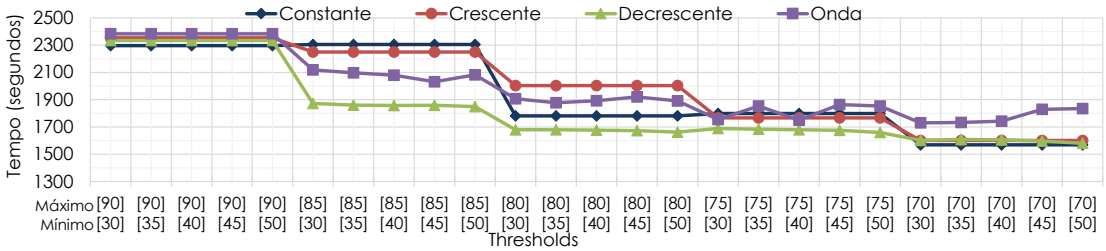

Figura 1. Tempo de execução da aplicação com diferentes cargas e *thresholds*.

3. Conclusão

Este artigo apresentou uma análise de desempenho de aplicações paralelas em nuvem com elasticidade automática de recursos. Resultados demonstraram o impacto da variação dos *thresholds* no desempenho de tais aplicações, nos quais foi demonstrado que é possível obter um ganho desempenho de 27% a 32% conforme a variação do *threshold* máximo. Quanto a trabalhos futuros, pode-se citar a auto-organização dos *thresholds* de acordo com o histórico da aplicação. Por fim, planeja-se estender AutoElastic para contemplar outros modelos de programação paralela como Divisão-e-Conquista e Fases Síncronas.

Referências

Al-Haidari, F., Sqalli, M., and Salah, K. (2013). Impact of CPU Utilization Thresholds and Scaling Size on Autoscaling Cloud Resources. *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, pages 256–261.

Kouki, Y., Oliveira, F. A. d., Dupont, S., and Ledoux, T. (2014). A language support for cloud elasticity management. In *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on*, pages 206–215.

Righi, R. (2013). Elasticidade em cloud computing: conceito, estado da arte e novos desafios. *Revista Brasileira de Computação Aplicada*, pages 2–17.