GetLB++: Otimizando o Escalonamento de Transações em Sistemas de Transferência Eletrônica de Fundos

Felipe Rabuske Costa[1], **Rodrigo da Rosa Righi**[1]

[1]Universidade do Vale do Rio dos Sinos (Unisinos)
São Leopoldo – RS – Brasil

rabuske@outlook.com, rrrighi@unisinos.br

Resumo. *Este artigo apresenta o framework GetLB++, cujo foco é melhorar o balanceamento de carga em sistemas de processamento de transações eletrônica de fundos. Sua arquitetura consiste em uma versão melhorada do modelo GetLB. Os resultados mostram que o GetLB++ apresenta um menor tempo de decisão, uma maior escalabilidade e flexibilidade, mantendo a mesma qualidade do balanceamento de carga apresentado pelo GetLB.*

1. Introdução

Os meios eletrônicos de pagamento, como cartões de crédito e débito, estão sendo cada vez mais adotados pela população para a realização de transações comerciais. Os benefícios trazidos por esse tipo de meio de pagamento vão desde uma maior comodidade para o consumidor final, até uma maior segurança para as instituições de comércio [Singh et al. 2014].

Com a intenção de melhorar o balanceamento de carga nas centrais de processamento, ao substituir o normalmente utilizado algoritmo *Round Robin*, Rodrigo Righi et al. propuseram em seu trabalho um novo *framework* para escalonamento de transações, chamado GetLB. Esse novo *framework* leva em consideração propriedades das transações e dos nós do cluster para realizar a distribuição do trabalho. Porém, de acordo com os testes realizados pelos autores, apesar de distribuir de uma forma melhor a carga de trabalho, o cálculo realizado pelo algoritmo de escalonamento utilizado pelo GetLB (*Load Level*) é mais lento que o *Round Robin* tendendo a diminuir sua velocidade conforme o número de nós do cluster aumenta [Righi et al. 2014].

Nesse contexto, esse trabalho apresenta uma nova versão desse framework, batizada de **GetLB++**, cuja heurística de balanceamento é desvencilhada do *framework* de despacho de tarefas, e arquitetura foi totalmente repensada para torná-lo mais flexível. A avaliação do modelo através de testes em diferentes configurações mostra que o modelo GetLB++ consegue obter uma maior velocidade de escalonamento e uma maior escalabilidade que o GetLB.

2. Trabalhos Relacionados

A literatura atualmente é escassa ao tratarmos de pesquisas relacionadas a sistemas distribuídos voltados para transferência eletrônica de fundos. Alguns dos trabalhos são apresentados aqui.

O trabalho de [Sousa et al. 2009] propõe um modelo estocástico polinomial para a realização de análise de desempenho em sistemas de transferência eletrônica de fundos

e um modelo para análise da confiabilidade da sua infraestrutura. Os autores utilizam redes de Petri estocásticas generalizadas(RPEGs) para a criação do modelo de análise de desempenho. Os testes demostram um erro relativo de no máximo 13,60% entre o modelo proposto e o sistema real.

O trabalho [Staskauskas 1988] apresenta o design e a especificação de um sistema de transferência eletrônica de fundos utilizando a linguagem de programação paralela *UNITY*, que tenta prover uma base para a especificação de aplicações voltadas para arquiteturas paralelas e distribuídas ao separar conceitos referentes à lógica da aplicação da arquitetura do sistema. Além disso, após a definição do design, o autor realizou uma série de refinamentos.

Ambos os trabalhos apontam as mesmas dificuldades para realizar uma avaliação de desempenho ou a criação de um algoritmo de balanceamento de carga para esse tipo de sistema, que incluem decisões de projeto que implicam a necessidade de um conhecimento maior do sistema abordado, assim como aspectos de monitoramento de recursos.

3. Fundamentação Teórica

Esta seção apresenta o funcionamento do GetLB original, fundamental para o entendimento do funcionamento do GetLB++. Seu projeto leva em consideração as seguintes premissas: I) o escalonamento deve levar em consideração diversas informações para compor sua noção de carga; II) as máquinas processadoras (MP) devem ser capazes de notificar o comutador caso ocorra algum evento que tenha algum impacto no escalonamento; III) o *framework* deve ser capaz de lidar com recursos heterogêneos. Essas premissas são consideradas pelos autores como os três pilares do GetLB.

Para realizar o escalonamento das transações, o GetLB utiliza uma heurística desenvolvida especialmente para o *framework*, chamada LL (*Load Level*), que leva em consideração tanto as características da transação quanto das máquinas processadoras. Essa heurística pode ser vista como uma função de decisão $LL(i, j)$ onde i é um tipo de transação e j uma máquina candidata a realizar o processamento da transação i. Portanto, a cada nova transação, o comutador realiza a computação de LL para todas as máquinas processadoras, a fim de descobrir qual delas deverá processar a referida transação.

Nos testes realizados, os autores comparam uma implementação do *Round Robin* e um protótipo do GetLB. Foram utilizados dados de transações coletados pela empresa GetNet (preservando se dados sigilosos) em um *cluster* de máquinas heterogêneas e um *cluster* de máquinas homogêneas. Através da análise dos resultados, os autores concluíram que o uso do GetLB proporcionou uma redução de 16,83% no tempo de processamento no *cluster* homogêneo e de 10,33% no ambiente heterogêneo.

4. Proposta: Modelo GetLB++

O GetLB++ consiste em duas propostas distintas. A primeira delas é o *framework* de processamento de tarefas e seu modelo básico para sistemas distribuídos. A segunda proposta trata-se de uma evolução do algoritmo de escalonamento de tarefas apresentado pelo GetLB. Esse novo algoritmo atua em conjunto com o *framework* proposto, de modo a otimizar todo o processo de balanceamento de carga. O conjunto dessas duas propostas é chamado de **Modelo GetLB++**. O modelo foi criado visando o paradigma de orientação

à objetos e sua arquitetura define quatro componentes fundamentais para seu funcionamento:

- **Task:** toda a tarefa a ser processada pelo framework deve implementar essa interface, a qual define métodos para buscar dados relacionados ao seu processamento e um método que executa de fato seu processamento.
- **Scheduler:** qualquer algoritmo de escalonamento que implementar essa interface pode ser utilizado pelo GetLB++.
- **Comutador:** possui os dados das MPs e utiliza o algoritmo definido pelo Scheduler para enviar os dados às máquinas.
- **ProcessingMachine:** são as máquinas processadoras (MPs). Suas responsabilidades são processar as tarefas recebidas e atualizar o comutador com o seus dados de carga de acordo com um período ou *threshold* previamente estipulado.

O escalonamento do GetLB++ foi repensado de forma a manter uma fila de recursos ordenada por capacidade de processamento. Sempre que uma nova tarefa chega ao comutador ela é enviada para a máquina do topo da fila e sua nova carga é calculada utilizando as mesmas fórmulas do GetLB. A máquina então é reposicionada na fila, que também é atualizada quando uma máquina faz uma atualização de seu estado ou uma tarefa é finalizada.

5. Avaliação e Resultados

Para a avaliação do modelo GetLB++, foram realizados testes que envolvem mensurar aspectos de velocidade e qualidade do escalonamento utilizando os algoritmos *Round Robin*, GetLB e GetLB++ com a finalidade de realizar uma comparação entre eles. Todos os testes foram realizados utilizando a infraestrutura do *Amazon Elastic Computer Cloud (Amazon EC2)*.

Os resultados do teste de escalabilidade em um cluster homogêneo, presentes na Figura 1, mostram que o escalonador do GetLB++ é em média 6,5 vezes mais rápido que o GetLB. Essa diferença tente a ser maior quanto maior for o *cluster*.

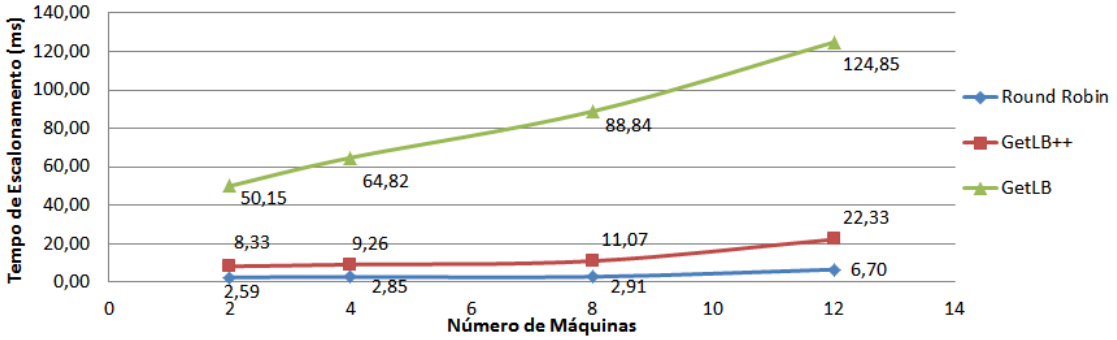

Figura 1. Comparação do tempo de escalonamento

Os testes de qualidade do balanceamento de carga foram realizados utilizando ambientes heterogêneos e homogêneos tando em termos de carga quanto em termos de hardware, visando a métrica de número de transações por segundo. O tempo médio de processamento de 5 mil tarefas é apresentado na Tabela 1, em segundos. Esse tempo de processamento começa quando o escalonador recebe a tarefa em sua fila, e termina

quando ele recebe a confirmação de processamento da MP. As medidas foram realizadas três vezes, e o desvio padrão médio foi de 29 segundos. Nota-se que, com exceção do ambiente totalmente homogêneo, onde o *Round Robin* apresenta o balanceamento de carga ideal, o algoritmo de escalonamento proposto no GetLB++ apresenta os melhores resultados de desempenho.

Tabela 1. Tempo total de processamento das tarefas.

	Cluster Homogêneo		Cluster Heterogêneo	
	Tarefas Homogêneas	Tarefas Heterogêneas	Tarefas Homogêneas	Tarefas Heterogêneas
Round Robin	470,56	1225,13	2371,11	4516,31
GetLB	537,25	826,81	2853,62	2807,98
GetLB++	531,57	764,89	2184,69	2516,26

6. Conclusão

O GetLB é um *framework* com arquitetura capaz de amparar as exigências de eficiência, disponibilidade e segurança de sistemas de processamento de transações eletrônicas de fundos. Porém, o GetLB ainda apresentava alguns efeitos colaterais de sua arquitetura e heurística de escalonamento dinâmicas. Essas características não são desejáveis, principalmente atualmente, onde há uma tendência na substituição de *clusters* especializados e supercomputadores por *clusters* formados por centenas de máquinas simples, a fim reduzir o investimento em infraestrutura, o que torna a escalabilidade propriedade fundamental para as aplicações [Conejero et al. 2014].

Portanto, a contribuição científica desse trabalho é a apresentação do *framework* GetLB++ que, de acordo com os testes realizados, é em média 6,5 vezes mais rápido que o GetLB ao realizar o escalonamento de uma transação e obtém um tempo total de processamento em média 11,78% menor. Além disso, o GetLB++ conta com uma nova arquitetura mais flexível, que permite a utilização de múltiplos algoritmos de escalonamento e o processamento de diversos tipos de tarefas, sendo possível que a mesma implementação seja utilizada para diferentes fins, o que é uma área ainda a ser explorada.

Referências

Conejero, J., Caminero, B., and Carrion, C. (2014). Analysing hadoop performance in a multi-user iaas cloud. In *High Performance Computing Simulation (HPCS), 2014 International Conference on*, pages 399–406.

Righi, R., Costa, C., Gonzaga, L. J., Farias, K., Andrade, A., and Graebin, L. (2014). Redesigning transaction load balancing on electronic funds transfer scenarios. In *Proceedings of the 29th Symposium On Applied Computing*, Gyeongju, Korea.

Singh, B., Singh, R., and Singh Tanwar, P. (2014). Electronic payment systems for online smart cards transaction system. *International Journal of Technology Research and Management*, 1(1).

Sousa, E., Maciel, P., and Araujo, C. (2009). Performability evaluation of eft systems using expolinomial stochastic models. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pages 3328–3333.

Staskauskas, M. (1988). The formal specification and design of a distributed electronic funds-transfer system. *Computers, IEEE Transactions on*, 37(12):1515–1528.