Avaliação de sistemas de Memórias Transacionais de *software*

Fernando Furlan Rui, Luiz Gustavo Fernandes,
Mateus Raeder

Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Av. Ipiranga, 6681 - Prédio 32 - PPGCC
{fernando.rui, mateus.raeder}@acad.pucrs.br, luiz.fernandes@pucrs.br

Introdução

A tecnologia *multicore* provou ser uma solução promissora para o problema de alcançar alto desempenho sem o aumento no consumo de energia. Devido a isso, é esperado que o número de *cores* em processadores continue a aumentar. Entretanto, para que seja utilizada toda a capacidade desta tecnologia, é necessário que as aplicações atinjam um alto grau de concorrência. Neste contexto, o desenvolvimento de aplicações com um alto grau de paralelismo e um correto gerenciamento de sincronização tornou-se uma grande preocupação [RF 10].

Para endereçar essa necessidade de paralelismo surgiram pesquisas em Memórias Transacionais (ver [RF 10]). Memórias Transacionais (TM - *Transactional Memory*) fornecem um mecanismo flexível e simples para programação paralela em processadores *multicore*, usando abstração de alto nível ao contrário do mecanismo de *locks* [RF 10]. Segundo [AJK 09], Memória Transacional é uma promissora abstração de programação concorrente que facilita o desenvolvimento de programas paralelos. Devido a esses fatos, Memória Transacional tornou-se uma área de pesquisa ativa e promissora para simplificar o desenvolvimento de aplicações paralelas altamente escaláveis. Além disso, há um aumento nas atividades de pesquisas na área de sistemas TM, pois a demanda por *softwares* escaláveis está cada vez maior devido às vantagens de sua utilização nas próximas gerações de multiprocessadores.

Memórias Transacionais podem ser implementadas em *hardware* (HTM - *Hardware Transactional Memory)*, em *software* (STM - *Software Transactional Memory)*, ou com combinação de ambas, chamado de híbrida *(HyTM - Hybrid Transactional Memory)*. Dentre as possibilidades, a analisada neste trabalho é a STM, devido as seguintes vantagens: são portáveis sem a necessidade de *hardware* específico, são mais simples de modificar e possuem poucas limitações de arquitetura, além de não possuirem limitação de tamanho de trasanção ou tempo de execução.

Entretanto, ao abstrair o mecanismo de sincronização para o sistema de Memória Transacional, ocorre uma perda de desempenho e importantes decisões de projeto precisam ser tomadas para minimizar tal impacto [RF 10]. Independentemente das decisões de projeto, existem questões semânticas importantes que impossibilitam a programação transacional ideal esperada pela comunidade [CBM 08]. Sistemas TM aumentam consideravelmente o *overhead*, visto que todo seu mecanismo de controle de versão e contenção é implementado em *software* [CD 09]. Segundo [GKV 07], as implementações STM ainda não demonstram que seu *overhead* está em um nível aceitável.

Avaliação de sistemas de Memória Transacional

De acordo com esse avanço das bibliotecas STM na literatura em contraste com a necessidade de melhorias, é necessário avaliar efetivamente tais bibliotecas, de modo a obter resultados reais do quão usável estão os sistemas STM atualmente. Segundo [MCK 08], embora vários sistemas TM tenham sido propostos na literatura, ainda faltam ferramentas e mecanismos necessários para analisar e comparar os trabalhos propostos. Além disso, muitos sistemas TM foram avaliados através de *microbenchmarks* (designados para medir o desempenho de partes específicas de uma aplicação), que não representam o comportamento do mundo real ou de uma aplicação individual, e não exploram uma grande faixa de cenários de execução.

Alguns *benchmarks* foram propostos para avaliar os mecanismos de sistemas STM, como: STMBench7 [GKV 07], STAMP [MCK 08], entre outros. Entretanto, além de *benchmarks*, são necessários outros mecanismos para comparação e análise dos sistemas TM. Segundo [RFL 10], não há um padrão nos resultados de execuções STM devido a ampla variedade de características transacionais. Dessa forma, pode-se notar a dificuldade em analisar efetivamente os resultados gerados através das bibliotecas STM.

Como objetivo desse trabalho, é possível buscar mecanismos mais eficientes para avaliação de bibliotecas STM, analisando mais profundamente o funcionamento e os mecanismos de controle de contenção, versionamento e demais características transacionais. Dessa forma, faz-se necessário algum mecanismo de *tracing* que possa investigar as execuções das implementações STM e coletar dados relevantes para análise. Assim, será possível entender o funcionamento interno de tais sistemas, seus controles de transações, versionamento e contenção, descobrindo, portanto, quais pontos podem ser otimizados.

Referências

[RF 10] Rui, F.; Fernandes, L.. Memórias Transacionais no contexto MPSoCs. Technical Report, 2010

[CBM 08] Cascavel, C.; Blundell, C.; Michael, M.; Cain, H. W.; Wu, P.; Chiras, S.; Chatterjee, S.. Software Transactional Memory: Why is it Only a Research Toy?, Communications of the ACM, Vol 51, No 11, 2008.

[AJK 09] Ansari, M.; Jarvis, K.; Kotselidis, C.; Luján, M.; Kirkham, C.; Watson, I.. Profiling Transactional Memory Applications. In PDP 09: Proceedings of the 17th International Conference on Parallel, Distributed, and Network-based Processing, 2009, pp. 11-20.

[GKV 07] Guerraoui, R.; Kapalka, M.; Vitek, J.. STMBench7: A Benchmark for Software Transactional Memory. In Proceedings of the Second European Systems Conference EuroSys, Lisboa, Portugal, 2007.

[CD 09] Castro, M.; Degomme, A.. Transactional Memory: state of the art and trends, Techinal Report, 2009

[MCK 08] Minh, C. C.; Chung, J.; Kozyrakis, C.; Olukotun, K.. STAMP: Stanford Transactional Applications for Multi-Processing. In IISWC 08: Proceedings of The IEEE International Symposium on Workload Characterization, Sept. 2008.

[RFL 10] Rui, F.; Fernandes, L.. Avaliação de Bibliotecas STM com Benchmark STAMP. Technical Report, 2010