AVALIAÇÃO DE TEMPO DO BENCHMARK LEE-TM ADAPTADO A TINYSTM

MICHAEL ALEXANDRE COSTA[1*]; FELIPE LEIVAS TEIXEIRA[1†]; ANDRÉ RAUBER DU BOIS[1]; MAURICIO LIMA PILLA[1]

[1]*Universidade Federal de Pelotas – {macosta, flteixeira, dubois, pilla}@inf.ufpel.edu.br*

1. INTRODUÇÃO

A programação concorrente tem ganhado importância na computação devido a evolução dos computadores com múltiplos processadores. Isto porque a programação concorrente explora o paralelismo conseguindo um desempenho melhor. Entretanto, existe o problema de condição de corrida quando se compartilha a memória. Esse problema consiste em várias *threads* ou processos acessando e manipulando os mesmos dados concorrentemente com leituras e escritas. Sem sincronização e o resultado destes acessos pode variar de execução para execução, sem determinismo.

As memórias transacionais, ou *Software Transactional Memories* (STM) (Herlihy et al. 1993), são alternativas à sincronização com *locks* e monitores. As memórias transacionais são baseadas em um conceito similar às transações de banco de dados. O programador declara os acessos às variáveis compartilhadas em transações e o sistema de execução é responsável por garantir a exclusão mútua. Suas vantagens são, entre outras, a facilidade de programação e a inexistência de *deadlock*.

O *Lee-TM* é um *benchmark* de memórias transacionais baseado no algoritmo de roteamento *lee* (Ansari et al. 2008). O algoritmo funciona da seguinte forma: são recebidos dois pontos, por vez, como entrada e o algoritmo encontra um caminho entre estes pontos, esse caminho não pode ser interceptado por um caminho já existente. O *Lee-TM* trata cada tentativa de estabelecer um caminho como uma transação, que pode ou não ser confirmada.

A TinySTM (Felber et al. 2008), é uma implementação de STM para as linguagens C e C++. Seu algoritmo é baseado em outros algoritmos de STM como o TL2 (Dice; Shalev; Shavit et al. 2006). Ela é uma biblioteca utilizada para escrever aplicativos que usam memórias transacionais para sincronização, em substituição aos tradicionais *locks*.

Neste artigo, será avaliado o desempenho de tempo entre a última versão do *benchmark Lee-TM* para *TinySTM* de 2007 e uma adaptação do *Lee-TM* para a nova versão da *TinySTM* de 2014.

O artigo é dividido da seguinte forma: A seção 2 apresenta as metodologias utilizadas. A seção 3 mostra os resultados e discussões do trabalho. A seção 4 apresenta as conclusões obtidas.

2. METODOLOGIA

Para este trabalho, foi feita uma adaptação do *benchmark Lee-TM* para a versão mais recente da biblioteca de memórias transacionais *TinySTM*. O trabalho tem como objetivo, comparar o tempo de execução do *benchmark Lee-TM* adaptado a versão mais recente da *TinySTM* e do *benchmark Lee-TM* com a anti-

[*]Bolsista de Iniciação Científica da FAPERGS
[†]Bolsista de Mestrado da FAPERGS

ga versão *TinySTM.* No decorrer do artigo será utilizado a nomenclatura de ***New Lee-TM***, para a adaptação do *benchmark Lee-TM* para a versão mais recente da biblioteca *TinySTM,* e ***Old Lee-TM*** para o *benchmark Lee-TM* com a versão antiga da biblioteca *TinySTM*.

O *Lee-TM* é um *benchmark* caracterizado por *workloads* extensos e realistas. Este é baseado no algoritmo de *Lee*, que se propõem a encontrar a menor rota entre dois pontos (Ansari et al. 2008). Cada rota encontrada é vista pelo *benchmark* como uma transação. Ao estabelecer um caminho entre dois pontos, o Lee-TM tentara confirmar a transação. Caso este caminho seja interceptado por outro ou outro caminho seja confirmado primeiro, ocorrera um *abort* nesta transação.

A nova versão do *Lee-TM* foi adaptada para a *TinySTM* versão 1.0.5 para permitir o uso de novas funcionalidades e melhorias da *TinySTM*. A antiga versão do *Lee-TM* utiliza a biblioteca *TinySTM* versão 0.7.3.

Foram realizadas 30 execuções para cada um dos cinco testes dos *benchmarks*, em cenários de 1, 2, 4, 8, 16, 32 e 64 *threads*. Os tempos de execuções foram medidos pelo *benchmark Lee-TM*. Os testes foram realizados em uma máquina com processador Opteron com 4 nós de 16 cores cada, somando-se um total de 64 cores. No melhor caso a variação dos resultados definida pelo desvio-padrão dividido pela média dos resultados dos experimentos foi de aproximadamente 0%, que ocorreu no teste *sparselong* com 16 *threads* Old Lee-TM, e no pior caso, 25%, que ocorreu no *testboard* com 4 *threads* Old Lee-TM.

3. RESULTADOS E DISCUSSÃO

Os resultados obtiveram valores diferentes em cada um dos testes. Como pode ser visto na Figura 1, todos os testes do *benchmark New Lee-TM* mostraram um melhor desempenho em relação aos testes do *Old* Lee-TM.

O *New Lee-TM* apresenta uma queda de tempo com relação ao *Old Lee-TM*, no melhor caso, ocorrido no teste *memboard* 4 *threads* de aproximadamente 84%, e em seu pior caso, ocorrido no teste *sparseshort* com 64 *threads*, de aproximadamente 12%, este pior caso pode ser explicado devido ao tempo de execução ser muito baixo nas duas versões do *benchmark*.

O ganho de desempenho apresentado no *New Lee-TM* está relacionado a função *Thread Local Storage* (TLS) que está implementada nas versões mais recentes da TinySTM. A TLS armazena informações de *threads* em variáveis locais assim substituindo o armazenamento em variáveis globais (Yair Shade and Shaham et al. 2005), fazendo com que não sejam serializados os acessos a estas variáveis, assim proporcionando um ganho de desempenho.

Outra vantagem é que ao armazenar as informações em variáveis locais, o custo de acesso a estes dados se tornam menores que o custo de acesso com sincronização das *threads* as variáveis globais, assim a TLS contribui com um custo menor para a TinySTM, obtendo o ganho de desempenho apresentado.

Podemos notar um maior tempo de execução nos testes, *mainboard* e *memboard* para as duas versões do *benchmark*, isto ocorre devido ao maior número de dados na entrada destes testes.

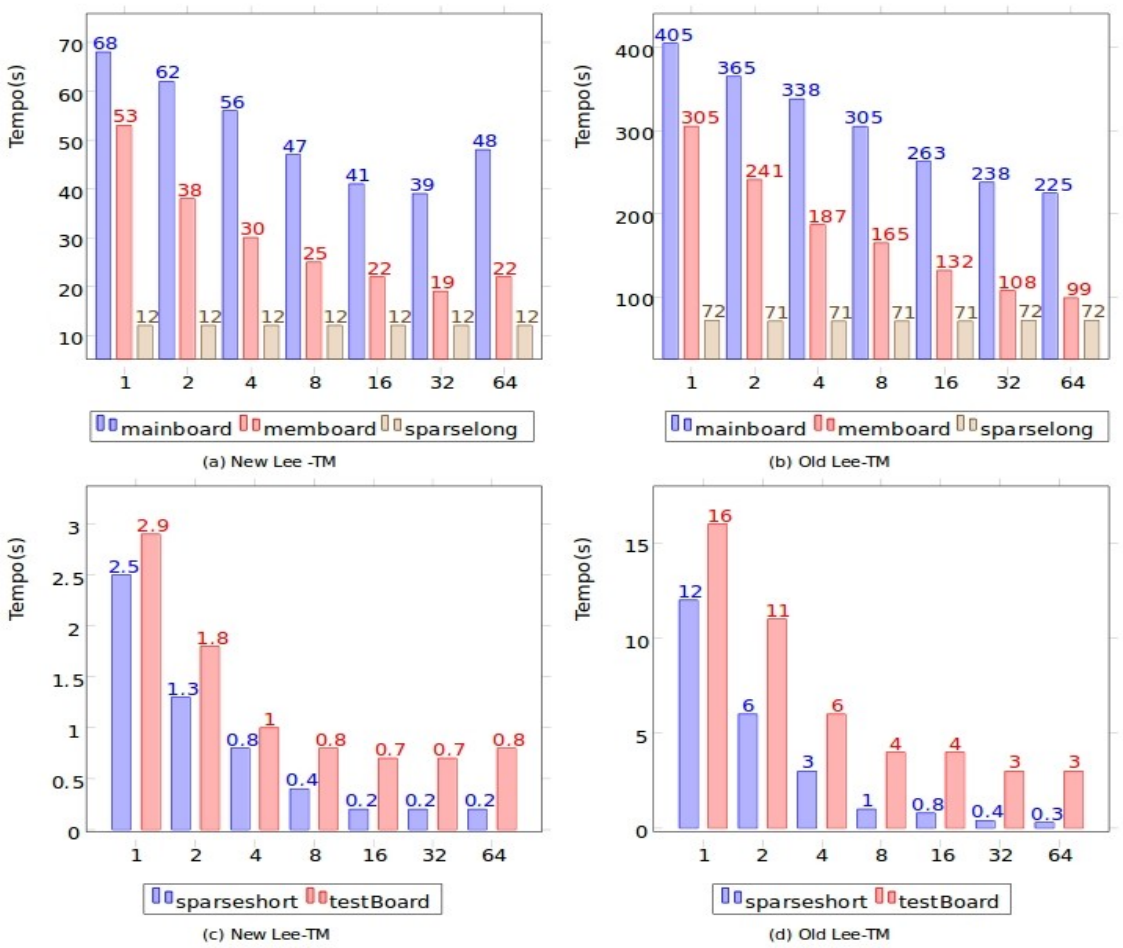

Figura 1: Gráficos das Médias de Tempo dos *Benchmarks Lee-TM*

4. CONCLUSÕES

Neste trabalho foi implementado uma adaptação do *benchmark Lee-TM* para a versão mais recente da biblioteca de memórias transacionais TinySTM e analisado o desempenho de tempo, comparando-o com a versão mais antiga do *benchmark Lee-TM* utilizando a biblioteca *TinySTM*. Os resultados mostraram um ganho de desempenho de até 84% para a adaptação feita. Esse ganho de desempenho ocorre pelo uso da TLS, na nova versão da TinySTM. O uso da TLS substitui o armazenamento das informações de cada *thread,* antes armazenado em uma variável global para as variáveis locais em cada *thread*, assim reduzindo seu tempo de execução por não precisar sincronizar o acesso a variável global.

Para trabalhos futuros, pretende-se fazer uma análise do número de *aborts* e de outras características do *benchmark New Lee-TM* comparando com o Old Lee-TM, para avaliar o comportamento do Lee-TM utilizado junto com a TinySTM. Também será feita uma comparação entre a adaptação implementada neste artigo com outras bibliotecas de memórias transacionais.

5. REFERÊNCIAS BIBLIOGRÁFICAS

Ansari, M.; Kotselidis, C.; Watson, I.; Kirkham, C.; Luján, M.; Jarvis, K. Lee-tm: A non-trivial benchmark suite for transactional memory. In: **8TH INTERNATIONAL CONFERENCE ON ALGORITHMS AND ARCHITECTURES FOR PARALLEL PROCESSING,** 1., 2008, **Anais...** Springer: 8TH ICA3PP, 2008.

Dice, D.; Shalev, O.; Shavit, N. Transactional Locking II. In: **DISC**, 2006, **Anais...** 20TH International Symposium DISC 2006, 2006.

Felber, P.; Fetzer, C.; Riegel, T. Dynamic performance tuning od word-based software transactional memory. In: **PPoPP '08: 13TH ACM SIGPLAN SYMPOSIUM ON PRINCIPLES AND PRACTICE OF PARALLEL PROGRAMMING,** 2., New York, 2008, **Anais...** New York: 13TH ACM SIGPLAN Symposium, 2008. p.237-246.

Herlihy, M.; Eliot, J.; Moss, B. Transactional Memory: Architectural support for lock-free data structures. In: **20TH ANNUAL SYMPOSIUM ON COMPUTER ARCHITECTURE,** 3., 1993, **Anais...** 20TH Annual international Symposium on Computer Architecture, 1993. p.289-300.

Yair Sade, M. S. and Shaham, R. Optimizing e Multithreaded Memory Management Using Thread-Local Storage. In: **14TH INTERNATIONAL CONFERENCE CC,** 2005, **Anais...** 14TH International Conference CC, 2005.