Análise do Consumo de Energia e Desempenho de Memórias Transacionais em *Software* em Cenário de Alta Contenção*

Timóteo M. Rico[1], Maurício L. Pilla[1], André R. Du Bois[1], Rodrigo M. Duarte[1]

[1]Universidade Federal de Pelotas (UFPel)
Programa de Pós-Graduação em Computação (PPGC)

{tmrico,pilla,dubois,rmduarte}@inf.ufpel.edu.br

Resumo. *Este trabalho avalia o consumo de energia e desempenho das bibliotecas AdaptSTM e SwissTM, utilizando-se aplicações com alta contenção em ambiente não-simulado. Resultados mostram a SwissTM como a abordagem mais eficaz na maioria dos casos, exceto na execução com 32 threads.*

1. Introdução

Uma das principais dificuldades em programação concorrente é garantir a correta sincronização entre as *threads*, evitando assim condições de corrida [Rico et al. 2012]. Um novo mecanismo de sincronização, denominado Memória Transacional (TM), foi desenvolvido com objetivo de reduzir as dificuldades e limitações encontradas em tradicionais métodos de sincronização [Harris et al. 2010]. Por se tratar de uma alternativa recentemente proposta, pouco se conhece a respeito do consumo de energia devido ao uso de TM, em especial, em ambiente computacional não-simulado.

Nesse contexto, este trabalho analisa o consumo de energia e desempenho de duas bibliotecas de TM em *Software* (STM), AdaptSTM [Payer and Gross 2011] e SwissTM [Dragojevic et al. 2009], em um sistema de computação real.

2. Memórias Transacionais em *Software*

As STMs garantem a execução atômica e isolada das transações (seção crítica) concorrentes através de dois mecanismos chave: *(i)* detecção/resolução de conflitos e *(ii)* versionamento de dados [Harris et al. 2010]. Há ocorrência de conflitos entre transações concorrentes quando no mesmo intervalo de tempo duas ou mais transações acessam o mesmo dado compartilhado e ao menos um dos acessos é de escrita. Em caso de conflito, um componente do sistema transacional, denominado gerenciador de contenção, é invocado para resolver os conflitos e garantir o progresso do sistema. O versionamento de dados lida com o gerenciamento das diferentes versões dos dados (originais e especulativas) manipulados por uma transação. Em casos de efetivação da transação, os dados especulativos são armazenados na memória, caso contrário permanecem os originais e a transação é reexecutada. Na Tabela 1 são sintetizadas as principais características das STMs utilizadas no presente trabalho.

3. Metodologia e Resultados

A avaliação das STMs baseou-se nas aplicações Intruder e Yada do *benchmark* STAMP [Payer and Gross 2011], com os parâmetros de execução sugeridos para sistemas

*Projeto financiado pelo CNPq e FAPERGS

Tabela 1. Características das STMs AdaptSTM e SwissTM.

STM	Versionamento de dados	Detecção de conflitos	Gerenciador de contenção
AdaptSTM	adiantado (baixa contenção) e atrasado (alta contenção)	adiantado	tímido (baixa contenção) e *backoff* exponencial (alta contenção)
SwissTM	atrasado	adiantado (transações escrita-escrita) e atrasado (transações leitura-escrita)	tímido (transações curtas) e guloso (transações complexas)

não-simulados. O consumo de energia (em *Joules*) foi coletado através de um microcontrolador embutido na placa-mãe, denominado *Baseboard Management Controller*, via IPMI. O tempo de execução em segundos foi mensurado pelo resultado de saída do *benchmark*. Os testes foram realizados em um sistema computacional com 2 processadores Intel Xeon E5620 com 6 GB de memória RAM cada nodo, sistema operacional SUSE Linux SP11 e G++ 4.5.2. Cada configuração foi executada 10 vezes e os dados de consumo de energia e desempenho foram integrados e as médias calculadas. O desvio padrão foi pequeno em relacao à média, sempre menor que 4,22% no consumo de energia e menor que 4,4% no tempo de execução.

Conforme os resultados na Tabela 2, as STMs obtiveram pouca melhora no desempenho de 4 para 8 *threads*, e acima de 8 *threads* degradou-se o desempenho. Esse comportamento é devido à alta contenção presente nas aplicações Intruder e Yada. Conforme aumenta-se o número de *threads*, maior será a taxa de conflitos, degradando consequentemente o desempenho e o consumo de energia. Note-se que há apenas 8 *cores* no sistema, portanto espera-se contenção em aplicações *cpu-bound* com mais que 8 *threads*.

De modo geral, a SwissTM apresentou os melhores resultados em consumo de energia e desempenho. No entanto, na execução com 32 *threads* a SwissTM não apresentou bons resultados em relação à AdaptSTM, sendo em média cerca de 145% menos eficiente no consumo de energia e desempenho. Este resultado mostra uma boa escalabilidade da AdaptSTM, devido principalmente à adaptação de suas estratégias de versionamento de dados e gerenciador de contenção em tempo de execução.

Tabela 2. Resultados das execuções das STMs nas aplicações Intruder e Yada.

	2 *threads*		4 *threads*		8 *threads*		16 *threads*		32 *threads*	
STM	tempo	energia	tempo	energia	tempo	energia	tempo	energia	tempo	energia
Intruder										
AdaptSTM	21,454	3081,230	12,272	1881,631	9,613	1587,870	12,259	2072,383	**12,785**	**2197,575**
SwissTM	**17,781**	**2558,977**	**10,705**	**1645,275**	**8,075**	**1358,816**	**8,897**	**1564,135**	27,692	5040,432
Yada										
AdaptSTM	12,555	1844,757	8,500	1301,551	7,617	1229,918	7,955	1333,872	**10,365**	**1789,602**
SwissTM	**10,031**	**1473,683**	**6,245**	**942,731**	**4,509**	**689,491**	**5,961**	**939,697**	28,204	4255,435

Referências

Dragojevic, A., Guerraoui, R., and Kapalka, M. (2009). Stretching transactional memory. In *PLDI 2009*, pages 155–165, New York. ACM.

Harris, T., Larus, J., and Rajwar, R. (2010). Transactional memory. *Synthesis Lectures on Computer Architecture*, 5(1):1–263.

Payer, M. and Gross, T. (2011). Performance evaluation of adaptivity in software transactional memory. In *ISPASS 2011*, pages 165–174. IEEE.

Rico, T., Pilla, M., and Du Bois, A. (2012). Energy consumption on software transactional memories. In *WSCAD-SSC 2012*, pages 194–201, Petrópolis. IEEE.