ESTIMATIVA DOS CONSUMO DE ENREGIA DAS ESCRITAS DAS STM EM UMA MEMÓRIA PCM

FELIPE LEIVAS TEIXEIRA[1]; ANDRÉ RAUBER DU BOIS[1]; MAURÍCIO LIMA PILLA[1]

[1]*Universidade Federal de Pelotas (UFPEL)*
Computação - CDTec
Caixa Postal 354 · CEP 96001-970 · Pelotas, RS - Brasil
{flteixeira, dubois, pilla}@inf.ufpel.edu.br

*Bolsista FAPERGS

1. INTRODUÇÃO

O consumo de energia é um fator cada vez mais importante nos grandes *data centers*. A *Phase Change Memory* (PCM) (LEE et al., 2010) surgiu como uma alternativa para o consumo proibitivo de energia em memórias tradicionais, pois após o armazenamento de um bit não há consumo de energia para mantê-lo como em memórias RAM estáticas ou dinâmicas.

O problema das memórias PCM está em suas escritas, que são lentas, visto que o processo de armazenamento de um bit envolve alterar o estado do material da célula de memória em questão (LEE et al., 2010). Além das escritas serem lentas, elas desgastam o material devido à mudança de fase que fazem para armazenar os dados, diminuindo sua vida útil.

As Memórias Transacionais (HERLIHY et al., 1993) ou *Software Transactional Memories* (STM) são uma alternativa para sincronização de *threads*. O conceito por trás de STM é baseado em transações de banco de dados, assim tendo como uma vantagem sobre as sincronizações baseadas em *locks* a simplicidade de escrita de código. Outra vantagem das STMs é que não existe o problema de *deadlock* que ocorre na sincronização baseada em *locks* (HARRIS et al., 2010). A composição de componentes de softwares também é mais simples do que em sistemas baseados em *locks*, aumentando a reusabilidade do código (HARRIS et al., 2010). Memórias Transacionais são uma alternativa promissora para a escrita de código portável e escalável em memória compartilhada.

Nesse trabalho, foi feita uma estimativa do consumo de energia das escritas de *benchmarks* de STM quando executadas em uma memória PCM. Para tanto, a biblioteca de STM TinySTM (FELBER et al., 2008), a ferramenta PinTools (LUK et al., 2005) e o *benchmark* STAMP (CAO MINH et al., 2008) foram usados para gerar um ambiente de avaliação.

O artigo é dividido da seguinte forma: a Seção 2 apresenta a metodologia. A Seção 3 mostra os resultados obtidos. A Seção 4 discute as conclusões.

2. METODOLOGIA

Para a execução deste trabalho, foi implementada uma instrumentação de código utilizando a ferramenta Pintools. Para a implementação da instrumentação foi desenvolvido um programa em C++ que faz uma avaliação de quantos bits foram trocados na escrita e se ocorreu um *SET* ou um RESET, no bit modificado.

A implementação foi feita da seguinte forma: a cada escrita na memória é armazenado o endereço de memória que está sendo escrita e o valor que será escrito nela. Com o endereço de memória é obtido o valor atual que está armazenado em um vetor e é feita uma comparação com o valor que será escrito naquele endereço, de modo a verificar quantos bits foram modificados e também verificar se ocorreu um *SET* ou *RESET* em cada bit que foi modificado. Essa comparação é feita em tempo de execução. Logo depois de fazer essa comparação, o vetor é atualizado com o valor escrito.

3. RESULTADOS E DISCUSSÃO

Os *benchmarks* foram executados em uma máquina com processador Core i7 2600 4 cores físicos e 8 cores lógicos, com memória RAM de 8GiB. Para cada *benchmark* do STAMP e configuração foram medidas dez execuções para cada cenário com 1, 2, 4 e 8 *threads* com a instrumentação implementada.

Tabela 1: Tabela do número de SET's (em milhões)

	1 Thread	2 Threads	4 Threads	8 Threads
Bayes	1.512	1.517	1.511	1.506
nome	579	528	531	566
Intruder	2.811	3.051	3.329	3.984
Kmeans	4.656	4.629	4.663	4.804
Labyrinth	4.888	5.245	6.130	7.413
SSCA2	690	695	700	693
Vacation	4.619	4.653	4.648	4.674
Yada	6.474	7.229	8.072	8.976

A Tabela 1 mostra a média de *SETs* que ocorreram nos testes. Como pode ser visto na tabela a média de *SETs* varia de acordo com o *benchmark*. Em alguns *benchmarks* a média de *SETs* aumenta conforme o número de *threads* aumenta. Este resultado é esperado, pois aumentando o número de *threads* concorrentes, o número de escritas deve aumentar se não houver contenção. Já em outros a média varia de acordo com o número de *threads*. Um fator que pode explicar essas variações é a biblioteca TinySTM, que implementa um gerenciador de contenção tímido, que cancela a transação assim que foi detectado o conflito. Com isso tendo muitos cancelamentos em sua execução.

Tabela 2: Tabela do número de RESET's (em milhões)

	1 Thread	2 Threads	4 Threads	8 Threads
Bayes	1.350	1.355	1.350	1.344
Genome	555	504	507	542
Intruder	2.750	2.994	3.274	3.922
Kmeans	4.622	4.596	4.630	4.770
Labyrinth	4.878	5.235	6.108	7.363
SSCA2	616	619	625	614
Vacation	4.553	4.590	4.584	4.610
Yada	6.305	7.049	7.877	8.763

Conforme pode ser visto na Tabela 2 o comportamento da média de *RESETs* foi muito parecido com o comportamento da média de *SETs*.

Tabela 3: Tabela com o consumo de energia dos testes (em miliJoule)

	1 Thread	2 Threads	4 Threads	8 Threads
Bayes	46,4	46,5	46,3	46,1
Genome	18,5	16,8	16,9	18,1
Intruder	90,8	98,7	107,8	129,1
Kmeans	151,6	150,7	151,9	156,5
Labyrinth	159,7	171,3	200,1	241,5
SSCA2	21,2	21,3	21,5	21,2
Vacation	149,8	150,9	150,7	151,6
Yada	208,5	232,9	260,3	289,4

O cálculo para fazer a estimativa do consumo de energia das escritas das STM foi feita da seguinte forma: o número de *SETs* foi multiplicado pelo consumo de enegia de um *SET*, já o número de *RESETs* foi multiplicado pelo consumo de engergia de um *RESET* e no fim foi somado os dos resultados para obter uma estimativa do consumo de energia das escritas dde STM em uma memória PCM.

Os valores utilizados para calcular uma estimativa do consumo de energia utilizados no trabalho foi o do artigo (LEE et al., 2009). A Tabela 3 mostra quanto de energia foi consumida nas escritas dos *benchmarks*. O comportamento do consumo de energia é similar ao comportamento da média de *SETs* e de *RESETs*.

4. CONCLUSÕES

Neste artigo, foi estimado o consumo de energia das escritas de *benchmarks* de STM. Como *benchmark* foi usado o STAMP. Como ele implementa vários *benchmarks*, uma ampla área de aplicação das STMs é coberta. Com isso foi possível analisar as STMs em várias situações diferentes.

Os resultados mostraram um mesmo comportamento, de acordo com o número de *threads*, para a média de *SETs*, a média de *RESETs* e o consumo de energia. Além do fato interesssante que em todas as execuções o número de *SETs* foi maior que o número de *RESETs*.

Como trabalhos futuros, pretende-se fazer uma comparação da estimativa que foi obtida neste trabalho e comparar com o consumo de energia das escritas das STM em outras memórias.

5. REFERÊNCIAS BIBLIOGRÁFICAS

LEE, B. C.; ZHOU, P. Z. P.; YANG, J. Y. J.; ZHANG, Y. Z. Y.; ZHAO, B. Z. B.; IPEK, E.; MUTLU, O. and BURGER, D., “Phase-change technology and the future of main memory,” **IEEE Micro, vol. 30**, no. 1, pp. 131–141, 2010.

HERLIHY, M.; ELIOT, J. and MOSS, B., “Transactional memory: Architectural support for lock-free data structures,” in **Proceedings of the 20th Annual International Symposium on Computer Architecture**, 1993, pp. 289–300.

HARRIS, T.; LARUS, J. and RAJWAR, R., “Transactional memory, 2nd edition,” **Synthesis Lectures on Computer Architecture, vol. 5**, no. 1, pp. 1–263, 2010.

FELBER, P.; FETZER, C. and RIEGEL, T., “Dynamic performance tuning of word-based software transactional memory,” in **PPoPP ’08: Proceedings of the 13th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming**. New York, NY, USA: ACM, 2008, pp. 237–246.

LUK, C. K.; COHN, R.; MUTH, R.; PATIL, H.; KLAUSER, A.; LOWNEY, G.; WALLACE, S.; JANAPA; V. and HAZELWOOD, R. K., “Pin: Building customized program analysis tools with dynamic instrumentation,” In **Programming Language Design and Implementation. ACM Press**, 2005, pp. 190–200.

CAO MINH, C.; CHUNG, J.; KOZYRAKIS, C. and OLUKOTUN, K., “STAMP: Stanford transactional applications for multiprocessing,” in **IISWC ’08: Proceedings of The IEEE International Symposium on Workload Characterization**, September 2008.

LEE B. C.; IPEK E.; MUTLU O. and BURGER D., “Architecting phase change memory as a scalable dram alternative,” in **Proceedings of the 36th annual international symposium on Computer architecture, ser. ISCA ’09**. New York, NY, USA: ACM, 2009, pp. 2–13.