# Impacto de uma arquitetura híbrida, com Memória PCM e Memória DRAM, em Memórias Transacionais*

**Felipe L. Teixeira[1]†, Rodolfo M. Favaretto[1], Maurício L. Pilla[1], André R. Du Bois[1]**

{flteixeira, rmfavaretto, pilla, dubois}@inf.ufpel.edu.br

[1]Universidade Federal de Pelotas (UFPEL)
Computação - CDTec

***Resumo.*** *Memória PCM e Memórias Transacionais são novas alternativas em suas áreas e neste artigo, propõe-se analisar o impacto de uma arquitetura híbrida com memória PCM e memória DRAM, em* benchmarks *de memórias transacionais. Para o trabalho foi desenvolvida uma biblioteca com a ferramenta* PinTools*. Os resultados demonstram o comportamento esperado porém com alguma variação no tempo de execução..*

## 1. Introdução

Memórias Transacionais [Herlihy et al. 1993] são uma alternativa para sincronização de *threads*. As memórias transacionais são baseadas em transações de banco de dados. Uma vantagem das memórias transacionais é que como elas executam transações, não existe o problema de *deadlock* encontrado na sincronização baseada em *locks* [Harris et al. 2010].

*Phase Change Memories* (PCM) [Lee et al. 2010] são uma nova alternativa às memórias baseadas em *Dynamic Random Access Memories* (DRAM) utilizadas atualmente como memória principal. Memórias PCM possuem vantagens em termos de consumo de energia e em seu tempo de leitura. Porém, tempos de escrita são elevados em comparação.

O objetivo deste trabalho é analisar o impacto que uma arquitetura híbrida com uma memória PCM, como memória principal, e uma memória DRAM, como uma cache da memória PCM, tem sobre as memórias transacionais. Para a realização desse trabalho foi desenvolvido uma biblioteca que simula uma arquitetura híbrida com a ferramenta Pin Tools [keung Luk et al. 2005]. A biblioteca de memórias transacionais utilizada foi a *TinySTM* [Felber et al. 2008] e o *benchmark* de memórias transacionais utilizado foi o STAMP [Cao Minh et al. 2008]. Este trabalho é a continuação do trabalho apresentado em [Teixeira et al. 2012], agora verificando o impacto de diferentes percentuais de escrita na memória no desempenho.

O artigo apresenta-se da seguinte forma: a Seção 2 apresenta os fundamentos de Memórias Transacionais e *Phase-Change Memory*. A Seção 3 traz o ambiente de simulação utilizado. A Seção 4 apresenta a instrumentação realizada para desenvolver as simulações. Após, a Seção 5 mostra os resultados obtidos. Finalmente, a Seção 6 discute as conclusões e trabalhos futuros.

*Projetos PRONEX/FAPERGS/CNPq GREEN-GRID Computação de Alto Desempenho Sustentável e Composição de Ações Transacionais (Pesquisador Gaúcho/FAPERGS).

†Bolsista PIBIC/CNPq

## 2. Trabalhos Relacionados

### 2.1. Memórias Transacionais

Memória Transacional, ou *Transactional Memory* (TM), é um tipo de sincronização que fornece uma execução atômica e isolada de partes do código compartilhado. As TMs estão sendo estudadas para que num futuro próximo se tornem o principal meio de fazer a sincronização em um programa concorrente, substituindo a sincronização baseada em *locks* [Moreshet et al. 2006]. Neste artigo focou-se em STMs (Memórias Transacionais em *software*).

Transação é uma sequência finita de escritas e leituras na memória executada por uma *thread* [Herlihy et al. 1993], e que deve satisfazer duas propriedades, a propriedade de isolamento e de atomicidade [Herlihy et al. 1993].

### 2.2. *Phase Change Memory* (PCM)

A *Phase Change Memory* (PCM) é uma memória não volátil que vem sendo estudada para que num futuro seja utilizada como memória principal dos computadores, substituindo as memórias DRAM [Ferreira et al. 2010]. Isto se dá pelo seu custo-benefício e a sua eficiência em relação ao consumo de energia. A PCM utiliza-se das fases de seu material para armazenar dados.

A grande vantagem da memória PCM é o seu tempo de leitura que é mais rápido que o de uma DDR3. O principal problema das PCMs diz respeito ao desgaste do seu material, devido à mudança de fase do material no momento da escrita.

## 3. Ambiente de Simulação

Para executar o *benchmark* STAMP foram utilizadas a *TinySTM* e a *PinTools*, que serão descritas a seguir.

A *TinySTM* [Felber et al. 2008] é uma implementação de memórias transacionais para as linguagens C e C++. Seu algoritmo é baseado em outros algoritmos de TM como o TL2 [Dice et al. 2006]. A versão do *TinySTM* utilizada foi a 1.0.3.

Pin [keung Luk et al. 2005] é uma ferramenta de instrumentação dinâmica de programas. O Pin foi projetado para fornecer uma funcionalidade onde um código escrito em C ou em C++, pode ser inserido em um executável. A versão do Pin utilizada foi a 2.11.

STAMP [Cao Minh et al. 2008] é um conjunto de *benchmarks* criado para pesquisa de memórias transacionais. O STAMP é composto por oito *benchmarks*. A versão do STAMP utilizada foi a 0.9.10.

## 4. Biblioteca de Simulação de PCM

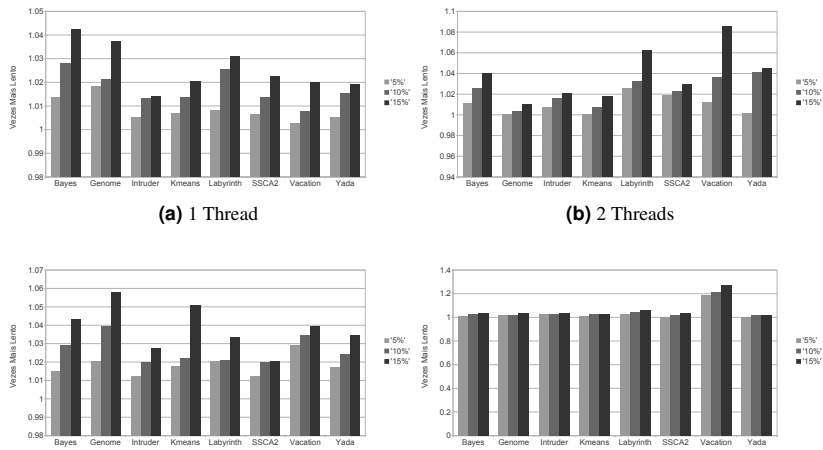
Para a realização do trabalho foi implementado uma biblioteca para simular o tempo de escrita de memórias PCM. Essa biblioteca foi implementada utilizando a ferramenta *Pintools*. Para a implementação da biblioteca foi desenvolvido um programa em C++ que avaliando uma probabilidade ele inseria ou não um atraso a cada escrita feita à memória, assim simulando uma memória PCM com uma memória DRAM de cache. O *delay* é inserido devido ao tempo de escrita da memória PCM ser maior que o tempo de escrita na memória DRAM.

## 5. Resultados

Os benchmarks foram executados em uma máquina com dois processadores Intel(R) Xeon(R) CPU E5620 @2.40GHz com 12Mb de cache cada. Cada programa do *benchmark* STAMP, foi executado com suas configurações recomendadas para simulação, os *benchmarks Vacation* e *Kmeans* foram executados com suas configurações para simulação com uma alta contenção do código.

Para cada *benchmark* e configuração, foram medidas dez execuções com 1, 2, 4 e 8 *threads* com a biblioteca implementada inserindo o atraso em 5, 10 e 15% nas escritas. Foi comparado quantas vezes mais lenta a implementação inserindo o atraso foi em relação à implementação que não inseriu o atraso.

A avaliação experimental demonstrou, como pode ser visto na Figura 1, que os *benchmarks* obtiveram um diferença esperado, quanto maior a porcentagem de escrita na Memória PCM maior o tempo, mas em relação à proporção de aumento de tempo que era esperado, alguns não obtiveram uma diferença esperada.




**(a)** 1 Thread **(b)** 2 Threads

**(c)** 4 Threads **(d)** 8 Threads

**Figura 1. Desempenho de memórias transacionais em uma arquitetura híbrida, com memória PCM e memória DRAM.**

As menores diferenças foram mostradas nos *benchmarks Intruder* na Figura 1a, *Yada* na Figura 1b, *SSCA2* e *Labyrinth* na Figura 1c. Já os que mostraram uma maior diferença foram o *Genome* e o *Vacation* na Figura 1a, o *Labyrinth* e o *Vacation* na Figura 1b e o *Kmeans* na Figura 1c. O *benchmark* que obteve o pior desempenho foi o *Vacation* na Figura 1d, que mostrou um tempo mais elevado que os demais.

O melhor coeficiente de variação foi de 0,04%, que ocorreu no experimento *Bayes* com 2 *threads* com inserção do atraso em 5% das escritas, e o pior foi de 7,6%, que ocorreu no experimento *Labyrinth* com 4 *threads* sem a inserção do atraso.

## 6. Conclusões e Trabalhos Futuros

Neste artigo, o impacto de uma arquitetura híbrida com memória PCM e memória DRAM em *benchmarks* de memórias transacionais em software foi estudado. O objetivo do trabalho foi mostrar o impacto que o tempo de escrita elevado da Memória PCM tem sobre as Memórias Transacionais, levando em conta uma maior hierarquia de memória, com uma Memória DRAM como cache da Memória PCM. Essa maior hierarquia é importante visto que muitas escritas na Memória PCM diminuem seu tempo de vida [Ferreira et al. 2010].

Os *benchmarks* mostraram um desempenho esperado, mas alguns mostraram uma diferença, em relação a proporção de escritas na memória PCM, maior e outros menor. Essas diferenças podem ser justificadas devido a implementação da biblioteca.

Como trabalhos futuros pretende-se utilizar um simulador de cache para analisar uma maior hierarquia. Também pretende-se verificar o número de bits que são modificado a cada escrita na memória PCM para verificar o desgaste que a memória PCM sofre.

## Referências

Cao Minh, C., Chung, J., Kozyrakis, C., and Olukotun, K. (2008). STAMP: Stanford transactional applications for multi-processing. In *IISWC '08: Proceedings of The IEEE International Symposium on Workload Characterization*.

Dice, D., Shalev, O., and Shavit, N. (2006). Transactional locking II. In *DISC 2006*, pages 194–208.

Felber, P., Fetzer, C., and Riegel, T. (2008). 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*, pages 237–246, New York, NY, USA. ACM.

Ferreira, A. P., Zhou, M., Bock, S., Childers, B., Melhem, R., and Mossé, D. (2010). Increasing pcm main memory lifetime. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '10, pages 914–919, 3001 Leuven, Belgium, Belgium. European Design and Automation Association.

Harris, T., Larus, J., and Rajwar, R. (2010). Transactional memory, 2nd edition. *Synthesis Lectures on Computer Architecture*, 5(1):1–263.

Herlihy, M., Eliot, J., and Moss, B. (1993). Transactional memory: Architectural support for lock-free data structures. In *in Proceedings of the 20th Annual International Symposium on Computer Architecture*, pages 289–300.

Chi-Keung Luk, C., Cohn, R., Muth, R., Patil, H., Klauser, A., Lowney, G., Wallace, S., Janapa, V., and Hazelwood, R. K. (2005). Pin: Building customized program analysis tools with dynamic instrumentation. In *In Programming Language Design and Implementation*, pages 190–200. ACM Press.

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. (2010). Phase-change technology and the future of main memory.

Moreshet, T., Bahar, R. I., and Herlihy, M. (2006). Energy-aware microprocessor synchronization: Transactional memory vs. locks.

Teixeira, F., Pilla, M., and Bois, A. D. (2012). Impacto do uso de phase-change memory em memorias transacionais em software. In *WSCAD-WIC 2012*.