Apresentação do Artigo

Abordagem de Memória Transacional e Identificação de Local-Threads e Read-Only Memory Data

Alexandre de Queiroz e Wilson Vendramel

RA 098321/075642

IC – UNICAMP

Prof. Doutor Paulo Cesar Centoducatte

M0401 – Arquitetura de Computadores I

Agenda

- Introdução
- Paralelismo
- Memória Transacional
- Avaliação de Memória Transacional
- Memória Transacional Local
- Conclusão

Introdução (I)

- Até o final dos anos 90 o desempenho alcançado pelos processadores crescia exponencialmente;
- Lei de Moore (*Gordon Earl Moore*);
- Esse panorama foi alterado no início desse século, com o aumento da energia dissipada.

Introdução (II)

- Embora seja verdade que, segundo a lei de Moore, o número de transistores encapsulados continua crescendo, o desempenho do mesmo não acompanhou o esse ritmo;

- Barreira térmica de 3.8GHz.

Introdução (III)

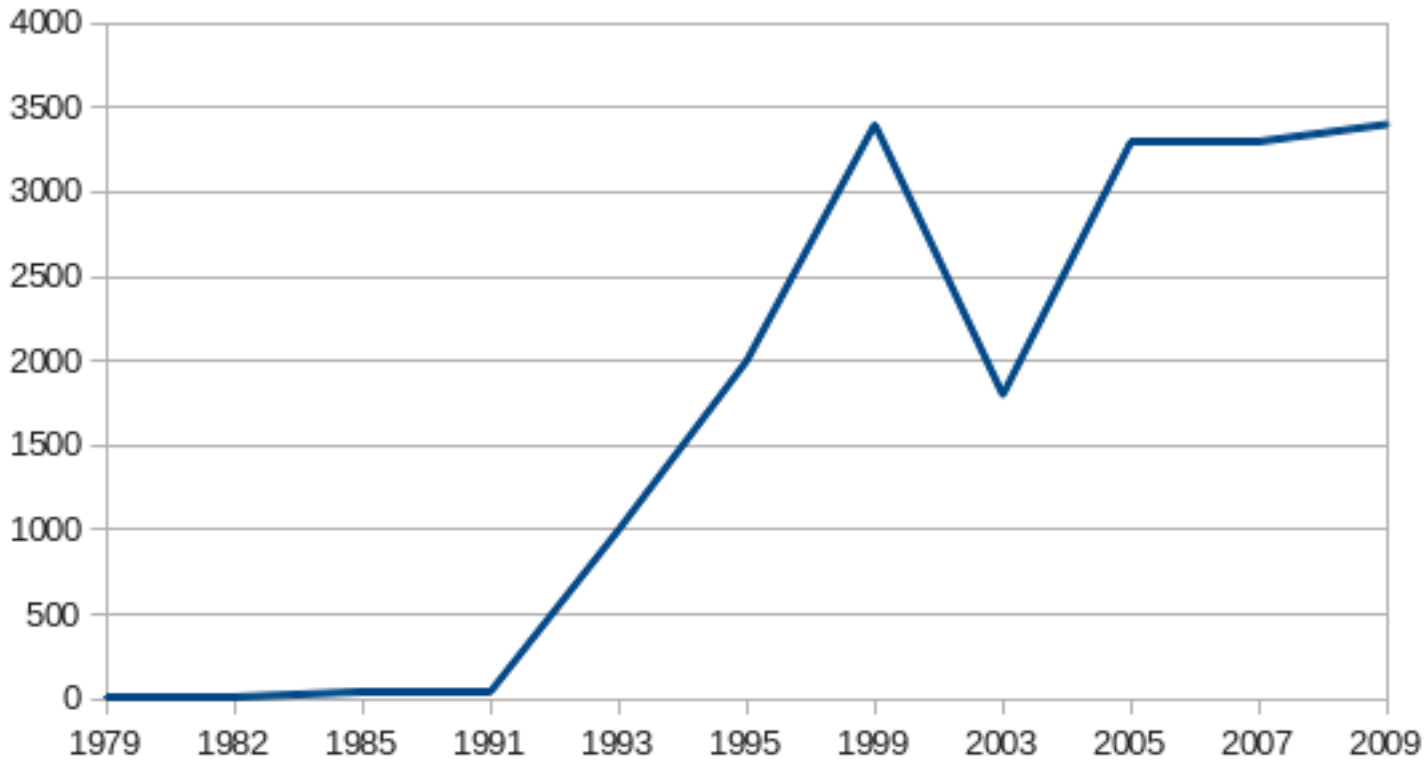

Figura 1: Velocidade dos processadores Intel [21][22].

Paralelismo (I)

- O termo paralelismo em computação é utilizado para indicar que mais de uma instrução será executada ao mesmo tempo por uma CPU.
  - Processos
  - *Threads*

Paralelismo (II)

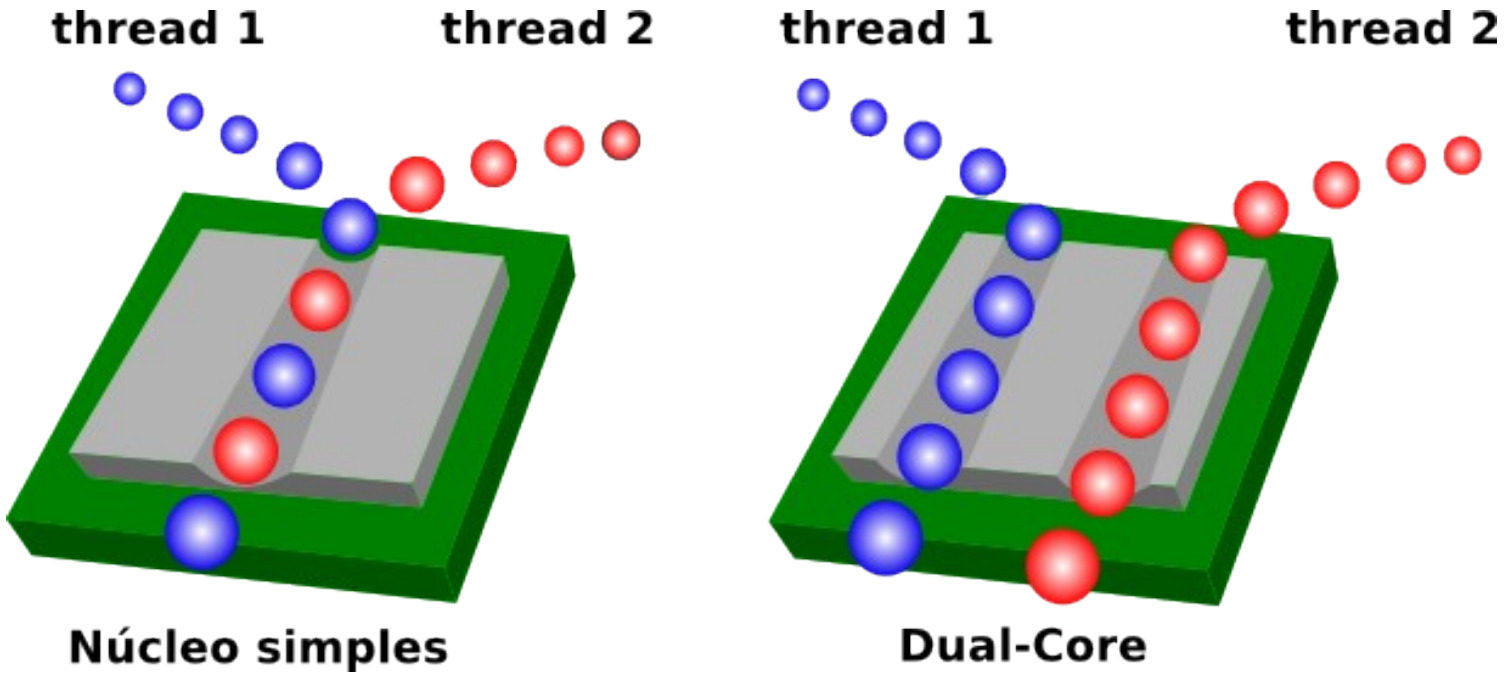

Figura 2: Processador Single-core e um Dual-core [21].

Memória Transacional (I)

- O que é uma transação?

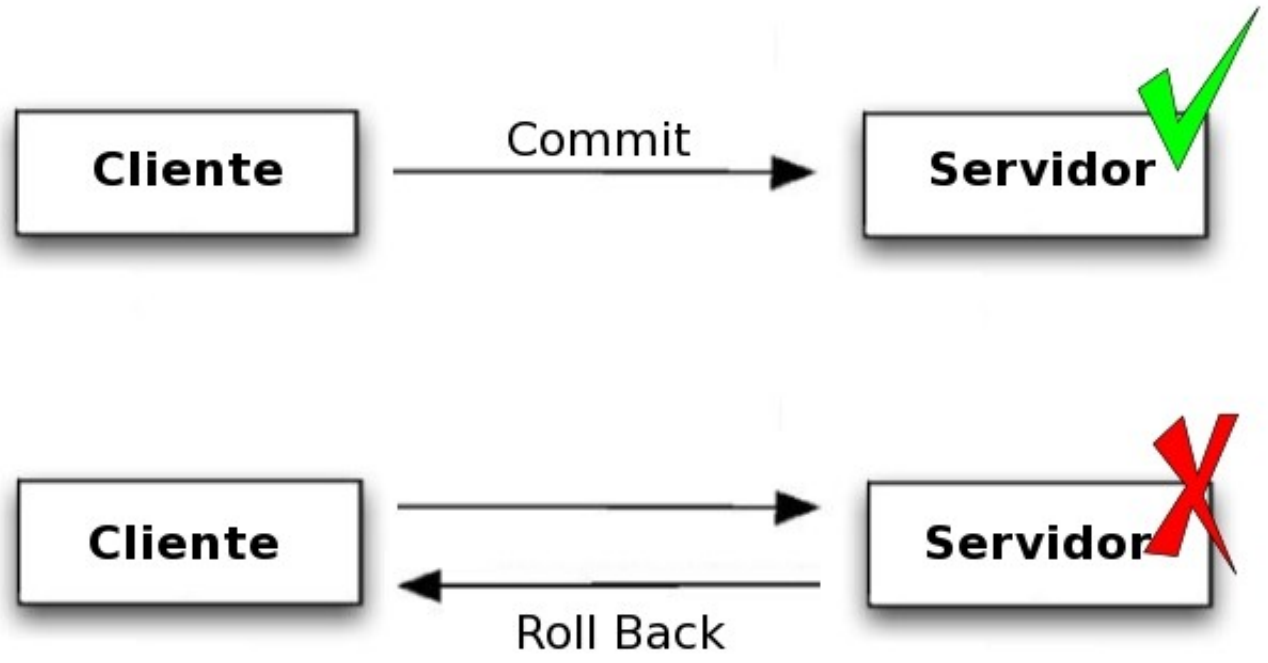

Figura 3: Operação de tudo ou nada em Banco de Dados [7][9].

Memória Transacional (II)

- Uma transação deve prover:
  - ➔ *Atomicidade*
  - ➔ *Facilidade de Codificação*
  - ➔ *Escalabilidade*
  - ➔ *Composição*

Memória Transacional (III)

- Implementações de Memória Transacional
  - Bloqueante e não bloqueante
  - STM *(Software Transaction Memory)*
  - HTM *(Hardware Transaction Memory)*
  - *HybridTM*

Memória Transacional (IV)

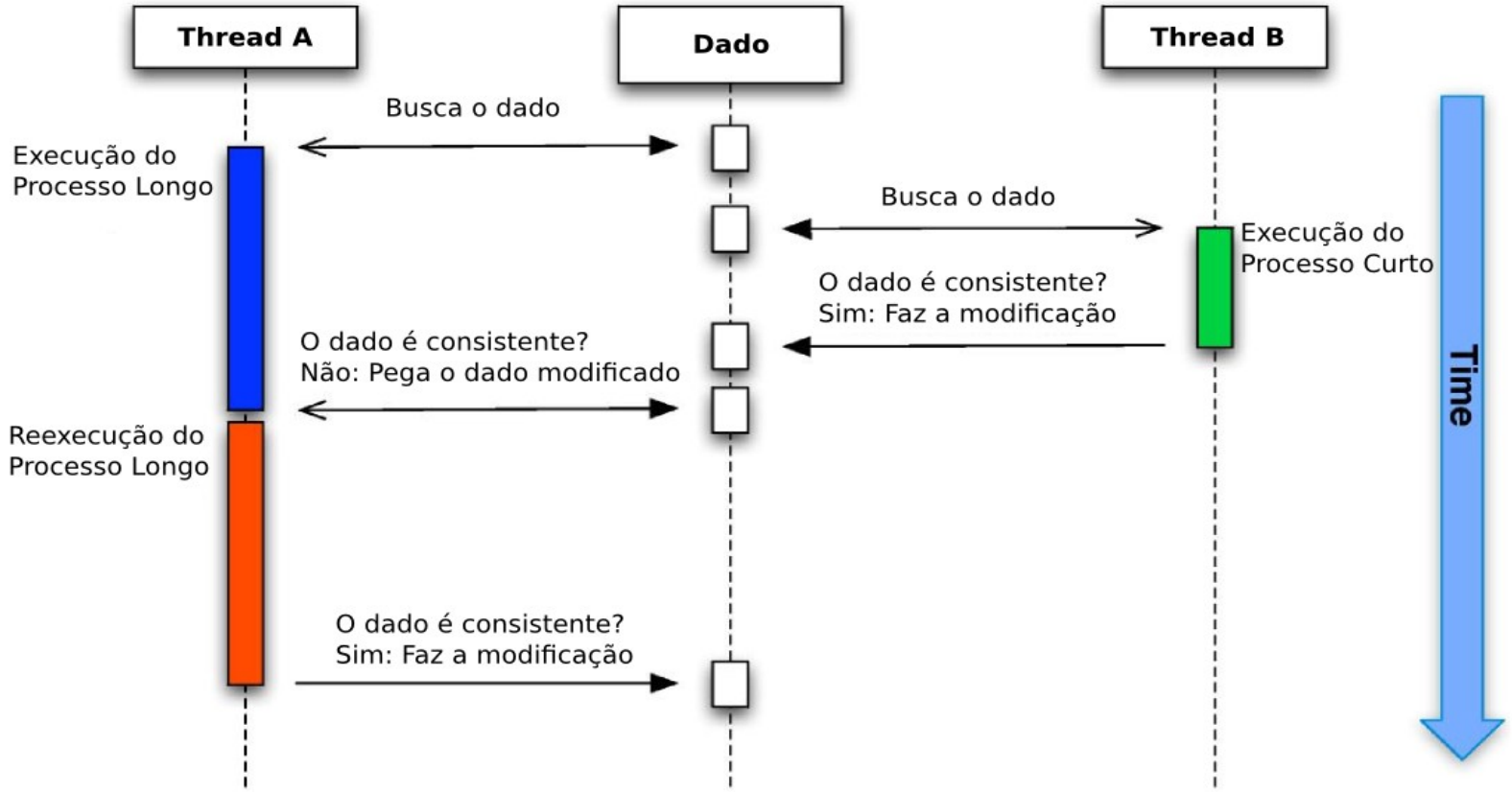

Figura 4: Threads em memória transacional [17].

Memória Transacional (V)

- Versionamento
  - ➔ Versionamento direto
  - ➔ Versionamento diferido

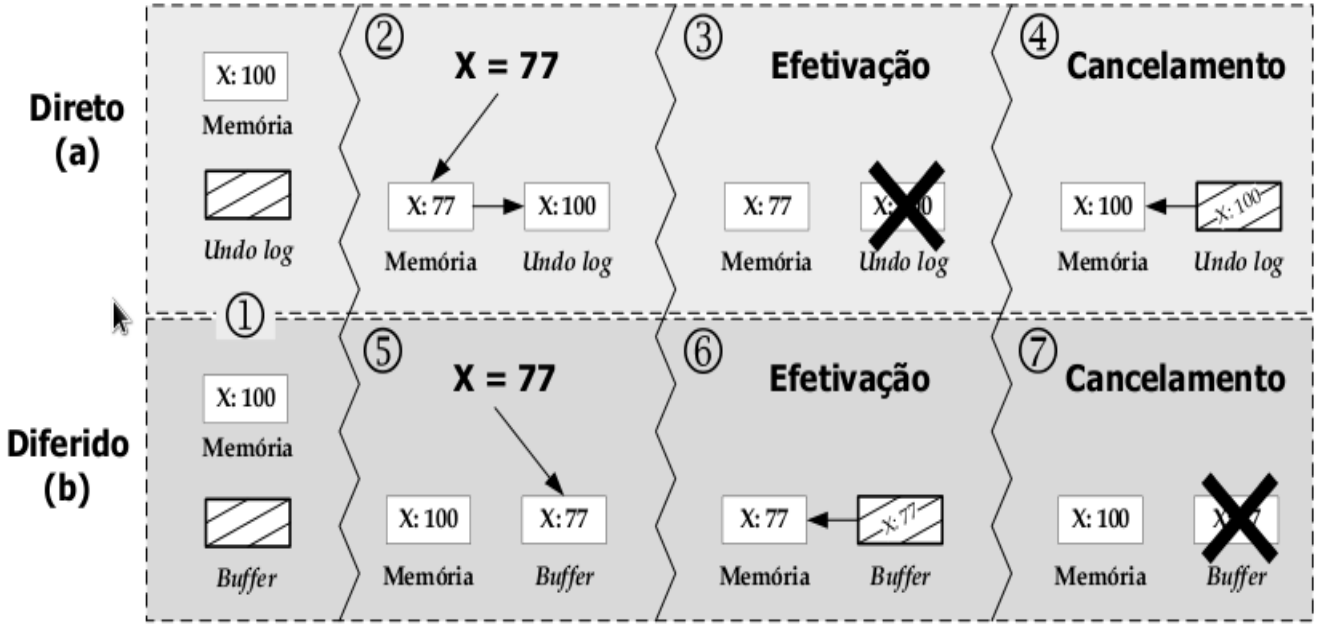

Figura 5: Versionamento direto e diferido [3]

Memória Transacional (VI)

- Detecção de Conflitos
  - pessimista
  - otimista

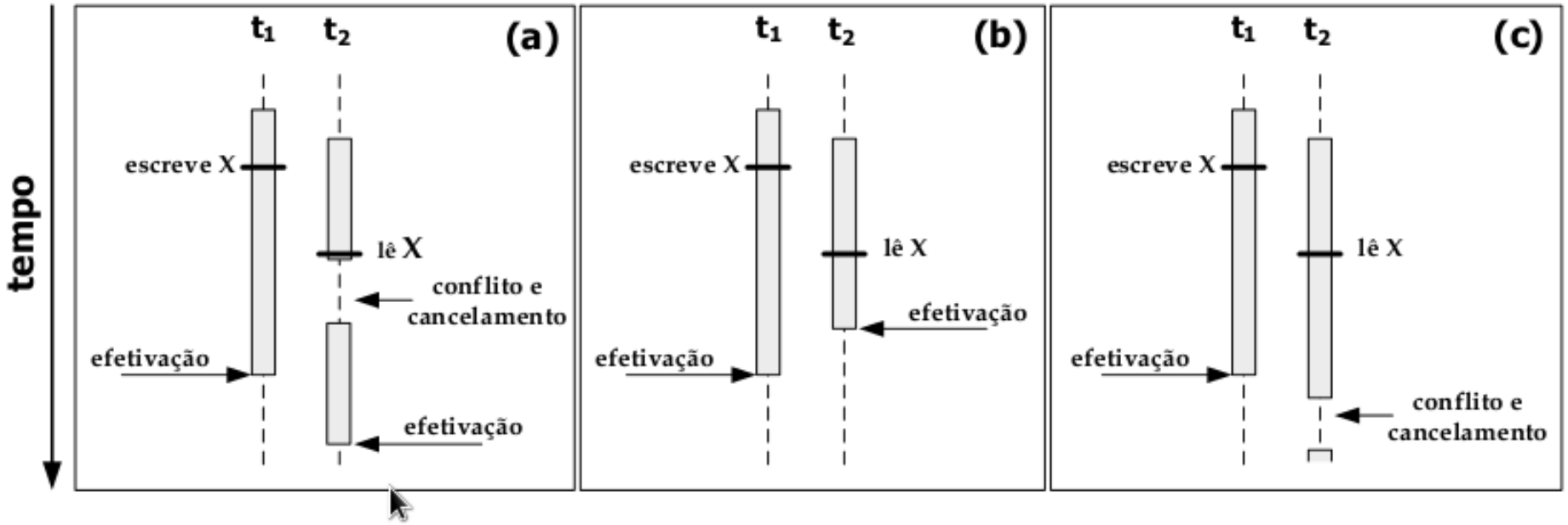

Figura 6: Cenáros para deteção de conflitos otimista e pessimista [3].

Avaliação de TM (I)

- *Benchmarks*
  - TM micro-benchmarks [8]
  - SPLASH-2 [26]
  - STM- Bench7 [10]
  - LeeTM [2]
  - STM Haskell [23]
  - STAMP [4]
  - WormBench [27]

Avaliação de TM (II)

- STAMP 2008
  - *Genome*
  - *Intruder*
  - *Kmeans High*
  - *Kmeans Low*
  - *Labyrinth*
  - *Ssca2*
  - *Vacation High*
  - *Vacation Low*
  - *Yada*

Memória Transacional Local (I)

- *Local-Thread Data*
- *Read-Only Memory Data*
- Análise e captura de LTD e ROMD
  - ➔ Intervenção do programador;
  - ➔ API
    - *addPrivateMemoryBlock*
    - *removePrivateMemoryBlock*

Memória Transacional Local (II)

- Avaliação da TM Local
  - *Benchmark* STAMP 0.9.9
    - Apenas 16 *threads*
  - Intel Dunnington
    - 4 processadores (6 núcleos cada)
    - 16 GB RAM
    - Red Hat Linux 7

Memória Transacional Local (III)

- Intervenção de software para estimar o número de barreiras;
- Algoritmo próprio para contar o número de acessos de transações locais a *heap* e *stack:*
  - ➔ Diminuição de barreiras nos programas STAMP;
  - ➔ Aumento de desempenho em quase todos os programas;
  - ➔ Número de leituras maior que escrita.

Memória Transacional Local (IV)

- Apesar do código simples, a análise do compilador é bastante eficaz para localização e eliminação de barreiras;
- *Labyrinth* não apresentou barreiras redundantes;
- *Yada* pode ser reduzido em 60% as barreiras existentes;

Memória Transacional Local (V)

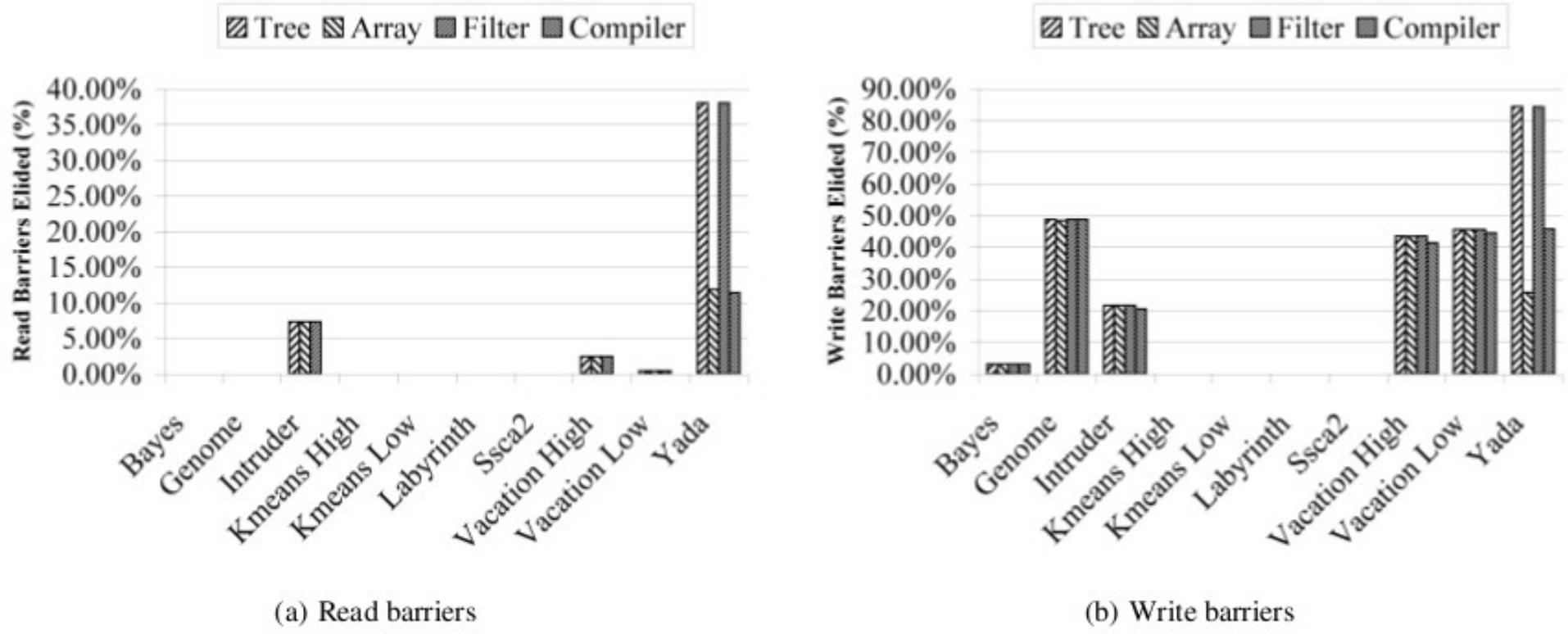

Figura 8: Barreiras removidas com diferentes técnicas [6].

Memória Transacional Local (VI)

Tabela 1: Taxa de *abort-to-commit* em 16 threads [6].

	Baseline	Tree	Array	Filtering	Compiler
Bayes	95	0.86	1.2	0.94	0.67
Genome	0.05	0	0	0	0
Intruder	0.78	0.79	0.78	0.78	0.78
Kmeans High	1.6	1.6	1.6	1.6	1.6
Kmeans Low	0.66	0.67	0.68	0.65	0.67
Labyrinth	0.18	0.17	0.17	0.17	0.17
Ssca2	0	0	0	0	0
Vacation High	0.28	0.01	0.01	0.01	0.02
Vacation Low	0.24	0	0	0.01	0.01
Yada	1.7	1.6	1.7	1.6	1.6

Memória Transacional Local (VII)

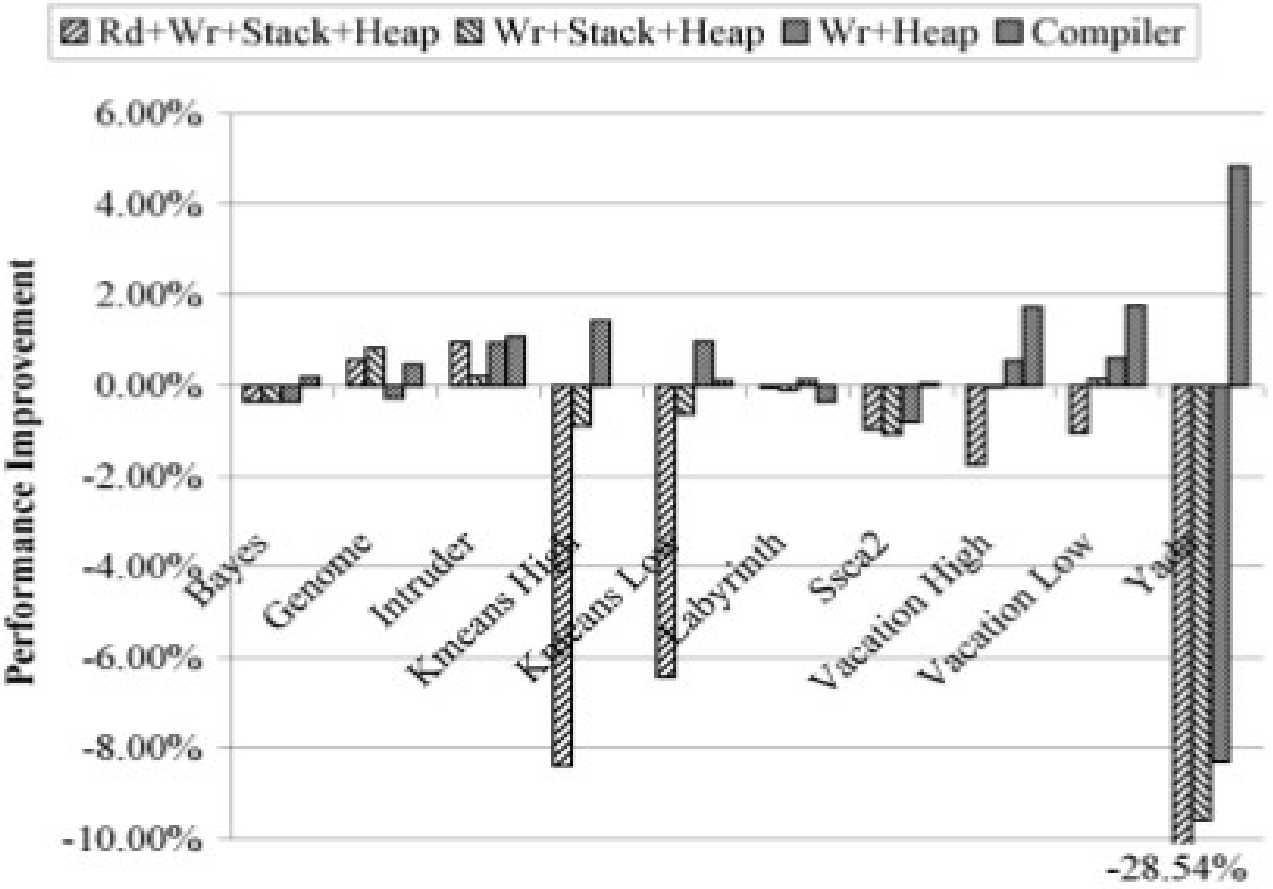

Figura 9: Otimizações em tempo de execução [6]

Conclusão (I)

- Embora ainda experimental, a utilização da Memória Transacional pode melhorar o desempenho dos aplicativos que exploram paralelismo, garantindo que os dados compartilhados da memória sejam utilizados de forma correta.

Conclusão (II)

- A proposta de TML demonstrou um grande avanço em TM que identifica os blocos de dados locais em uma transações e valores constantes, fazendo com que haja uma diminuição do gargalo, evitando possíveis barreiras (bloqueios) desnecessários para acessos à esses endereçamentos de memória.

Referências (I)

[1]Ananian, C. S., Asanovic, K., Kuszmaul, B. C., Leiserson, C. E., and Lie, S., 2005. Unbounded transactional memory. In *Proceedings of the 11th International Symposium on High-Performance Computer Architecture*, pages 316-327.

[2]Ansari, M., Kotselidis, C., Watson, I., Kirkham, C., Luj´ n, M., and Jarvis, K., 2008. Lee-tm: A non-trivial benchmark suite for transactional memory. In *ICA3PP'8: Proceedings of the 8th international conference on Algorithms and Architectures for Parallel Processing*, pages 196-207, Berlin, Heidelberg. Springer-Verlag.

[3]Baldassin, A. J. D. 2009. Explorando Memória Transacional em Software nos Contextos de Arquiteturas Assiméricas, Jogos Computacionais e Consumo de Energia, Doctoral Thesis. Cod 000768401. Universidade Estadual de Campinas . Instituto de Computação., IC/UNICAMP.

Referências (II)

[4]Cao Minh, C., Chung, J., Kozyrakis, C., and Olukotun, K., 2008. STAMP: Stanford transactional applications for multi-processing. In *IISWC'8: Proceedings of The IEEE International Symposium on Workload Characterization*.

[5]Chuang, W., Narayanasamy, S., Venkatesh, G., Sampson, J., Biesbrouck, M. V., Pokam, G., Calder, B., and Colavin, O., 2006. Unbounded page-based transactional memory. In *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 347-358.

[6]Dragojevic, A., Ni Y., Adl-Tabatabai, A., 2009. Optimizing transactions for captured memory. In *Proceedings of the twenty-first annual symposium on Parallelism in algorithms and architectures* (SPAA '09). ACM, New York, NY, USA, 214-222. DOI=10.1145/1583991.1584049 http://doi.acm.org/10.1145/1583991.1584049 .

Referências (III)

[7]Elmasri R.; Navathe, S; Sistemas de Bancos de Dados - Fundamentos e Aplicações, 3 edição, LTC, 2002

[8]Ennals, R. J., 2004. Adaptive evaluation of non-strict programs. Technical report.

[9]Eswaran K. P., Gray J. N., Lorie R. A., and Traiger I. L., 1976. The notions of consistency and predicate locks in a database system. *Commun. ACM* 19, 11 (November 1976), 624-633. DOI=10.1145/360363.360369 http://doi.acm.org/10.1145/360363.360369

[10]Goldstein, F. P. 2010., Um modelo de mem□ia transacional para arquiteturas heterogênas baseado em software Cache, Master Dissertation. Cod 000779116. Universidade Estadual de Campinas . Instituto de Computação., IC/UNICAMP.

Referências (IV)

[11]Gray, J. and Reuter, A. 1993. Transaction Processing: Concepts and Techniques. *Morgan Kaufmann Publishers*, Inc. 1993.

[12]Guerraoui, R., Kapa³ka, M., and Vitek, J., 2007. STMBench7: A benchmark for software transactional memory. In *Proceedings of the Second European Systems Conference EuroSys 2007*, pages 315-324. ACM.

[13]Hammond, L., Willey, M., and Olukotun, K. (1998). Data speculation support for a chip multiprocessor. In *Proceedings of the 8th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 58-69.

Referências (V)

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

[15]Kestor, G., Stipic, S., Unsal, O. S., Cristal, A., and Valero, M., 2009. RMS-TM: A transactional memory benchmark for recognition, mining and synthesis applications. In *TRANSACT'9: 4th Workshop on Transactional Computing*.

[16]Knight, T. 1986. An architecture for mostly functional languages. In *Proceedings of the 1986 ACM Conference on LISP and Functional Programming*, pages 105-112.

Referências (VI)

[17]Kraus, J. M., Kestler, H. A. 2010., A highly efficient multi-core algorithm for clustering extremely large datasets. In *BMC Bioinformatics* 2010, **11:**169 doi:10.1186/1471-2105-11-169 = http://www.biomedcentral.com/1471-2105/11/169

[18]Lupon, M., Magklis, G., and Gonzalez, A., 2010. A Dynamically Adaptable Hardware Transactional Memory. In *Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'43)*. IEEE Computer Society, Washington, DC, USA, 27-38. DOI=10.1109/MICRO.2010.23 http://dx.doi.org/10.1109/MICRO.2010.23

[19]Moore, G. E. 1975., Progress in digital integrated electronics. In *International Electron Devices Meeting*, pp. 11-13, 1975.

Referências (VII)

[20]Moore, K. E., Bobba, J., Moravan, M. J., Hill, M. D., and Wood, D. A., 2006. LogTM: Log-based transactional memory. In *Proceedings of the 12th International Symposium on High-Performance Computer Architecture*, pages 254-265.

[21]Morimoto, C. E., 2002. Hardware Manuel Completo. In *Guia do Hardware*, 2002. Last access 2011/06/19 - 14h30- http://www.hardware.com.br/livros/hardware-manual/evolucao-dos-processadores.html.

[22]Olukotun, K., Hammond, L. 2005., The future of microprocessors. *Queue*, 3(7):26-34, September 2005.

Referências (VIII)

[23]Perfumo, C., Sömez, N., Stipic, S., Unsal, O., Cristal, A., Harris, T., and Valero, M., 2008. The limits of software transactional memory (stm): dissecting haskell stm applications on a many-core environment. In *CF' 08: Proceedings of the 5th conference on Computing frontiers*, pages 67-78, New York, NY, USA. ACM.

[24]Rajwar, R., Herlihy, M., and Lai, K., 2005. Virtualizing transactional memory. In *Proceedings of the 32nd Annual International Symposium on Computer Architecture*, pages 494-505.

[25]Shavit, N. and Touitou, D. 1995. Software transactional memory. In *Proceedings of the 14th Annual ACM Symposium on Principles of Distributed Computing*, pages 204-213.

Referências (IX)

[26]Woo, S. C., Ohara, M., Torrie, E., Singh, J. P., and Gupta, A., 1995a. The splash-2 programs: Characterization and methodological considerations. In *INTERNATIONAL SYMPOSIUM ON COMPUTER ARCHITECTURE*, pages 24-36. ACM.

[27]Zyulkyarov, F., Cristal, A., Cvijic, S., Ayguadé, E., Valero, M., Unsal, O. S., and Harris, T., 2008. WormBench: a configurable workload for evaluating transactional memory systems. In *MEDEA '08: Proc. 9th workshop on MEmory performance*, pages 61-68.