

Michael Alexandre Costa

Memórias Transacionais

Trabalho Individual apresentado ao Programa de Pós-Graduação em Computação da Universidade Federal de Pelotas, como requisito parcial à obtenção do título de Mestre em Ciência da Computação

Orientador: Prof. Dr. André Rauber Du Bois
Coorientador: Prof. Dr. Mauricio Lima Pilla

Pelotas, 2018

RESUMO

COSTA, Michael Alexandre. **Memórias Transacionais**. 2018. 16 f. Trabalho Individual (Mestrado em Ciência da Computação) – Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2018.

...

Palavras-Chave: memória transacional; numa; uma; escalonamento

ABSTRACT

COSTA, Michael Alexandre. **Transaccional Memory**. 2018. 16 f. Trabalho Individual (Mestrado em Ciência da Computação) – Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2018.

...

Keywords: transaccional memory; numa; uma; scheduler

LISTA DE FIGURAS

Figura 1	Exemplo de versionamento adiantado (a) e atrasado (b). Fonte: (BALDASSIN, 2009)	9
----------	--	---

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SMP	Symmetric Multi-Processor
NUMA	Non-Uniform Memory Access
SIMD	Single Instruction Multiple Data
SPMD	Single Program Multiple Data
ABNT	Associação Brasileira de Normas Técnicas

SUMÁRIO

1	INTRODUÇÃO	7
1.1	Uma subseção	7
2	MEMÓRIA TRANSACIONAL	8
2.1	Propriedades	8
2.2	Versionamento de Dados	9
3	ESCALONAMENTO DE TRANSAÇÕES	10
4	ESCALONADORES NUMA	11
5	ESCALONAMENTO DE TRANSAÇÕES APLICADO À NUMA	12
6	DISCUSSÕES	13
7	CONCLUSÃO	14
	REFERÊNCIAS	15
	ANEXO A UM ANEXO	16

1 INTRODUÇÃO

...

1.1 Uma subseção

...

2 MEMÓRIA TRANSACIONAL

Memória Transacional, ou *Transactional Memory* (TM), é uma classe de mecanismos de sincronização que fornece uma execução atômica e isolada de alterações em um conjunto de dados compartilhados. Estas estão sendo desenvolvidas para que no futuro tornem-se o principal meio de fazer a sincronização em um programa concorrente, substituindo a sincronização baseada em *locks* (MORESHET; BAHAR; HERLIHY, 2006). As TMs podem ser implementadas em *software* (STM), em *hardware* (HTM) ou ainda em uma versão híbrida de *hardware* e *software*.

Na programação utilizando STMs, todo o acesso à memória compartilhada é realizado dentro de transações e todas as transações são executadas atomicamente em relação a transações concorrentes.

A principal vantagem na programação usando STM é que o programador apenas delimita as seções críticas e não é necessário preocupar-se com a aquisição e liberação de *locks*. Os *locks*, quando utilizados de forma incorreta, podem levar a problemas como *deadlocks* (BANDEIRA, 2010).

2.1 Propriedades

Transação é uma sequência finita de escritas e leituras na memória executada por uma *thread* (HERLIHY; ELIOT; MOSS, 1993), e deve satisfazer três propriedades:

- **Atomicidade:** cada transação faz uma sequência de mudanças provisórias na memória compartilhada. Quando a transação é concluída, pode ocorrer um *commit*, tornando suas mudanças visíveis a outras *threads* instantaneamente, ou pode ocorrer um *abort*, fazendo com que suas alterações sejam descartadas;
- **Consistência:** as transações devem garantir que um sistema consistente deve ser mantido consistente. Esta propriedade está relacionada com o conceito de invariância;
- **Isolamento:** as transações não interferem nas execuções de outras transações, assim parecendo que elas são executadas serialmente. Uma transação não

observa o estado intermediário de outra.

2.2 Versionamento de Dados

O versionamento de dados faz é responsável pelo gerenciamento das versões dos dados. Ele armazena tanto o valor do dado no início de uma transação como também o valor do dado modificado durante a transação, isso para garantir a propriedade de atomicidade (BALDASSIN, 2009).

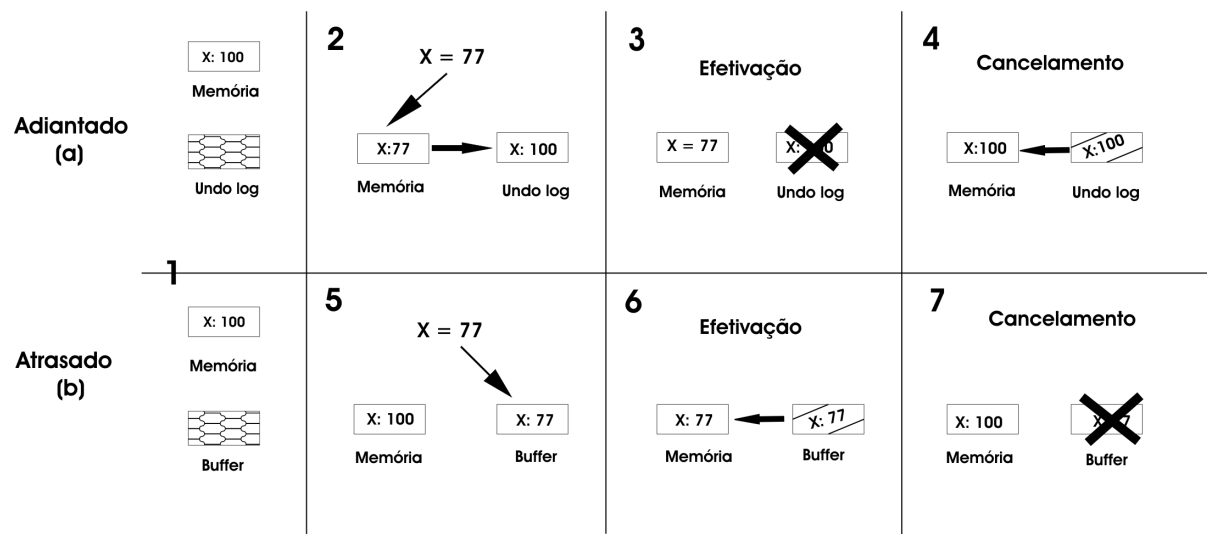


Figura 1 – Exemplo de versionamento adiantado (a) e atrasado (b). Fonte: (BALDASSIN, 2009)

Existem dois tipos de versionamento de dados:

- **Versionamento Adiantado:** como pode ser visto na Figura 1 (a), o valor modificado durante a transação é armazenado direto na memória e o valor inicial é armazenado em um *undo log*, para que no caso de cancelamento na transação o valor inicial seja restaurado na memória.
- **Versionamento Atrasado:** como pode ser visto na Figura 1 (b) neste versionamento o valor modificado durante a transação é armazenado em um *buffer* e o valor inicial é mantido na memória até que aconteça um *commit* na transação, onde o valor armazenado no *buffer* é escrito na memória. Caso aconteça o cancelamento na transação, o valor do *buffer* é descartado.

3 ESCALONAMENTO DE TRANSAÇÕES

...

4 ESCALONADORES NUMA

...

5 ESCALONAMENTO DE TRANSAÇÕES APLICADO À NUMA

...

6 DISCUSSÕES

...

7 CONCLUSÃO

...

REFERÊNCIAS

BALDASSIN, A. J. **Explorando Memória Transacional em Software nos Contextos de Arquiteturas Assimétricas, Jogos Computacionais e Consumo de Energia**. 2009. Dissertação de Doutorado — Universidade Estadual de Campinas.

BANDEIRA, R. de Leão. **Compilador para a linguagem CMTJava**. 2010. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) — Universidade Federal de Pelotas.

HERLIHY, M.; ELIOT, J.; MOSS, B. Transactional Memory: Architectural Support for Lock-Free Data Structures. In: PROC. OF THE 20TH ANNUAL INTL. SYMPOSIUM ON COMPUTER ARCHITECTURE, 1993. **Anais...** [S.l.: s.n.], 1993. p.289–300.

MORESHET, T.; BAHAR, R. I.; HERLIHY, M. Energy-Aware Microprocessor Synchronization: Transactional Memory vs. Locks. In: WORKSHOP ON MEMORY PERFORMANCE ISSUES, 2006. **Proceedings...** [S.l.: s.n.], 2006.

ANEXO A UM ANEXO

...