

MO644/MC900

Programação Paralela

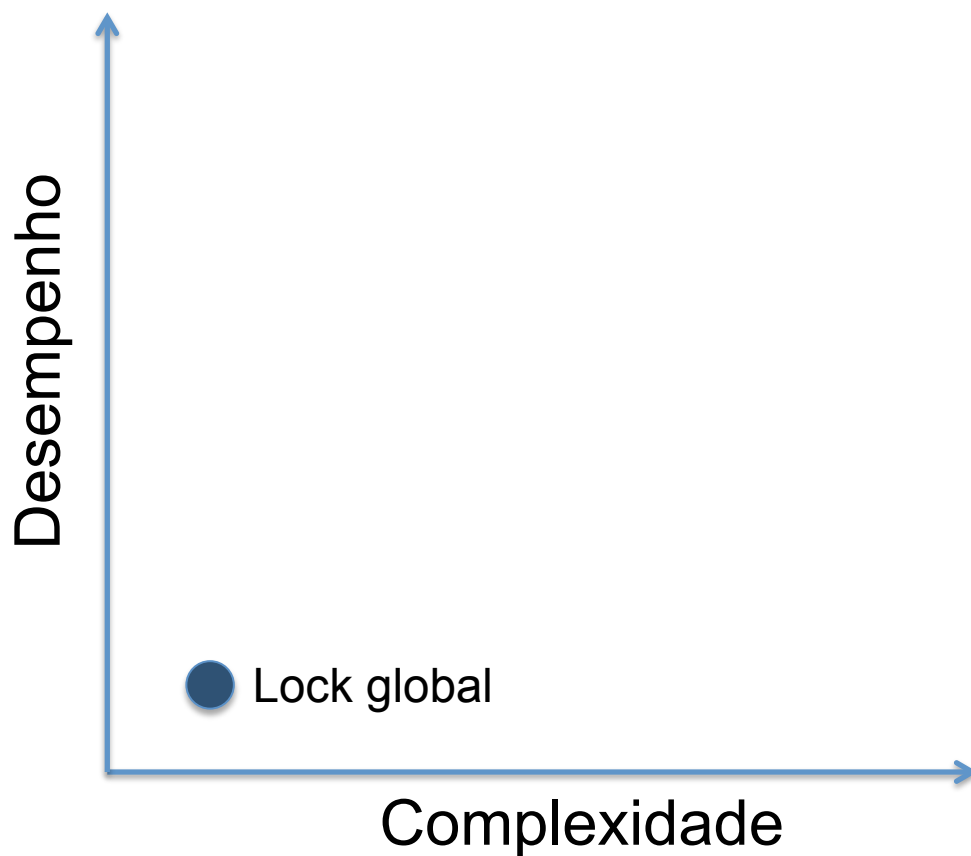
Memórias Transacionais

Guido Araujo
(contribuições de Alexandro
Baldassin e Nelson Amaral)

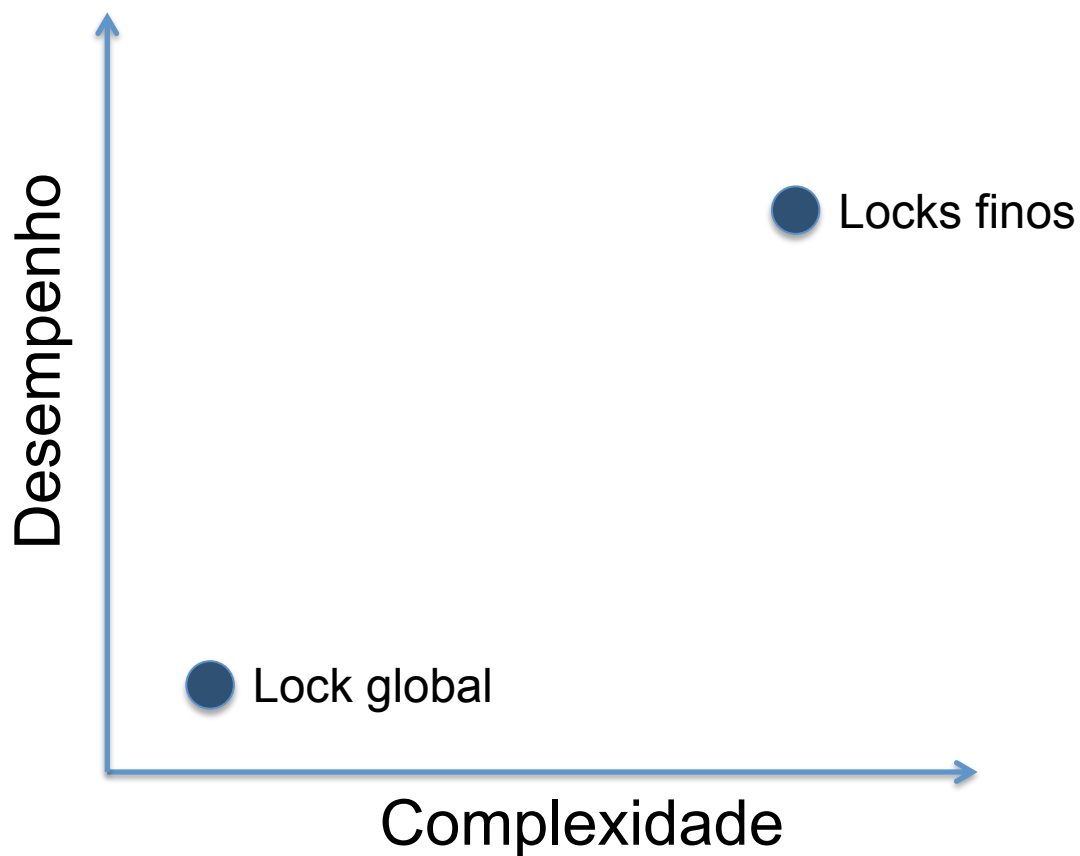
Roteiro

- Modelo de Programação
- Versionamento
- Software Transaction Memory (STM)
- Implementação TL2
- Hardware Transaction Memory (HTM)
- IBM BGQ e Intel Haswell

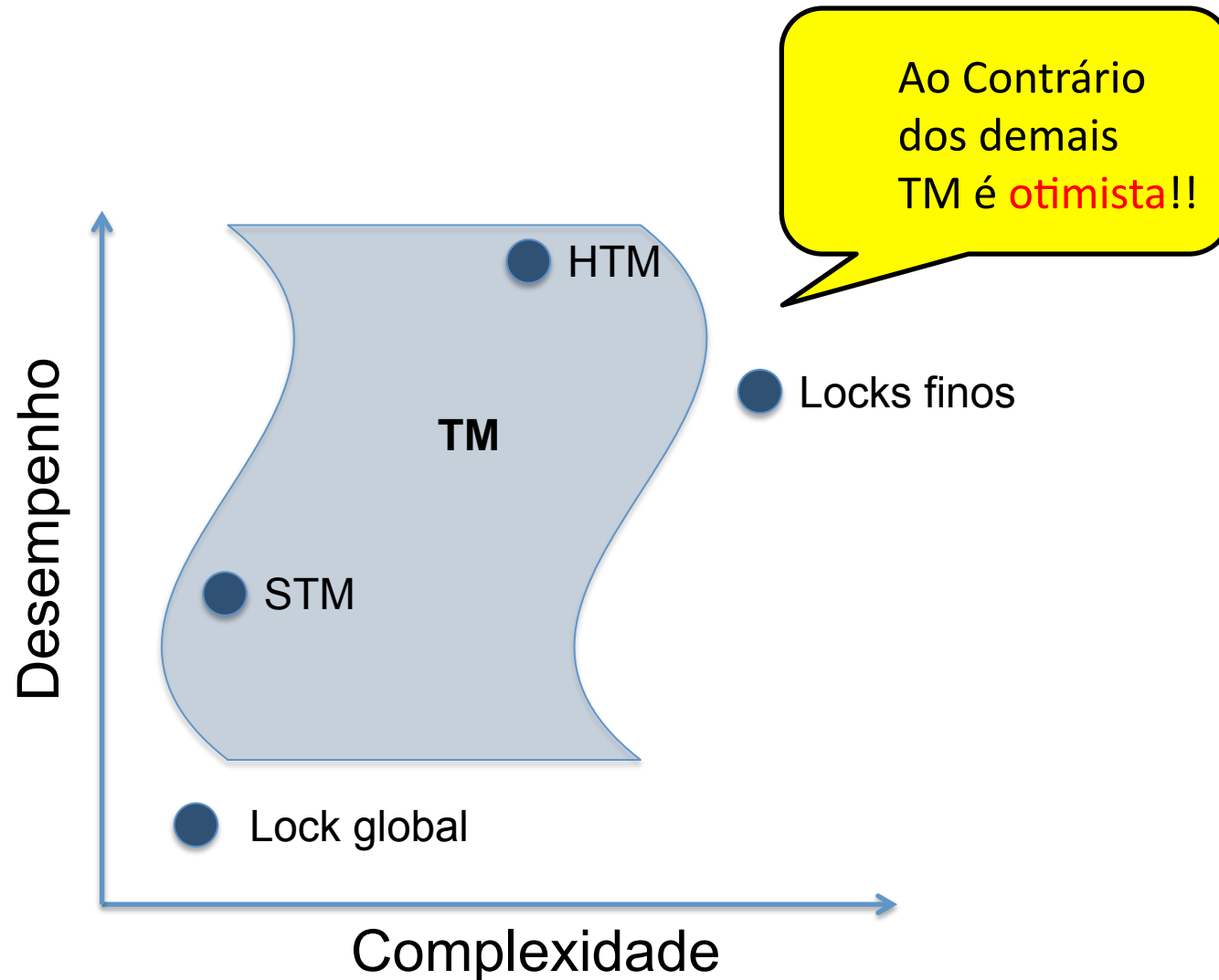
Programação concorrente



Programação concorrente



Programação concorrente



Programando com TM

- Programador delimita a região que deve ser executada atomicamente
 - Exemplo com lista ligada
- Sistema de execução (pode ser hardware ou software) cuida de garantir atomicidade, isolamento e consistência



```
public boolean add(int item) {  
    Node pred, curr;  
    boolean valid = false;  
  
    atomic {  
        pred = head;  
        curr = pred.next;  
        while (curr.key < item) {  
            pred = curr;  
            curr = curr.next;  
        }  
        if (item != curr.key) {  
            Node node = new Node(item);  
            node.next = curr;  
            pred.next = node;  
            valid = true;  
        }  
    }  
    if (valid) return true;  
    return false;  
}
```

Memória Transacional (TM)

- No modelo transacional, programadores usam o conceito de **transação** como abstração

- **A**tomicidade
- **C**onsistência
- **I**solamento

Detalhes de *como* realizar a sincronização são movidos do programador para o sistema de execução

- Vantagens

- Nível de abstração maior
- Potencial ganho de desempenho, dependente de implementação
- Composição de código

Memória Transacional (TM)

- Considere duas transações X1 e X2

X₁

i1: temp1 = load(counter);

i2: temp1 = temp1 + 1;

i3: store(counter, temp1);

X₂

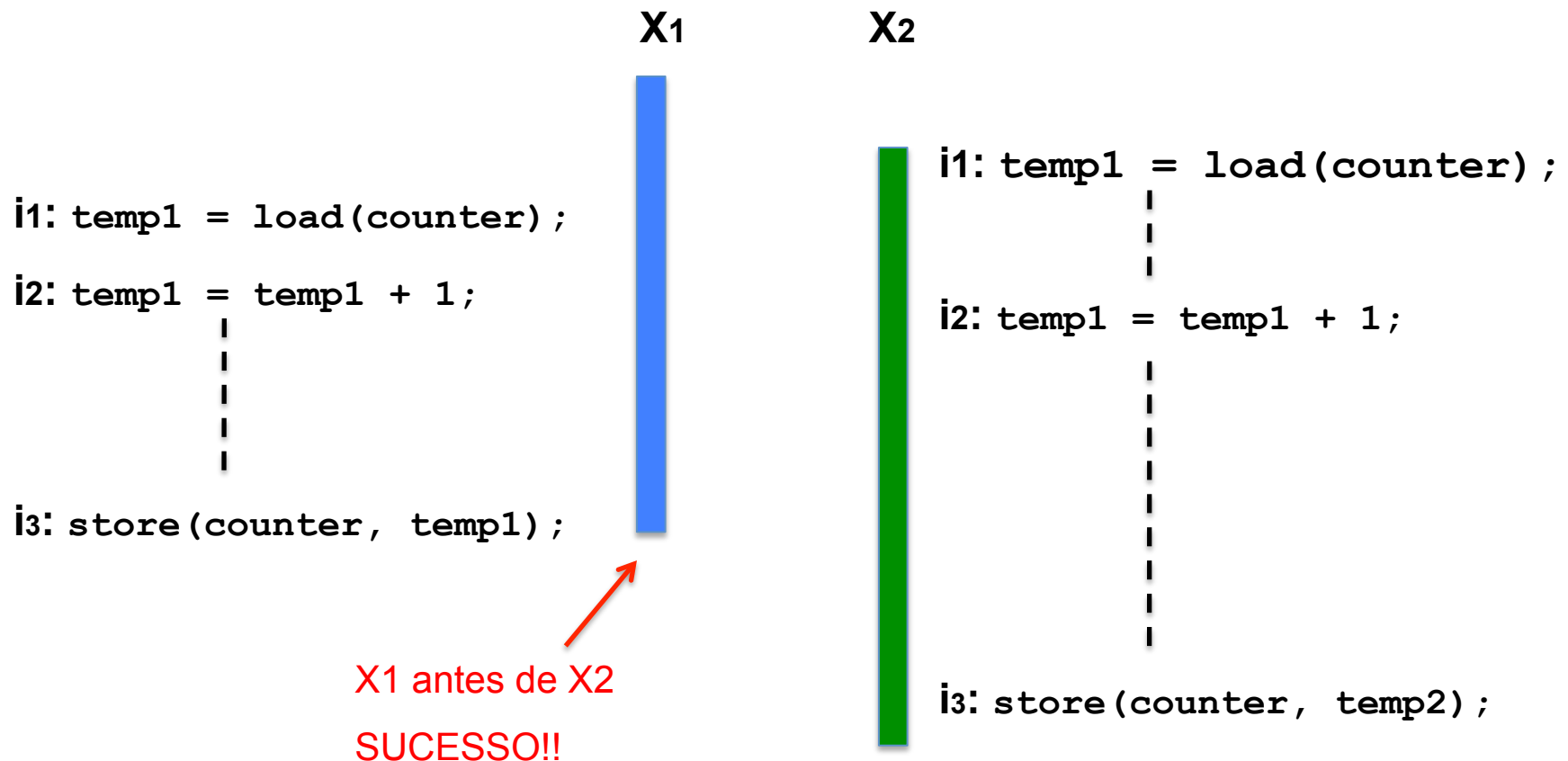
i1: temp1 = load(counter);

i2: temp1 = temp1 + 1;

i3: store(counter, temp1);

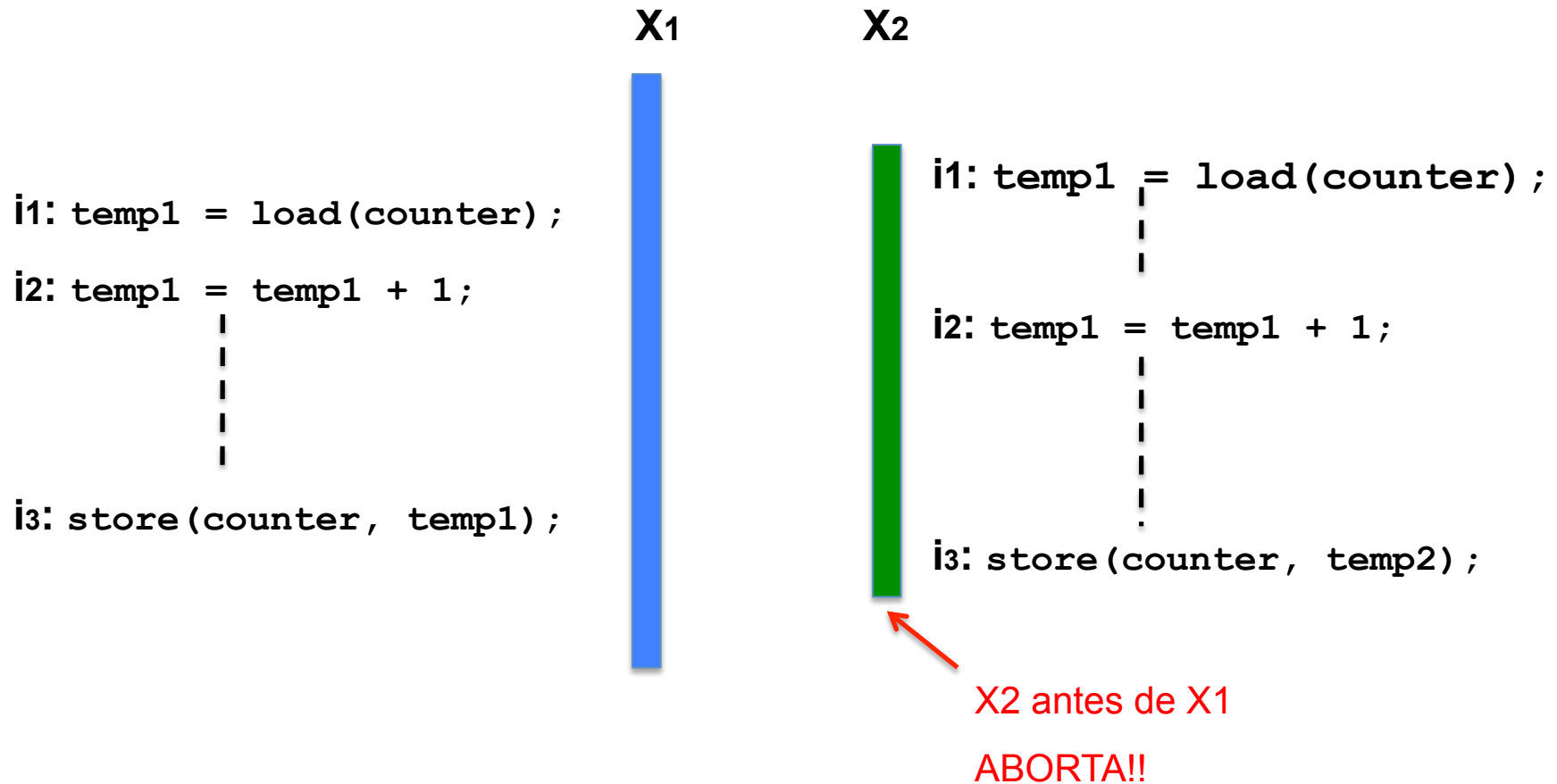
Memória Transacional (TM)

- Assuma que X1 inicia antes de X2



Memória Transacional (TM)

- Assuma que X1 inicia antes de X2



Memória Transacional (TM)

- No modelo transacional, programadores usam o conceito de **transação** como abstração
 - **A**tomicidade
 - **C**onsistência
 - **I**solamento
- Vantagens
 - Nível de abstração maior
 - Potencial ganho de desempenho, dependente de implementação (visto mais adiante)
 - Composição de código

Bom para quê?

- Estruturas de dados cuja escalabilidade é ruim com abordagens baseadas em locks
 - Exemplo: árvore rubro-negra
- Aplicações nas quais o uso de lock é muito conservativo
- Aplicações irregulares (uso extensivo de ponteiros)
 - Algoritmos de grafos
- Exemplo de sistema grande de porte
 - Servidor do jogo Quake (*Zyulkyarov et al.*)

Implementação

- O mecanismo transacional precisa fornecer...
- Versionamento de dados
 - Os dados temporários (especulativos) usados pela transação precisam ser mantidos em algum local
 - Essencial para garantir atomicidade e consistência
- Isolamento da execução
 - É necessário um mecanismo para **detectar** e **resolver** os conflitos entre transações

Versionamento de dados

- Imediato (*eager/pessimistic/direct*)
 - Memória compartilhada atualizada imediatamente (valor antigo armazenado em *buffer*)
 - Efetivação rápida, mas cancelamento lento
- Deferido (*lazy/optimistic/deferred*)
 - Armazena atualização em *buffer* interno
 - Cancelamento rápido, mas efetivação lenta

Versionamento immediato

Begin Xaction

Thread

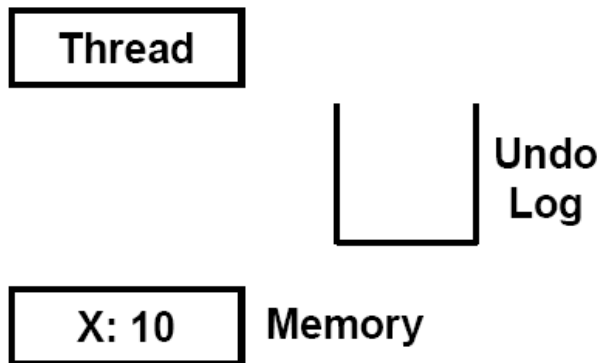
Undo
Log

X: 10

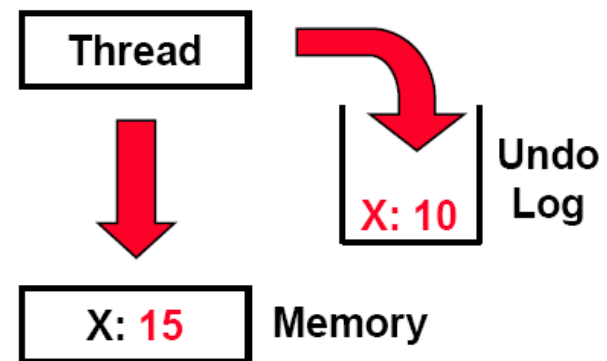
Memory

Versionamento immediato

Begin Xaction

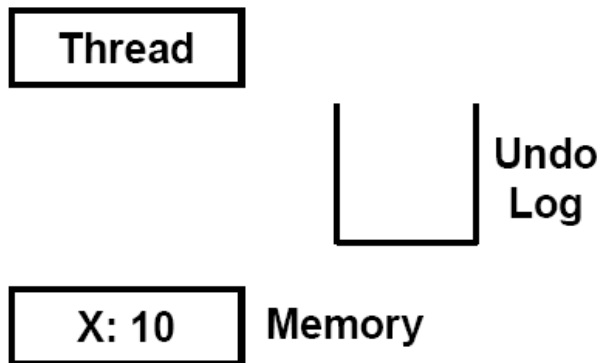


Write X ← 15

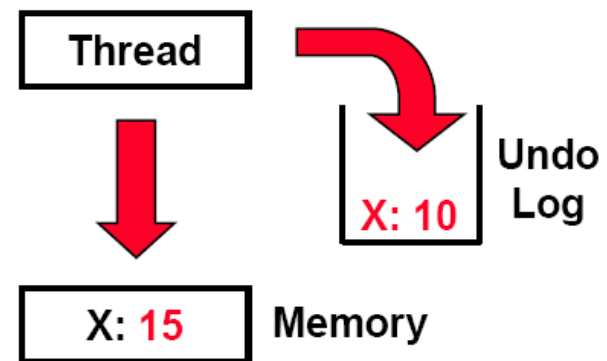


Versionamento immediato

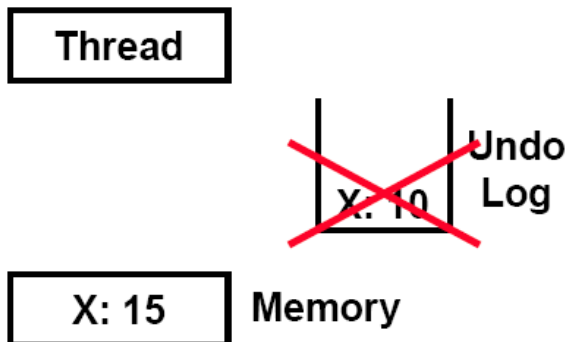
Begin Xaction



Write X ← 15

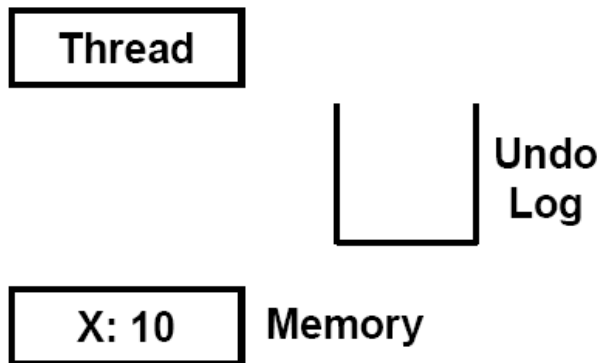


Commit Xaction

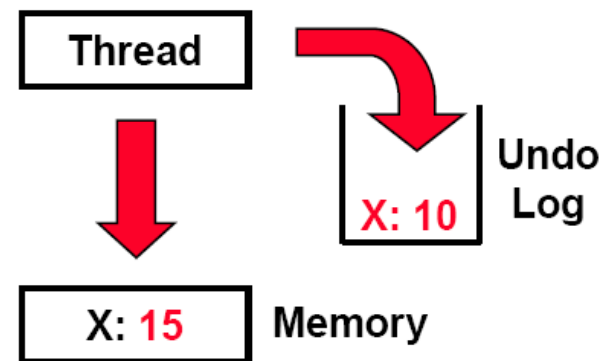


Versionamento immediato

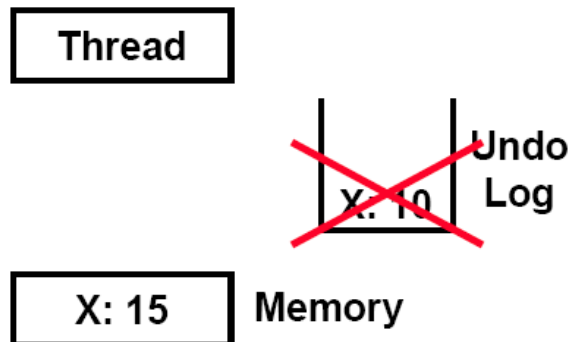
Begin Xaction



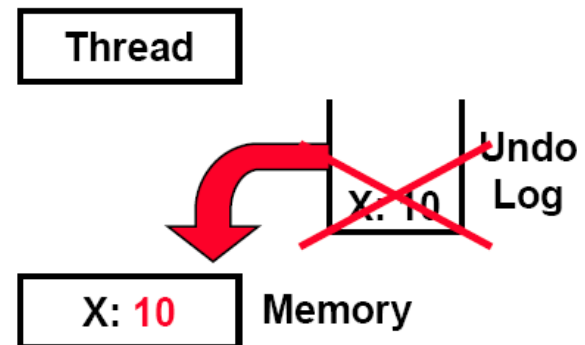
Write X ← 15



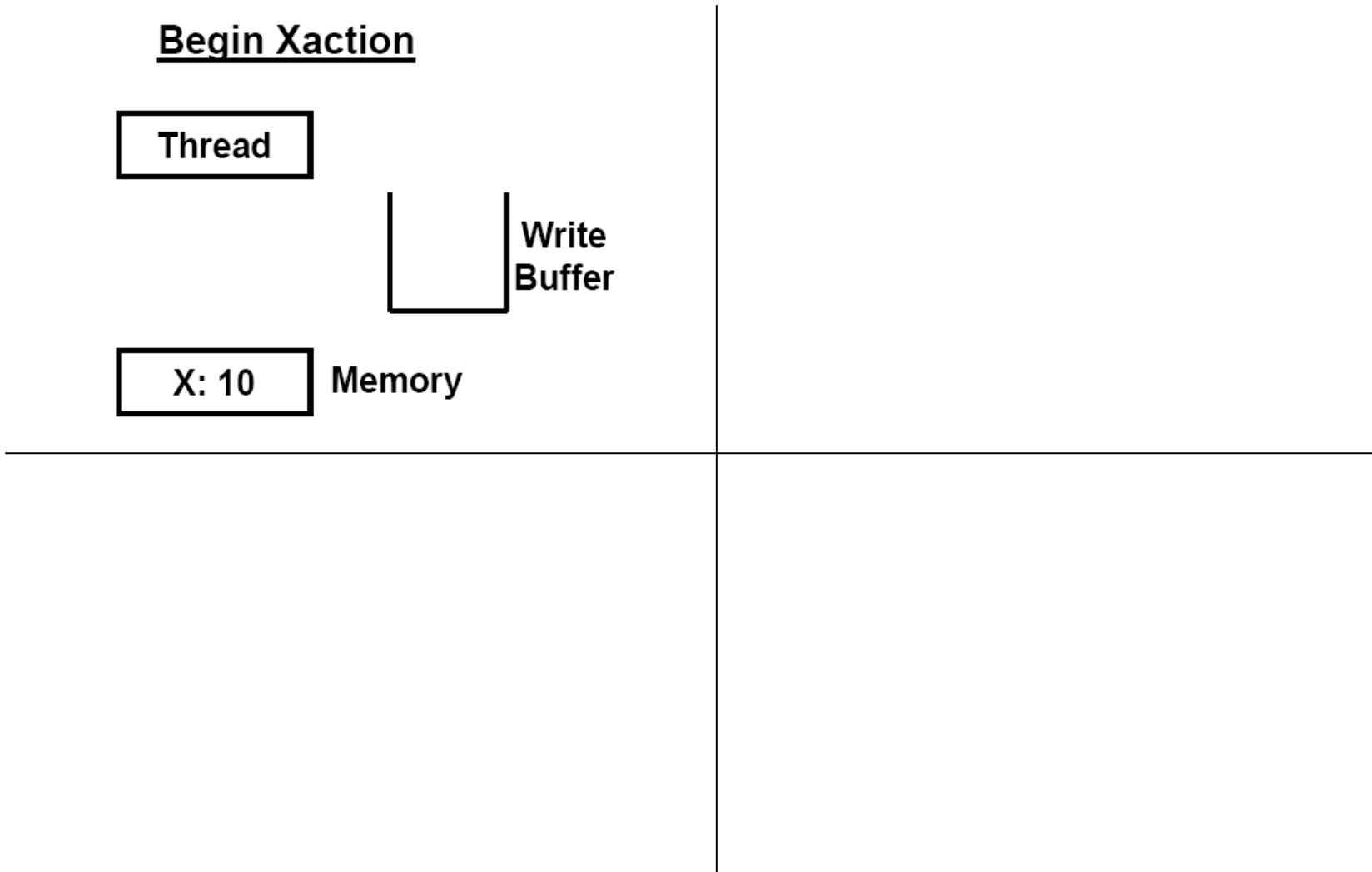
Commit Xaction



Abort Xaction

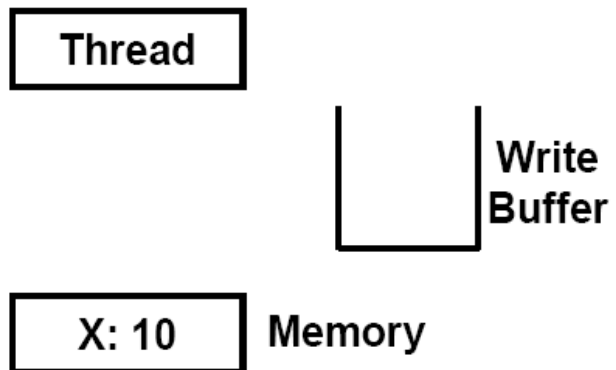


Versionamento diferido

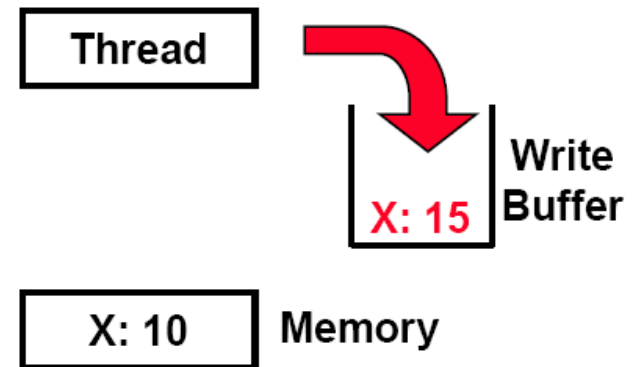


Versionamento diferido

Begin Xaction

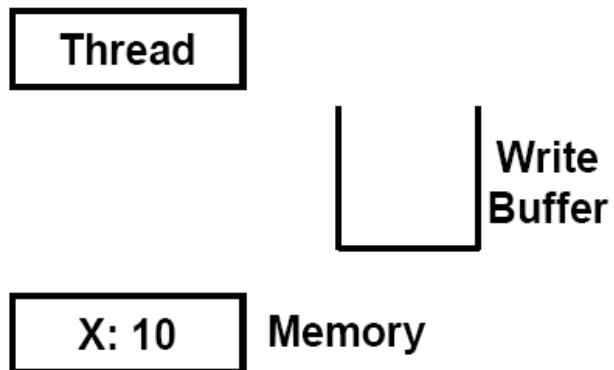


Write X ← 15

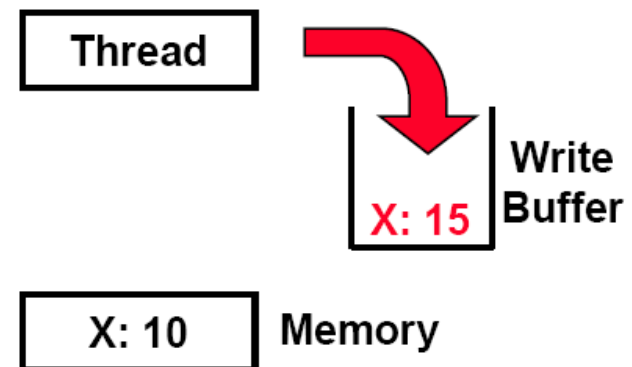


Versionamento diferido

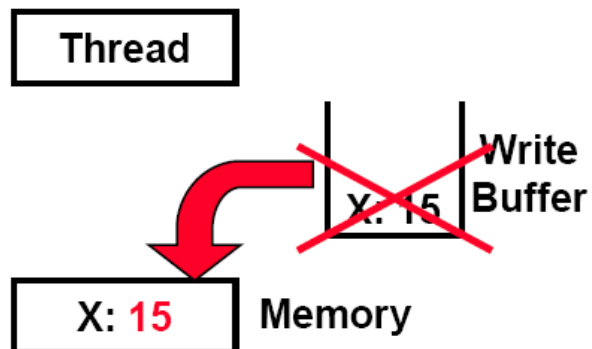
Begin Xaction



Write X ← 15

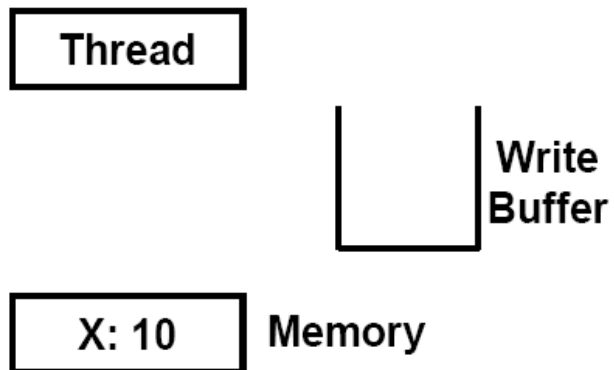


Commit Xaction

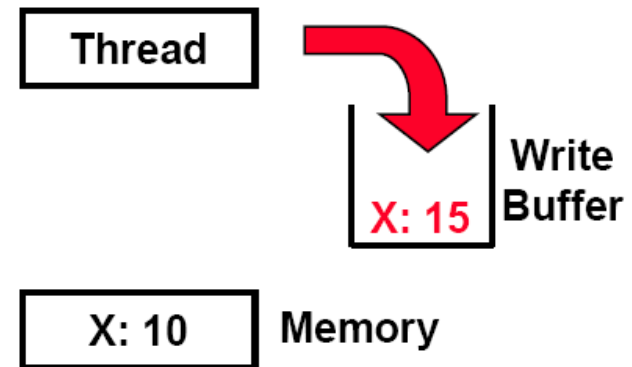


Versionamento diferido

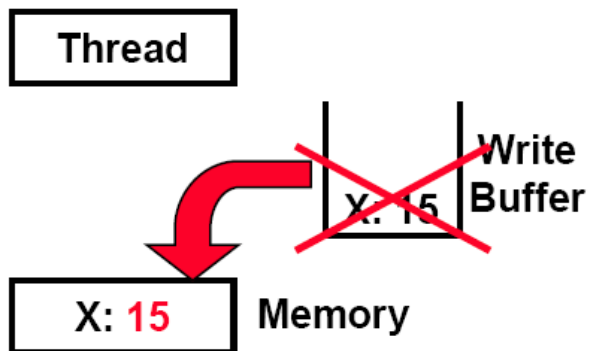
Begin Xaction



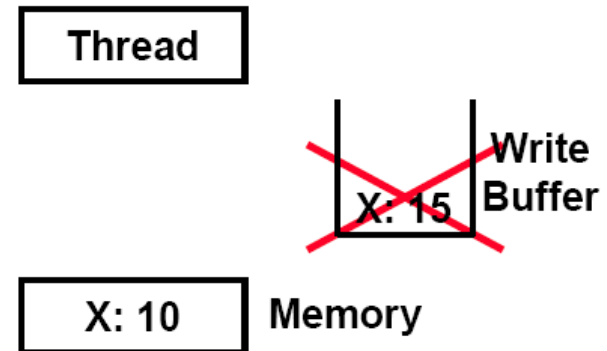
Write X ← 15



Commit Xaction



Abort Xaction



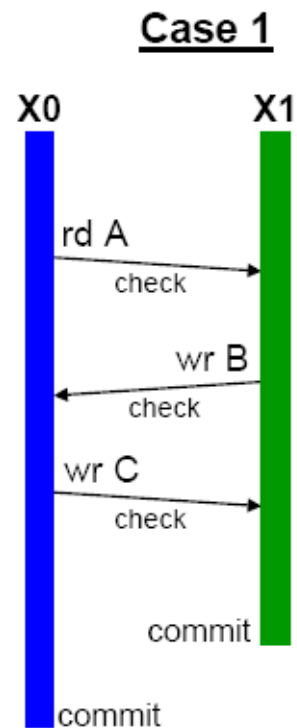
Isolamento da execução

- Detecção de conflitos
 - Usam-se dois conjuntos:
 - *Read set*: dados lidos
 - *Write set*: dados escritos
 - Ocorre conflito se há intersecção entre o conjunto de leitura e escrita de transações diferentes
- Resolução de conflitos
 - Depende do **gerenciador de contenção**
 - Exemplo: abortar imediatamente, esperar, ...
 - Importante para garantir progresso

Formas de detecção de conflitos

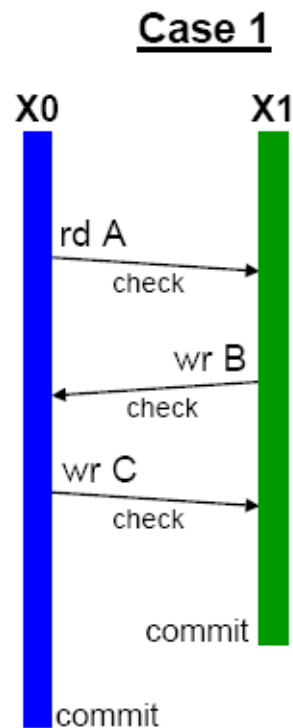
- Adiantado (*eager/pessimistic/encounter-time*)
 - Ocorre no momento dos acessos aos dados
 - Pode evitar executar código desnecessário
 - Mais suscetível a *livelock*
- Tardio (*lazy/optimistic/commit-time*)
 - Ocorre na efetivação da transação
 - Potencialmente menos conflitos
 - Menos suscetível a *livelock*, porém *starvation* pode ser um problema

Detecção de conflitos adiantado

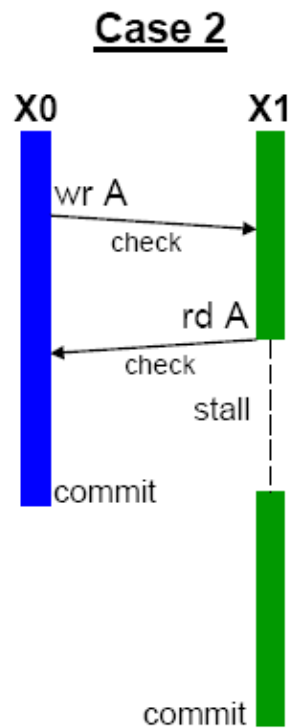


Success

Detecção de conflitos adiantado

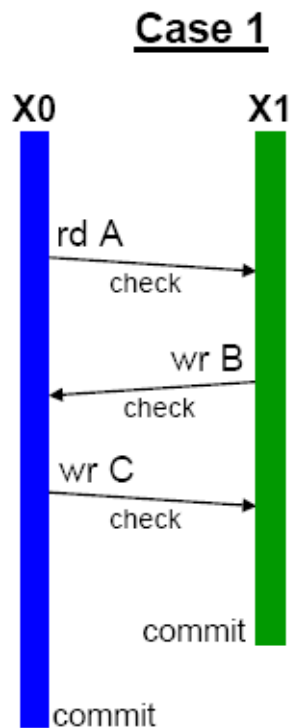


Success

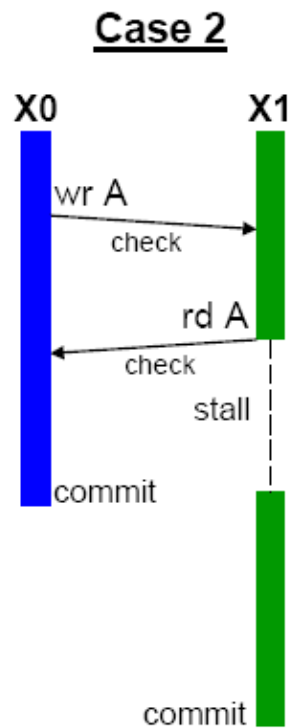


Early Detect

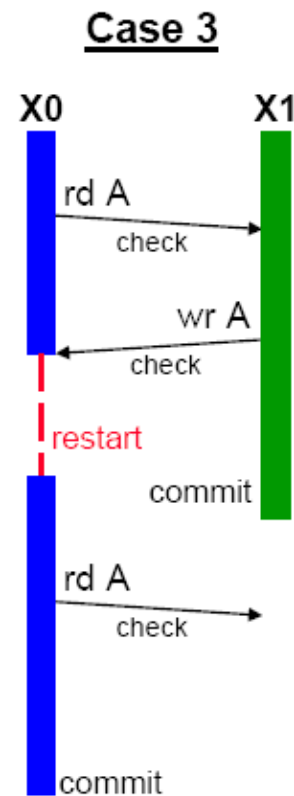
Detecção de conflitos adiantado



Success

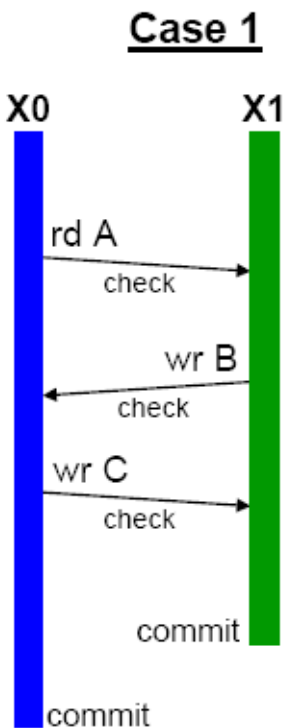


Early Detect

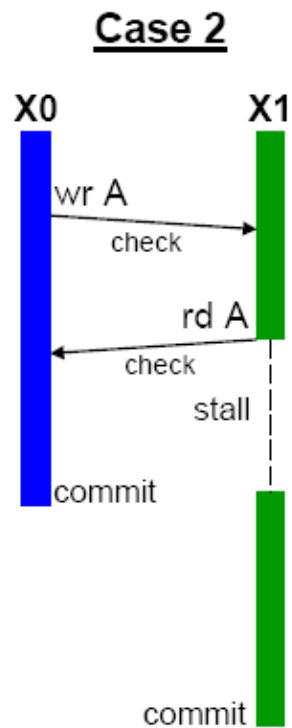


Abort

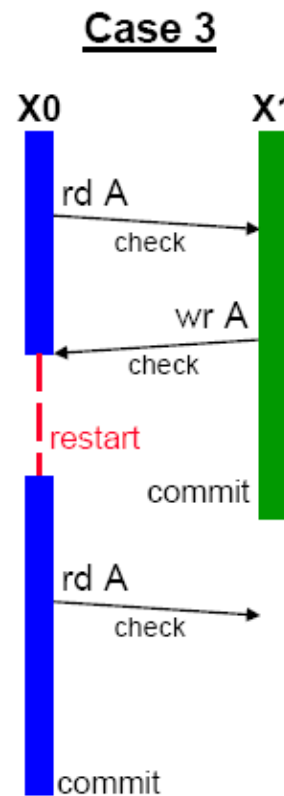
Detecção de conflitos adiantado



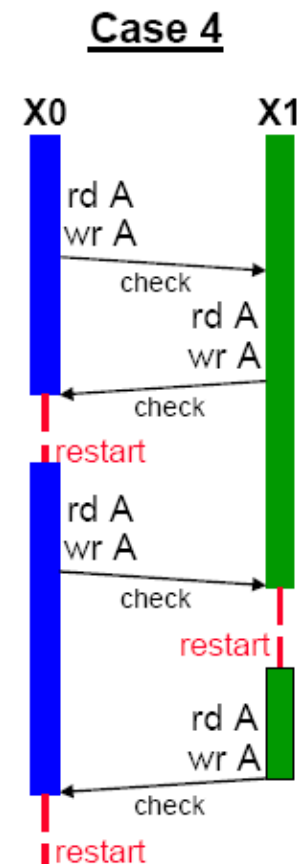
Success



Early Detect

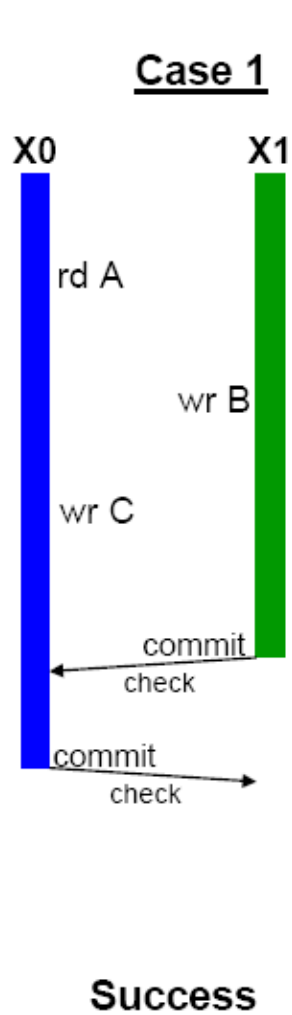


Abort

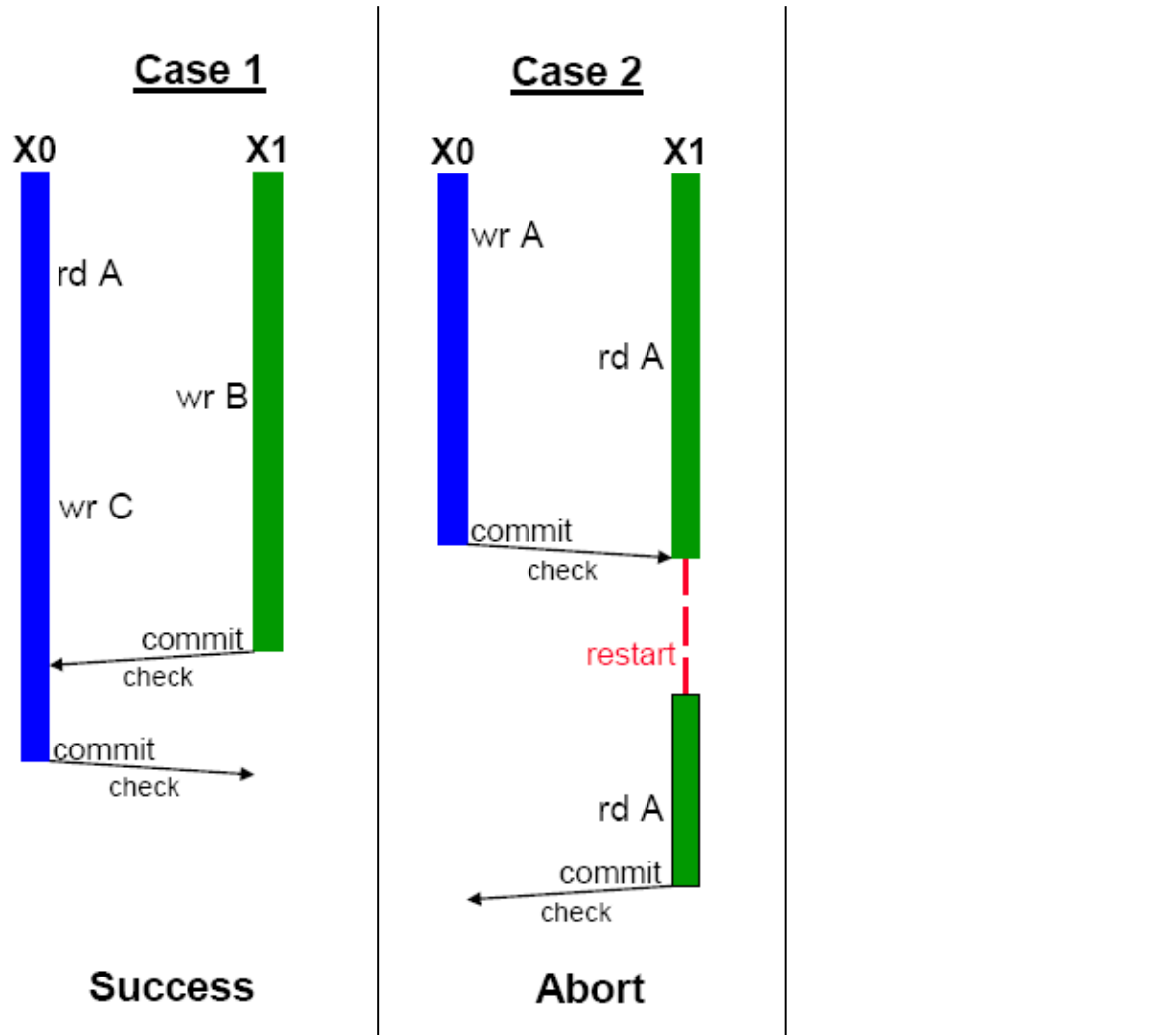


No progress

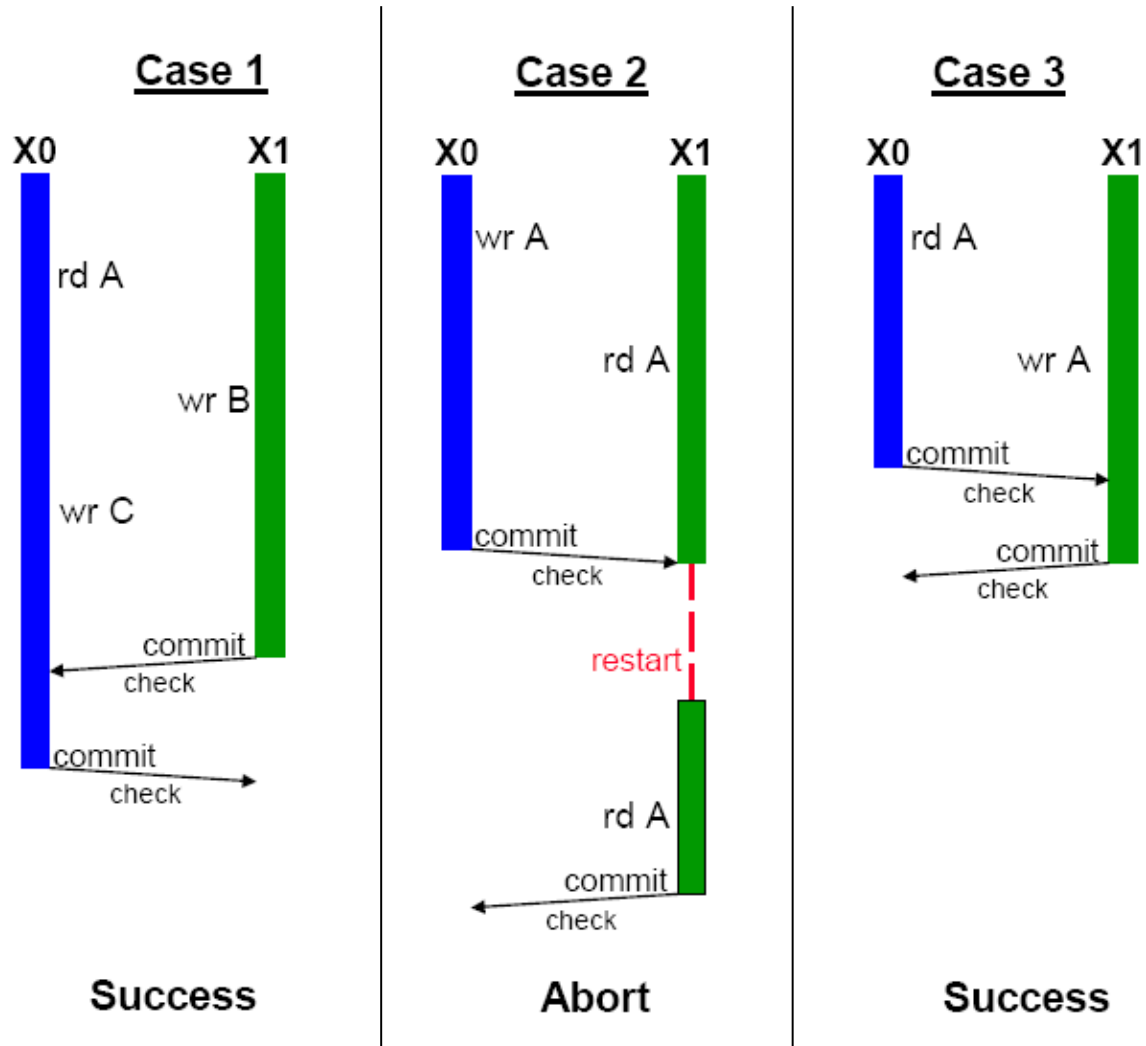
Detecção de conflitos tardio



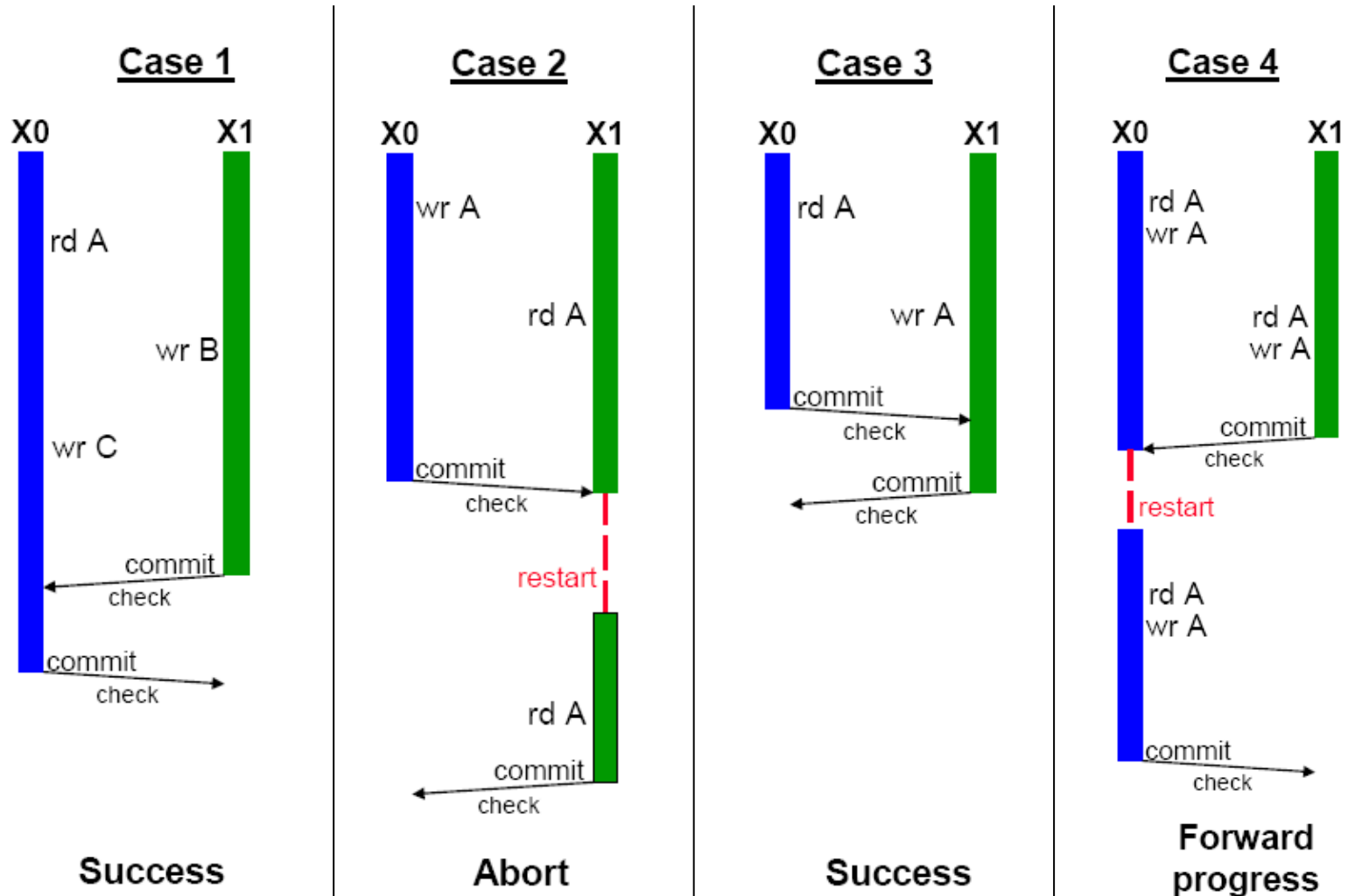
Detecção de conflitos tardio



Detecção de conflitos tardio



Detecção de conflitos tardio



Implementação de TM

- O suporte transacional pode ser realizado em hardware, software ou uma mescla de ambos (híbrido)
- Hardware (HTM)
 - Melhor desempenho
 - Problemas com virtualização (espaço, tempo)
- Software (STM)
 - Desempenho depende muito da aplicação
 - Extremamente flexível
 - Ideal para testar novas ideias

Exemplo de STM TL2

- Versionamento deferido (*lazy versioning*)
- Detecção de conflito tardio (*lazy detection*)
- User-level API
 - StartTx(), CommitTx(), AbortTx()
 - ReadTx(), WriteTx()

Exemplo de STM TL2

```
void PushLeft(DQueue *q, int val) {  
    QNode *qn = malloc(sizeof(QNode));  
    qn->val = val;  
    do {  
        StartTx();  
        QNode *leftSentinel = ReadTx(&(q->left));  
        ...  
        WriteTx(&(oldLeftNode->left), qn);  
    } while (!CommitTx());  
}
```

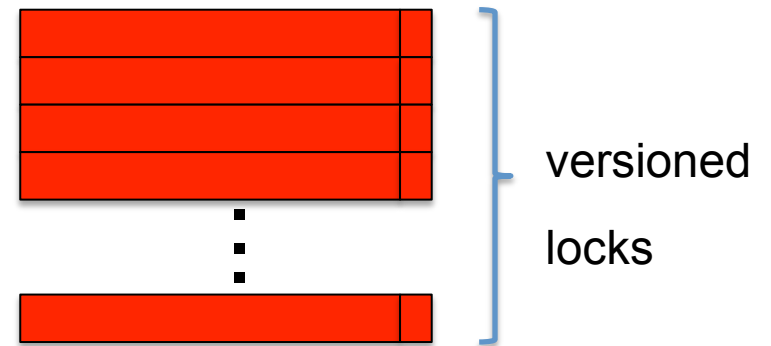
TL2 – Metadados

Compartilhado

Global Clock



Ownership Record Table (ORT)



TL2 – Metadados

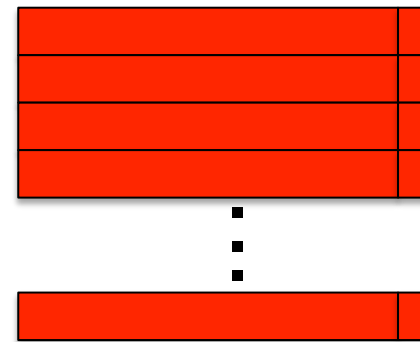
Compartilhado

Global Clock

GCLOCK

Sempre incrementado quando
uma transação é efetivada

Ownership Record Table (ORT)



versioned
locks

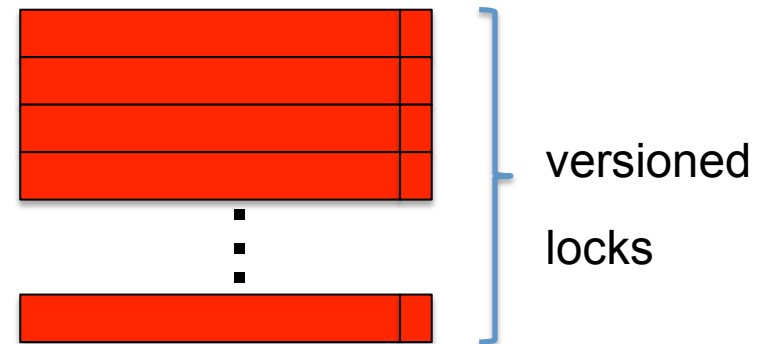
TL2 – Metadados

Compartilhado

Global Clock



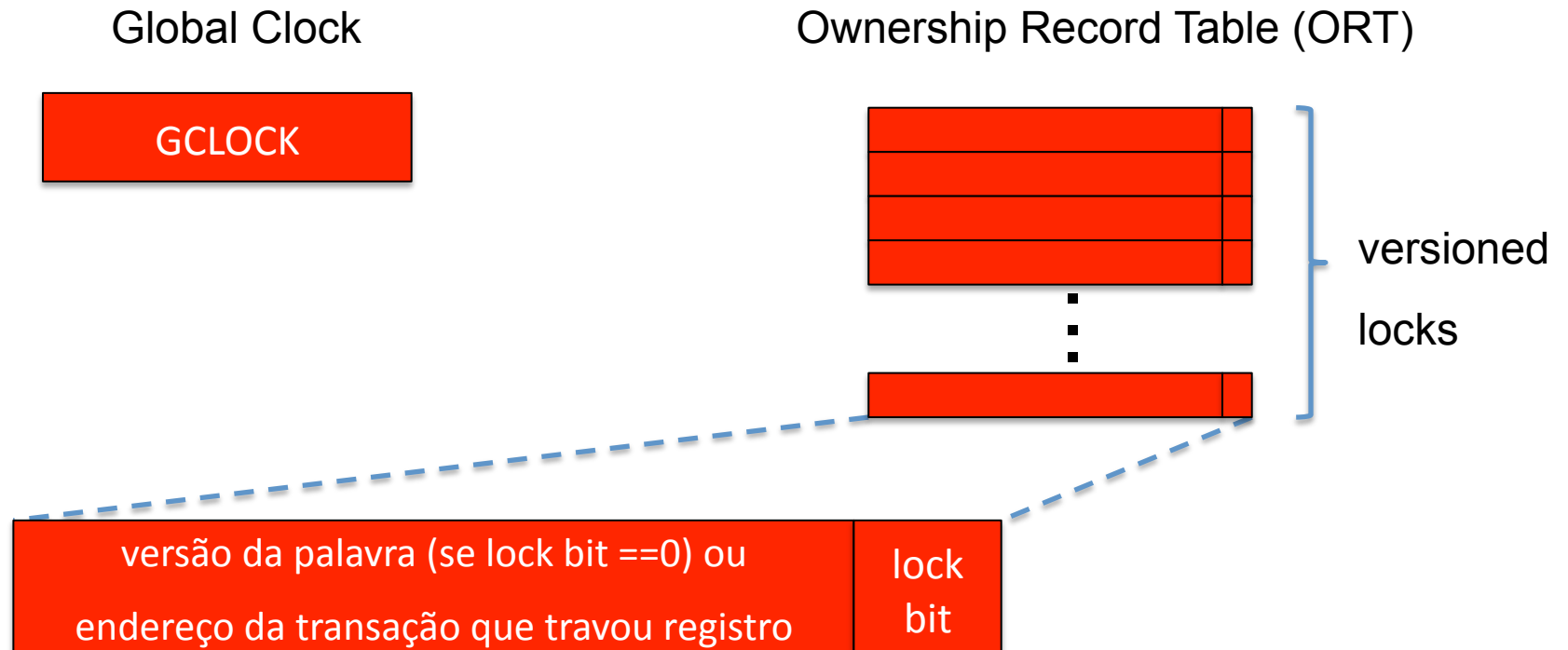
Ownership Record Table (ORT)



Toda posição de memória acessada transacionalmente é mapeada para um registro (**versioned lock**) nessa tabela através de uma função hash

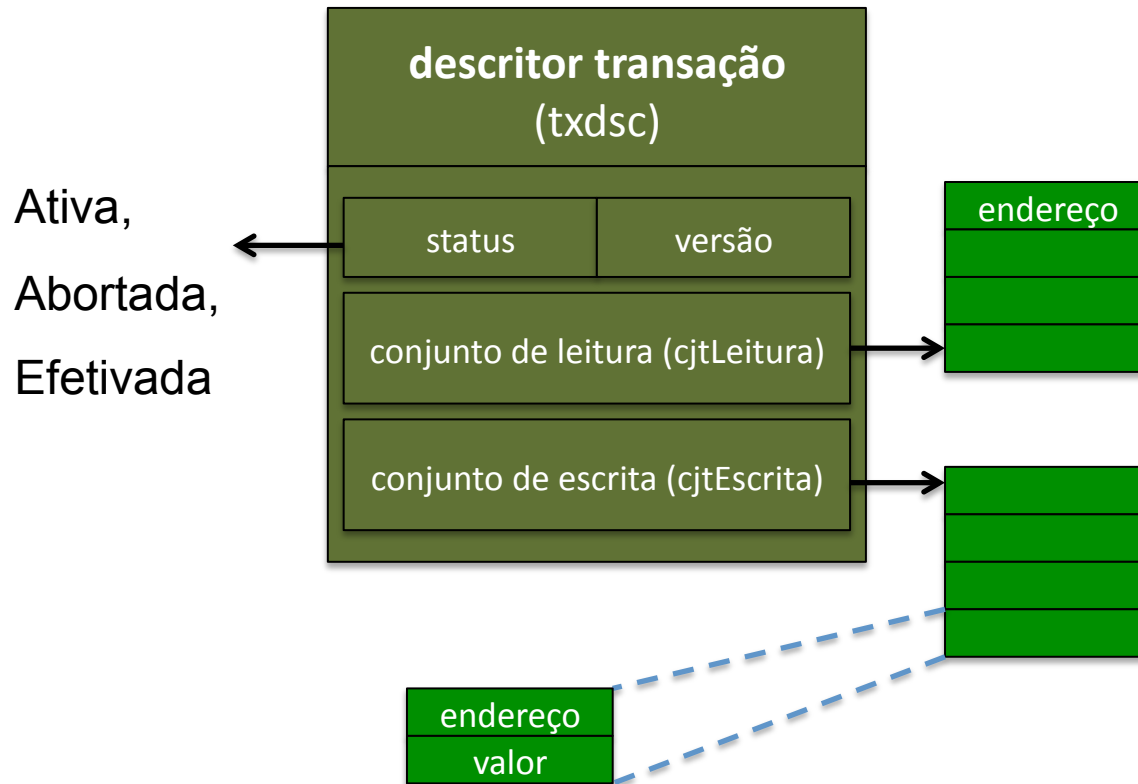
TL2 – Metadados

Compartilhado

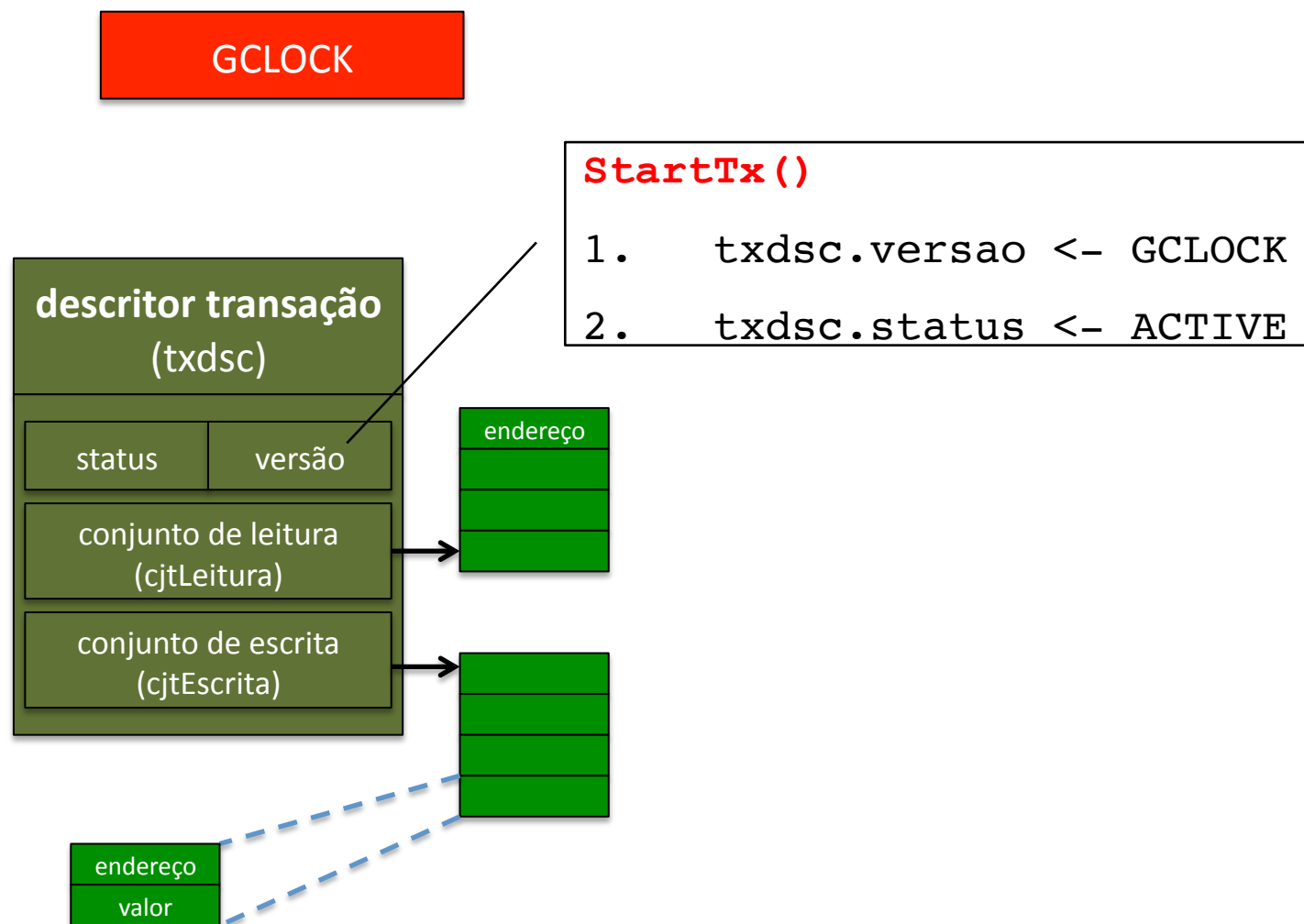


TL2 – Metadados

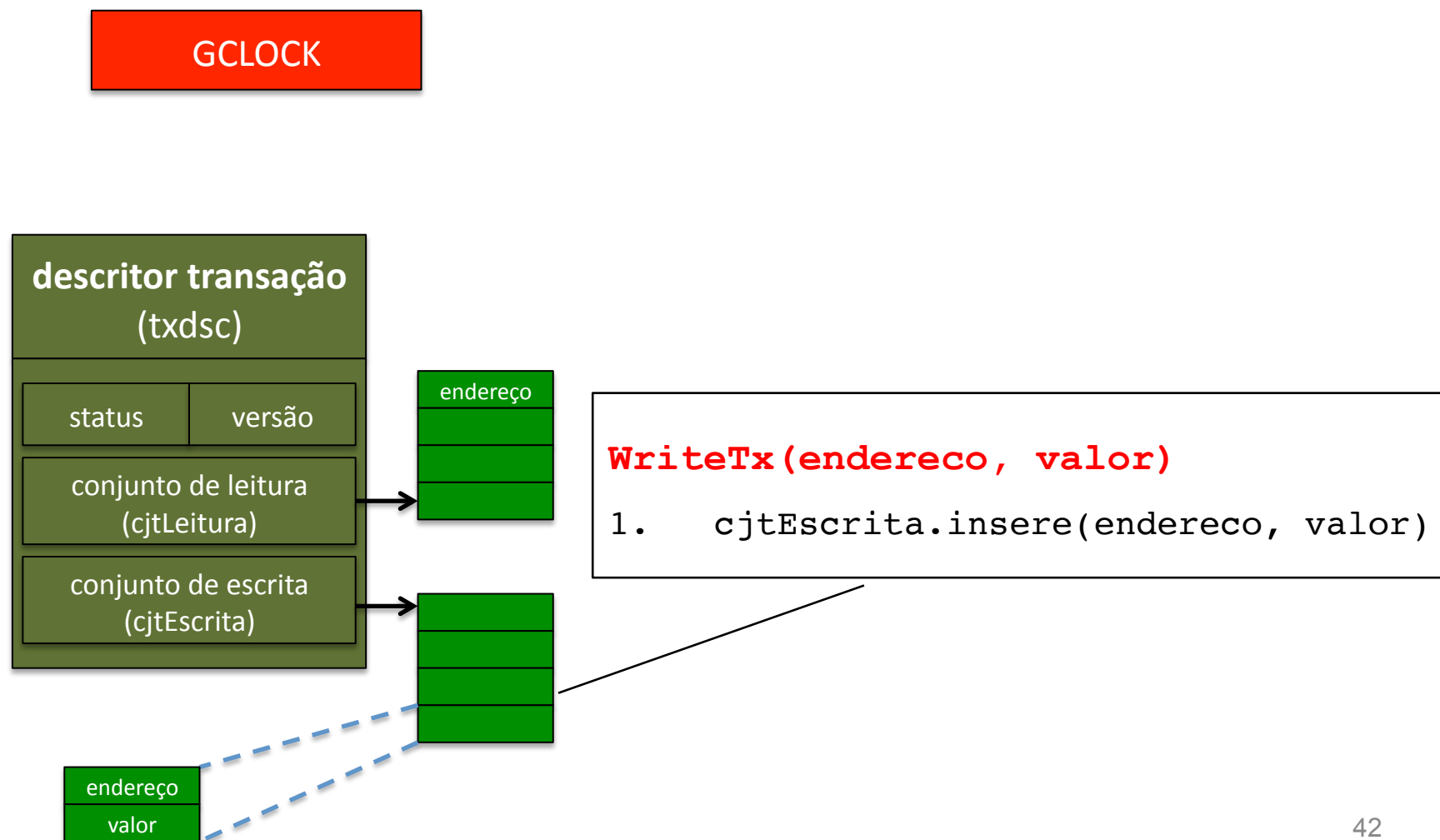
Privado



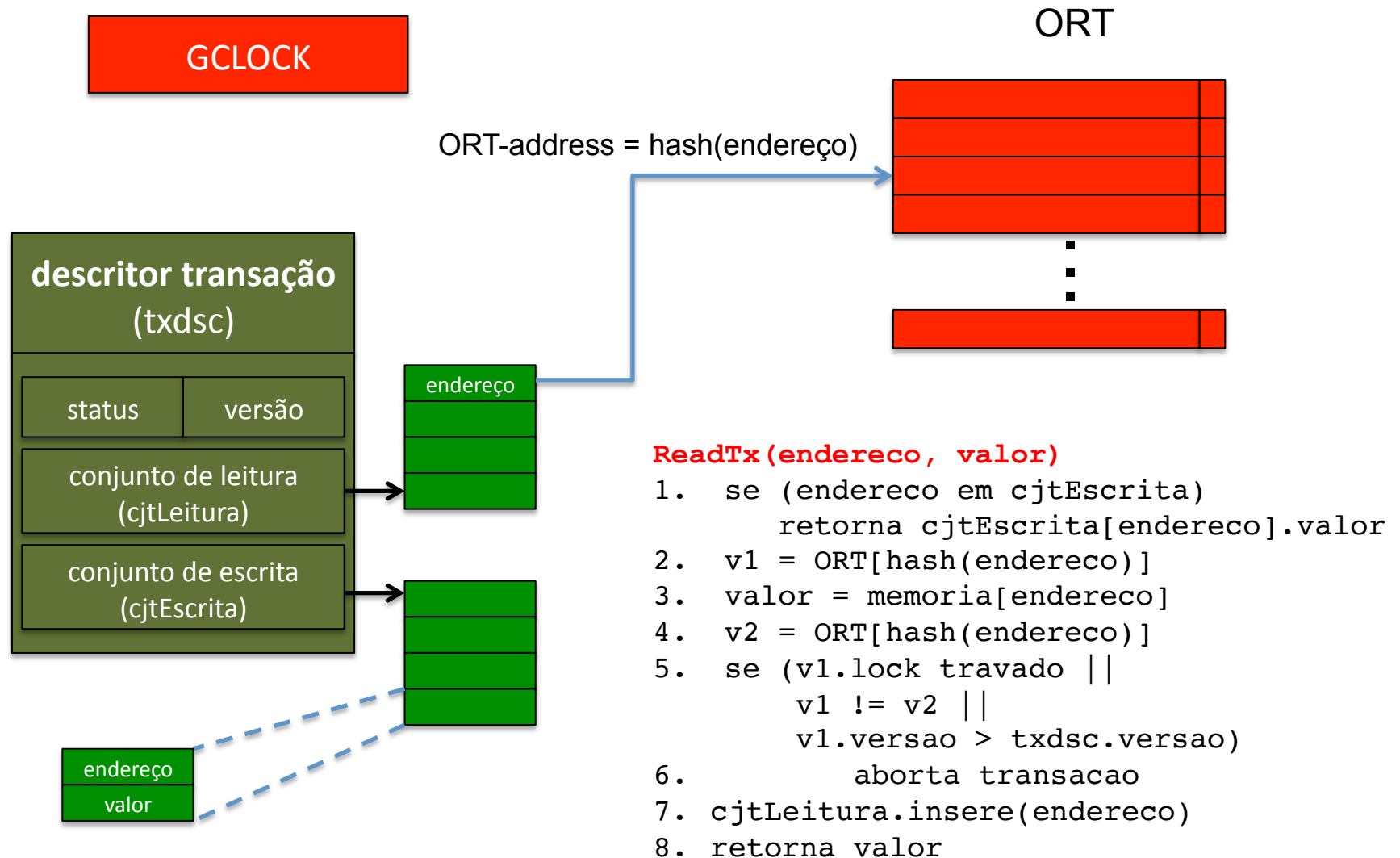
TL2 – Funcionamento



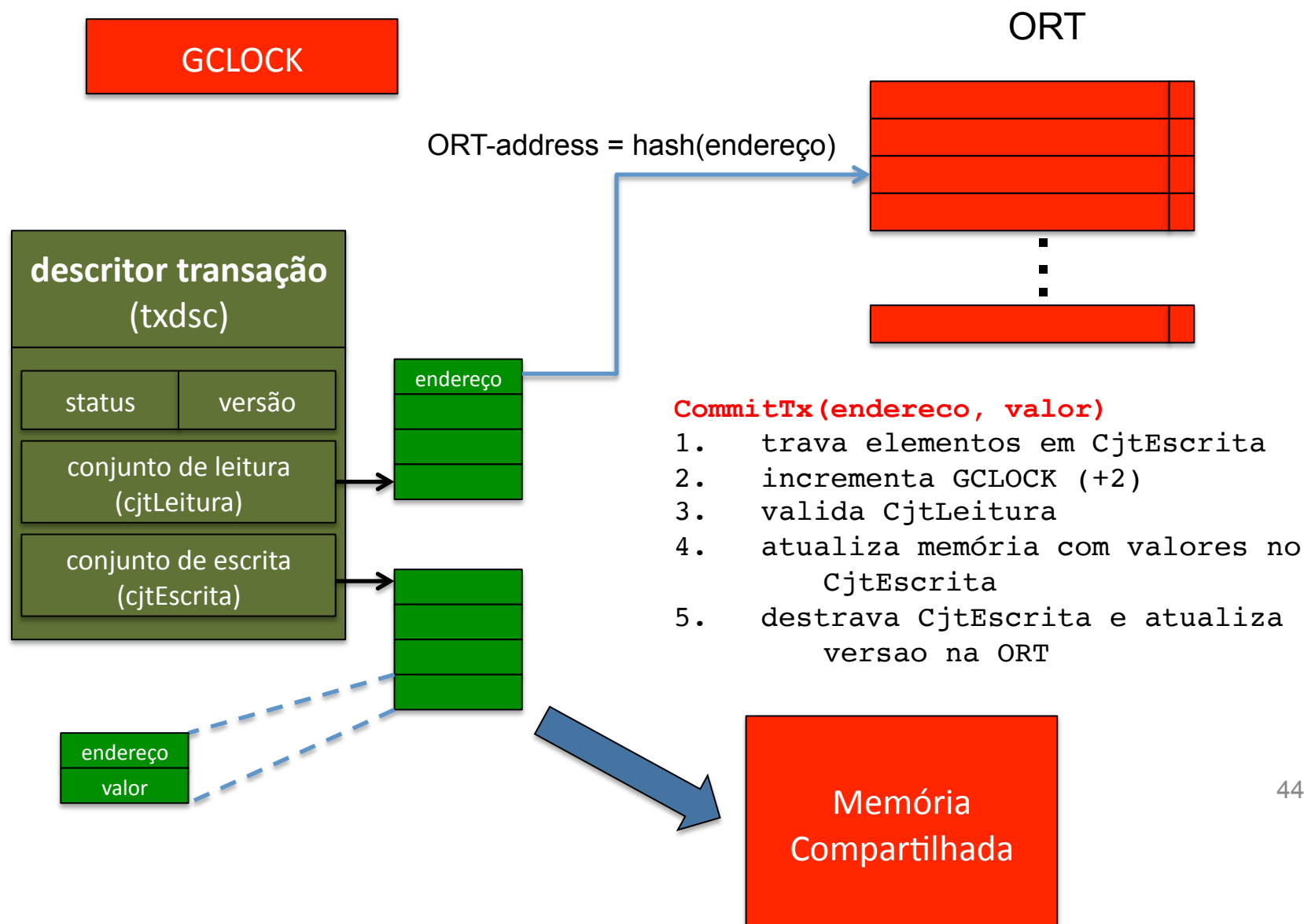
TL2 – Funcionamento



TL2 – Funcionamento



TL2 – Funcionamento



HTM – Suporte

- Interface
 - Conjunto de instruções do processador
 - Exemplo: Intel TSX
- Versionamento
 - Cache ou *buffer* de escrita
- Conflitos
 - Protocolo de coerência de cache (*snoop* ou diretório)
 - R/W bits adicionados à cache

HTM – Exemplo Execução

<u>CACHE 1</u>			
Tag	R	W	
	0	0	
	0	0	
	0	0	
	0	0	

MEMORY	
foo	x=9, y=7
bar	x=0, y=0

<u>CACHE 2</u>			
Tag	R	W	
	0	0	
	0	0	
	0	0	
	0	0	

T1 atomic {
 bar.x = foo.x;
 bar.y = foo.y;
}

T2 atomic {
 t1 = bar.x;
 t2 = bar.y;
}


Versionamento e detecção de conflitos atrasados


HTM – Exemplo Execução

CACHE 1			
Tag	R	W	
foo.x	1	0	9
bar.x	0	1	9
	0	0	
	0	0	

MEMORY	
foo	x=9, y=7
bar	x=0, y=0

CACHE 2			
Tag	R	W	
	0	0	
	0	0	
	0	0	
	0	0	

T1 atomic {
 bar.x = foo.x; 
 bar.y = foo.y;
 }

T2 atomic { 
 t1 = bar.x;
 t2 = bar.y;
 }

HTM – Exemplo Execução

CACHE 1


Tag	R	W	
foo.x	1	0	9
bar.x	0	1	9
	0	0	
	0	0	


MEMORY

foo	x=9, y=7
bar	x=0, y=0

CACHE 2

Tag	R	W	
bar.x	1	0	0
t1	0	1	0
	0	0	
	0	0	

T1 atomic {
 bar.x = foo.x; 
 bar.y = foo.y;
 }

T2 atomic {
 t1 = bar.x; 
 t2 = bar.y;
 }

HTM – Exemplo Execução

CACHE 1


Tag	R	W	
foo.x	1	0	9
bar.x	0	1	9
foo.y	1	0	7
bar.y	0	1	7


MEMORY

foo	x=9, y=7
bar	x=0, y=0

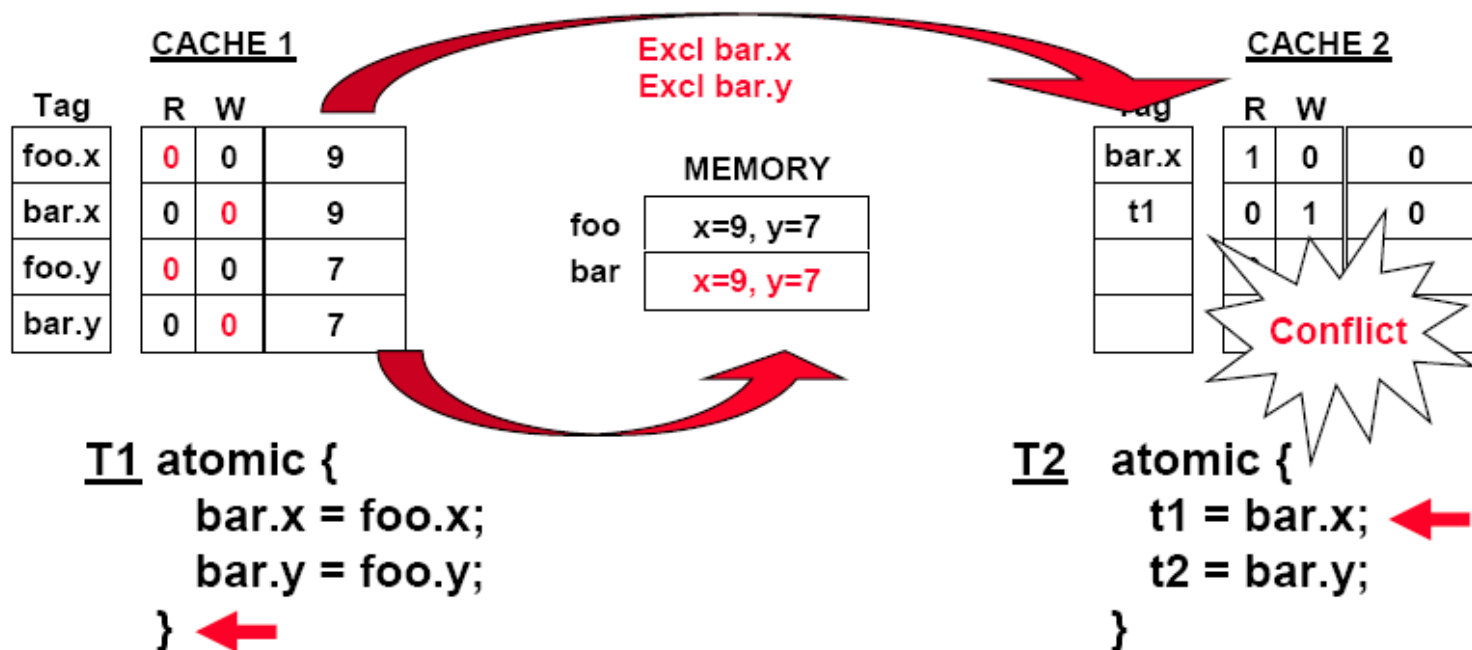
CACHE 2

Tag	R	W	
bar.x	1	0	0
t1	0	1	0
	0	0	
	0	0	

T1 atomic {
 bar.x = foo.x;
 bar.y = foo.y; 
 }

T2 atomic {
 t1 = bar.x; 
 t2 = bar.y;
 }

HTM – Exemplo Execução

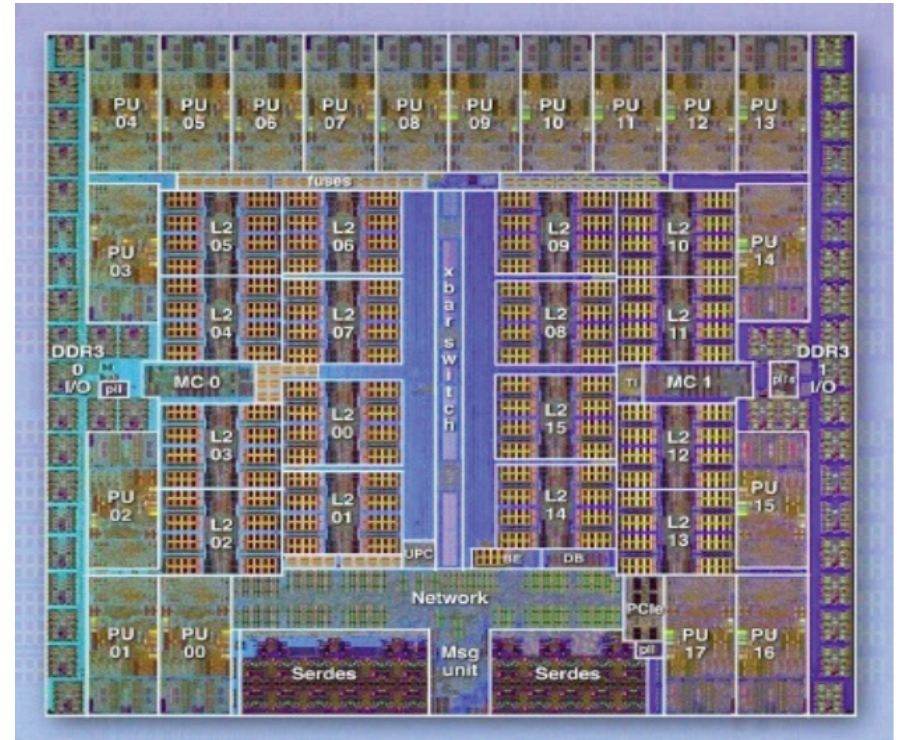


Novos HTMs

- 2009 - Sun's Rock
- 2009 – AMD Advanced Synchronization Facility (ASF)
- 2011 - IBM BlueGene/Q
- 2013 – Intel Haswell
 - Transactional Synchronization Extensions (TSX)

IBM BlueGene/Q

- Revelado no Hot Chips 2011
 - 18 cores @1.6GHz
 - 16 cores para aplicações, 1 para SO, 1 aumentar yield
- Usados no supercomputador Sequoia
- Each core
 - 1.47 Billion transistors
 - 55 Watts



Comentários

- Memória Transacional veio para ficar como um novo paradigma de programação paralela
- Duas grandes empresas na área lançaram processadores contendo extensões para TMs
 - IBM (BlueGene/Q e POWER8)
 - Intel (TSX)