

Transactional Memories: a theoretical introduction



Selim Arsever & Pascal Perez

Shared Memory Problems

$$M(w,r,v) := w(f).M\langle w,r,f\rangle + \underline{r}\langle v\rangle.M\langle w,r,v\rangle$$

$$A(w,r) := \underline{w}\langle v\rangle.WORK.r(v').[v'=v].A$$

$$\begin{aligned} & \underline{A}\langle w,r\rangle \mid \underline{A}\langle w,r\rangle \mid M\langle w,r,v\rangle = \\ & \underline{w}\langle v\rangle.WORK.r(v').[v'=v].A \mid \\ & \underline{w}\langle v\rangle.WORK.r(v').[v'=v].A \mid \\ & w(f).M\langle w,r,f\rangle + \underline{r}\langle v\rangle.M\langle w,r,v\rangle \end{aligned}$$

Shared Memory Problems

$$M(w,r,v) := w(f).M\langle w,r,f\rangle + \underline{r}\langle v\rangle.M\langle w,r,v\rangle$$

$$A(w,r) := \underline{w}\langle v\rangle.WORK.r(v').[v'=v].A$$

→

$$\underline{w}\langle a\rangle.WORK.r(v').[v'=a].A \mid$$

$$\underline{w}\langle v\rangle.WORK.r(v').[v'=v].A \mid$$

$$w(a).M\langle w,r,f\rangle + \underline{r}\langle v\rangle.M\langle w,r,v\rangle$$

Shared Memory Problems

$$M(w,r,v) := w(f).M\langle w,r,f \rangle + \underline{r}\langle v \rangle.M\langle w,r,v \rangle$$

$$A(w,r) := \underline{w}\langle v \rangle.WORK.r(v').[v'=v].A$$

→

$$\underline{w}\langle a \rangle.WORK.r(v').[v'=a].A \mid$$

$$\underline{w}\langle v \rangle.WORK.r(v').[v'=v].A \mid$$

$$w(a).M\langle w,r,a \rangle + \underline{r}\langle v \rangle.M\langle w,r,v \rangle$$

Shared Memory Problems

$$M(w,r,v) := w(f).M\langle w,r,f \rangle + \underline{r}\langle v \rangle.M\langle w,r,v \rangle$$

$$A(w,r) := \underline{w}\langle v \rangle.WORK.r(v').[v'=v].A$$

→

$$\underline{w}\langle a \rangle.WORK.r(v').[v'=a].A \mid$$

$$\underline{w}\langle v \rangle.WORK.r(v').[v'=v].A \mid$$

$$w(a).M\langle w,r,a \rangle + \underline{r}\langle v \rangle.M\langle w,r,v \rangle$$

Shared Memory Problems

$$M(w,r,v) := w(f).M\langle w,r,f \rangle + \underline{r}\langle v \rangle.M\langle w,r,v \rangle$$

$$A(w,r) := \underline{w}\langle v \rangle.WORK.r(v').[v'=v].A$$

→

$$\underline{w}\langle a \rangle.WORK.r(v').[v'=a].A \mid$$

$$\underline{w}\langle v \rangle.WORK.r(v').[v'=v].A \mid$$

$$w(f).M\langle w,r,f \rangle + \underline{r}\langle a \rangle.M\langle w,r,a \rangle$$

Shared Memory Problems

$$M(w,r,v) := w(f).M\langle w,r,f \rangle + \underline{r}\langle v \rangle.M\langle w,r,v \rangle$$

$$A(w,r) := \underline{w}\langle v \rangle.WORK.r(v').[v'=v].A$$

→

$$\underline{w}\langle a \rangle.WORK.r(v').[v'=a].A \mid$$

$$\underline{w}\langle b \rangle.WORK.r(v').[v'=b].A \mid$$

$$w(b).M\langle w,r,b \rangle + \underline{r}\langle a \rangle.M\langle w,r,a \rangle$$

Shared Memory Problems

$$M(w,r,v) := w(f).M\langle w,r,f \rangle + \underline{r}\langle v \rangle.M\langle w,r,v \rangle$$

$$A(w,r) := \underline{w}\langle v \rangle.WORK.r(v').[v'=v].A$$

→

$$\underline{w}\langle a \rangle.WORK.r(v').[v'=a].A \mid$$

$$\underline{w}\langle b \rangle.WORK.r(v').[v'=b].A \mid$$

$$w(b).M\langle w,r,b \rangle + \underline{r}\langle a \rangle.M\langle w,r,a \rangle$$

Shared Memory Problems

$$M(w,r,v) := w(f).M\langle w,r,f\rangle + \underline{r}\langle v\rangle.M\langle w,r,v\rangle$$

$$A(w,r) := \underline{w}\langle v\rangle.WORK.r(v').[v'=v].A$$

→

$$\underline{w}\langle a\rangle.WORK.r(v').[v'=a].A \mid$$

$$\underline{w}\langle b\rangle.WORK.r(v').[v'=b].A \mid$$

$$w(f).M\langle w,r,f\rangle + \underline{r}\langle b\rangle.M\langle w,r,b\rangle$$

Shared Memory Problems

$$M(w,r,v) := w(f).M\langle w,r,f \rangle + \underline{r}\langle v \rangle.M\langle w,r,v \rangle$$

$$A(w,r) := \underline{w}\langle v \rangle.WORK.r(v').[v'=v].A$$

→

$$\underline{w}\langle a \rangle.WORK.r(v').[b=a].A \mid$$

$$\underline{w}\langle b \rangle.WORK.r(v').[v'=b].A \mid$$

$$w(f).M\langle w,r,f \rangle + \underline{r}\langle b \rangle.M\langle w,r,b \rangle$$

Shared Memory Problems

$$M(w,r,v) := w(f).M\langle w,r,f\rangle + \underline{r}\langle v\rangle.M\langle w,r,v\rangle$$

$$A(w,r) := \underline{w}\langle v\rangle.WORK.r(v').[v'=v].A$$

→

$$\underline{w}\langle a\rangle.WORK.r(v').[b=a].A \mid$$

$$\underline{w}\langle b\rangle.WORK.r(v').[v'=b].A \mid$$

$$w(f).M\langle w,r,f\rangle + \underline{r}\langle b\rangle.M\langle w,r,b\rangle$$

Locking is dangerous



- What if a thread fails whilst holding a lock?
- Deadlocks happen!
- Linux pros on <http://lwn.net/Articles/86859/>

Locking does not scale well

Fine grained locked data structures implementation exist of the shelf but...

Global knowlegde!



The screenshot shows a web browser window displaying the Java documentation for the `ConcurrentHashMap` class. The browser's address bar shows the URL `http://java.sun.com/226/1.5.0/docs/api/`. The page title is "ConcurrentHashMap (Java 2 Platform SE 5.0) - Mozilla Firefox". The left sidebar contains a navigation menu with links to various Java packages and classes, including `java.io`, `java.lang`, `java.lang.annotation`, `java.lang.instrument`, `java.lang.management`, `java.lang.ref`, `java.lang.reflect`, `java.math`, `java.net`, `java.nio`, `java.nio.channels`, and `java.nio.channels.spi`. The main content area shows the "Overview" tab for the `ConcurrentHashMap` class. The class signature is `Class ConcurrentHashMap<K,V>`. It is listed as implementing the `Serializable`, `ConcurrentMap`, and `Map` interfaces. The class description states: "A hash table supporting full concurrency of retrievals and adjustable expected concurrency for updates. This class obeys the same functional specification as `Hashtable`, and includes versions of methods corresponding to each method of `Hashtable`. However, even though all operations are thread-safe, retrieval operations do *not* entail locking, and there is *not* any support for locking the entire table in a way that prevents all access. This class is fully interoperable with `Hashtable` in programs that rely on its thread safety but not on its synchronization details." The page also includes sections for "Type Parameters", "All Implemented Interfaces", and "Retrieval operations".

Transaction

Atomicity

Consistency

Isolation

Durability

Transaction

$A + B + C$ should remain constant under the execution of both transactions in any order.

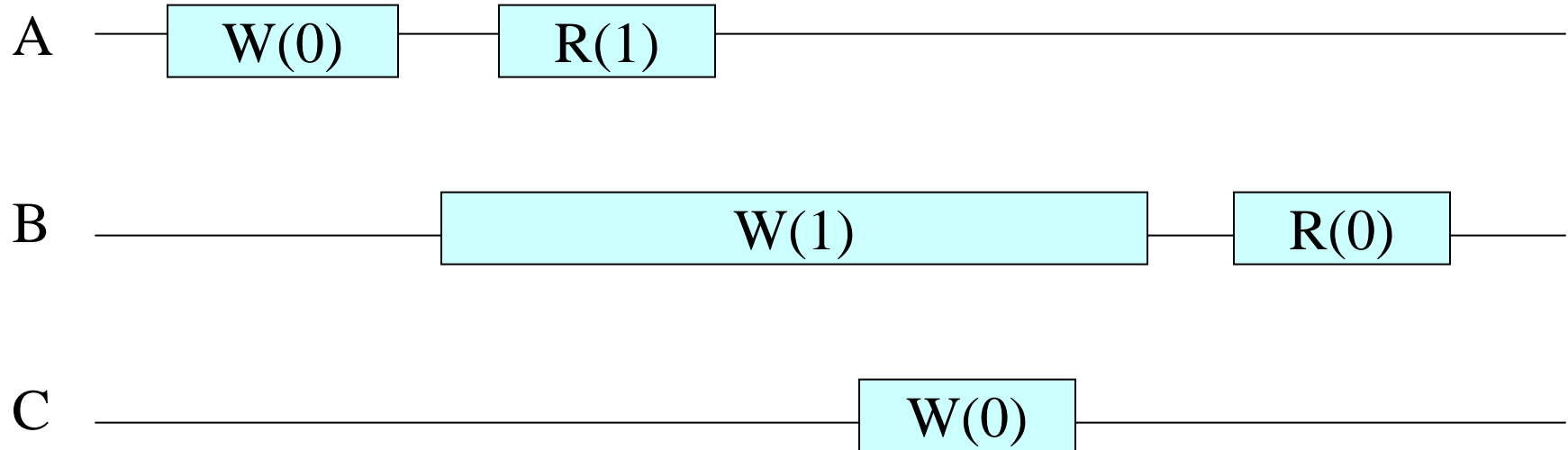
Correct:

$A = A - 10$	lock L_1
lock L_1	$B = B - 20$
$B = B + 10$	unlock L_1
unlock L_1	$C = C + 20$

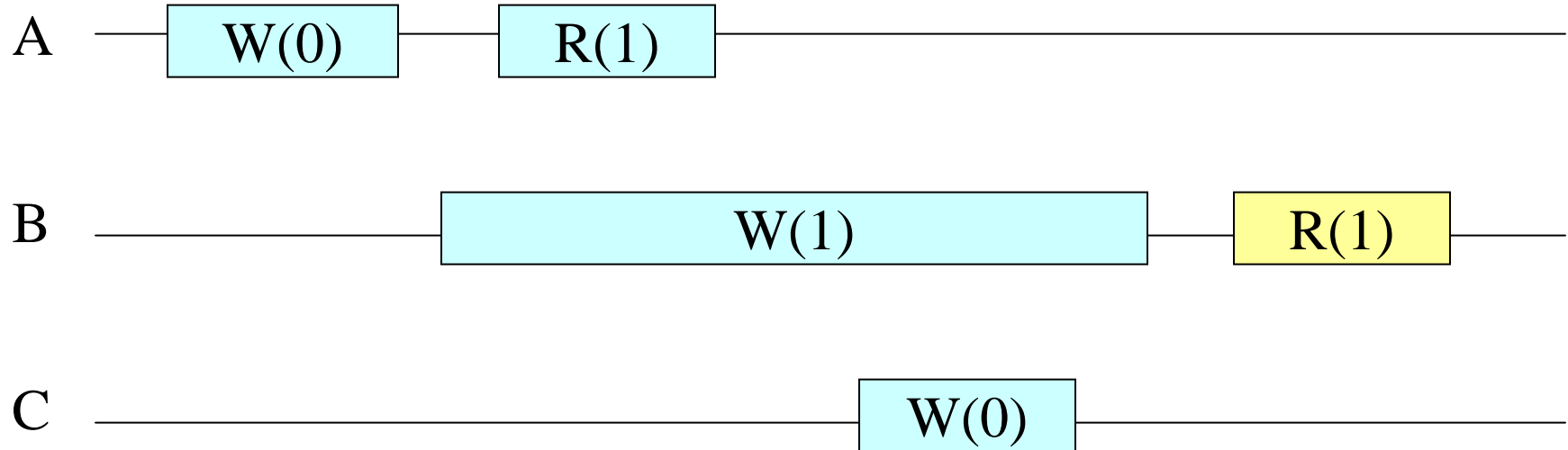
Wrong:

lock L_1	lock L_2
$A = A - 10$	$B = B - 20$
unlock L_1	unlock L_2
$B = B + 10$	$C = C + 20$

Transactional Memory

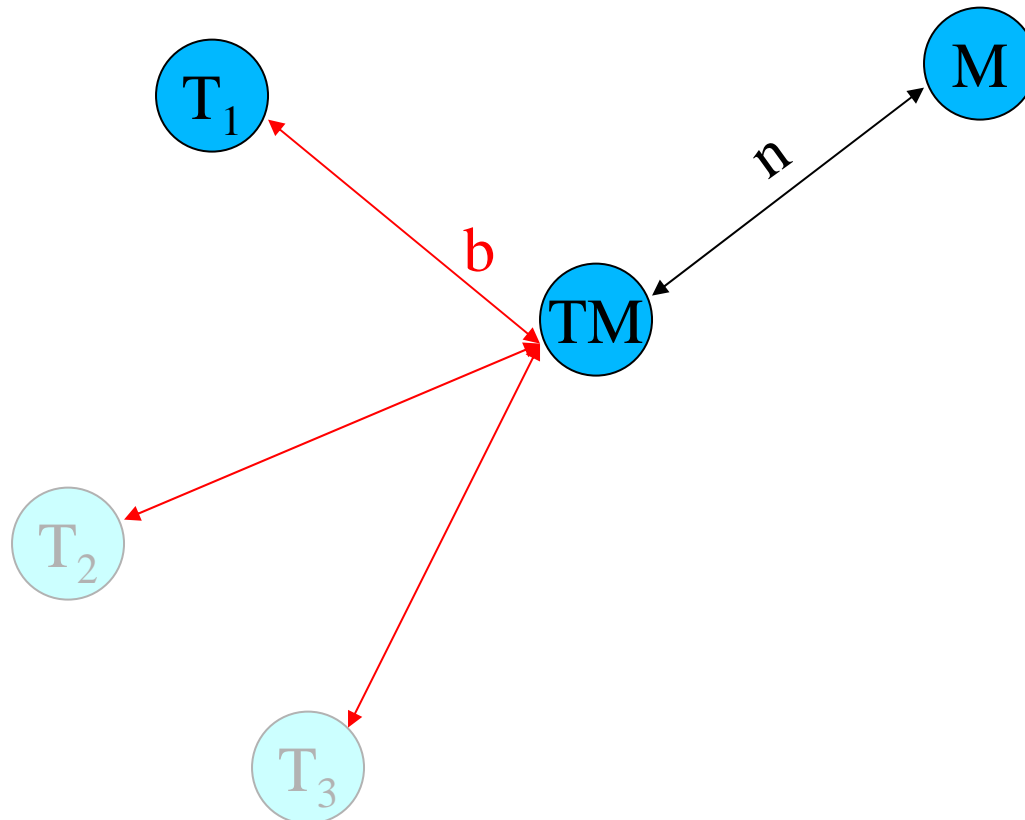


Transactional Memory



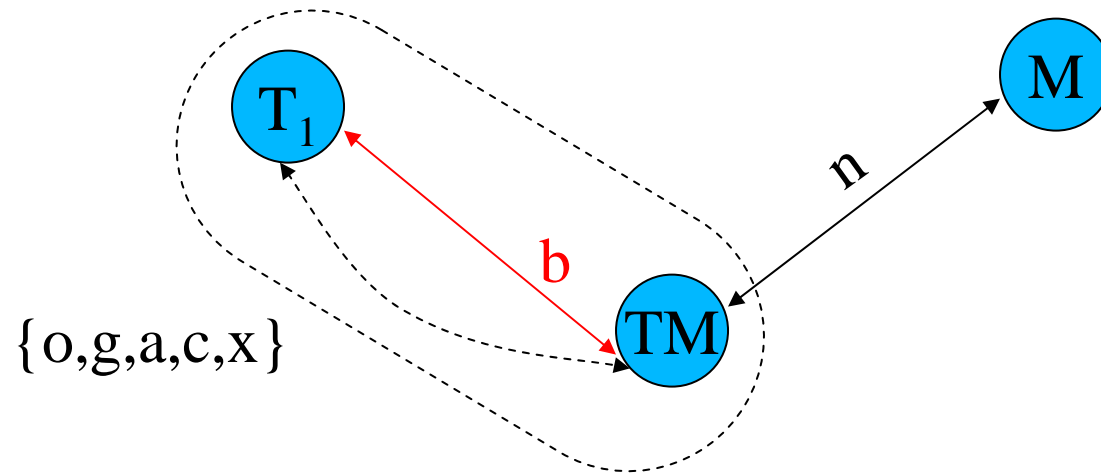
Transactional Memory

Initial Situation:



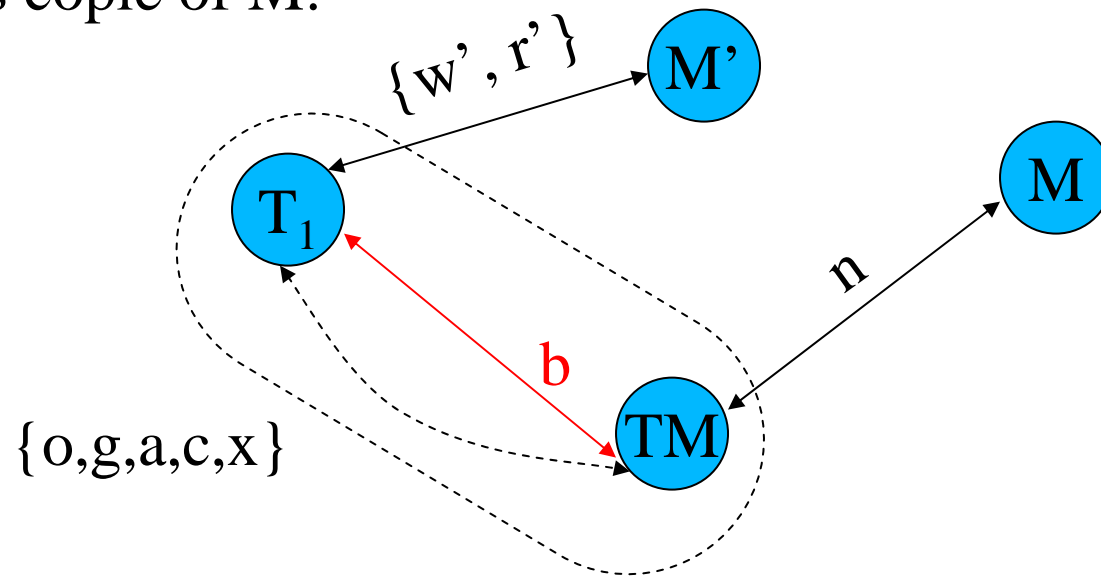
Transactional Memory

T_1 gives his channels to TM:



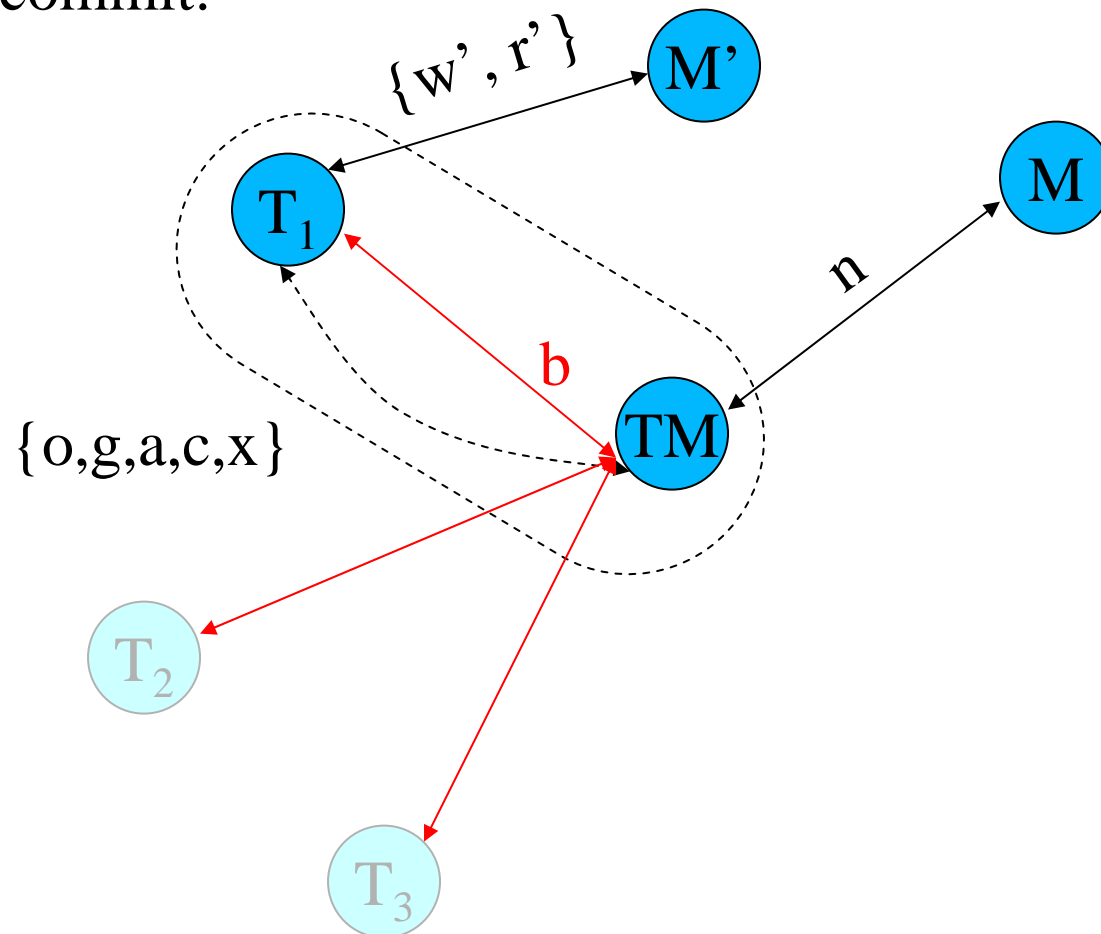
Transactional Memory

T_1 gets its copy of M :



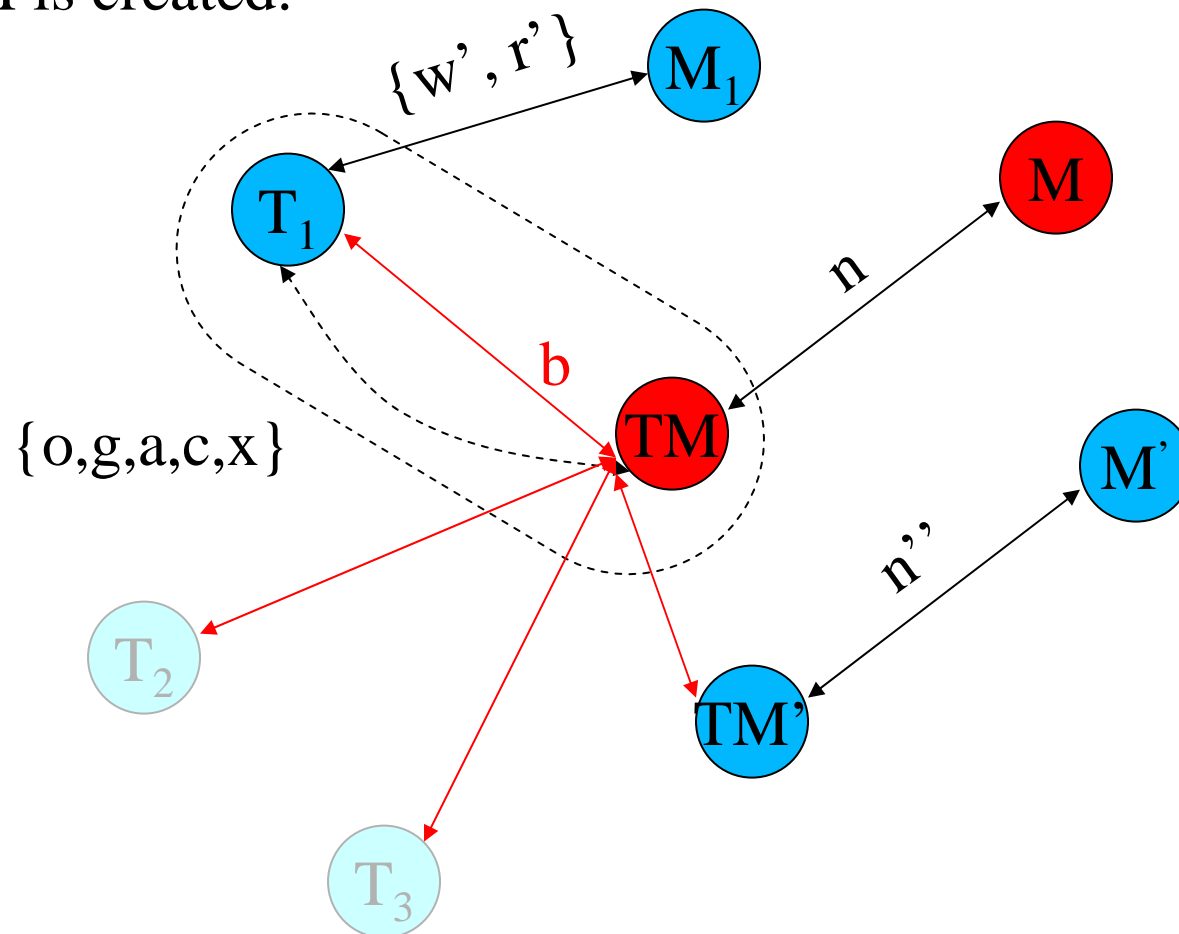
Transactional Memory

T_1 tries to commit:



Transactional Memory

A new TM is created:



Concurrent Commit

$T_1 : \underline{c}_1.(c_1 + x_1.T_1\langle b \rangle)$

$T_2 : \underline{c}_2.(c_2 + x_2.T_2\langle b \rangle)$

$TM : \underline{\text{crash}} \mid !\dots \mid k.\underline{x}_1 \mid k.\underline{x}_2 \mid ! p(x).k.\underline{x} \mid$

$c_1.\text{crash}.n_1'(w'', r'', n'').(TM\langle b, n'' \rangle \mid !\underline{k} \mid \underline{c}_1) \mid$

$c_2.\text{crash}.n_2'(w'', r'', n'').(TM\langle b, n'' \rangle \mid !\underline{k} \mid \underline{c}_2)$

Concurrent Commit

$T_1 : \underline{c}_1.(c_1 + x_1.T_1\langle b \rangle)$

$T_2 : \underline{c}_2.(c_2 + x_2.T_2\langle b \rangle)$

$TM : \underline{\text{crash}} \mid !\dots \mid k.\underline{x}_1 \mid k.\underline{x}_2 \mid ! p(x).k.\underline{x} \mid$

$c_1.\text{crash}.n_1'(w'', r'', n'').(TM\langle b, n'' \rangle \mid !\underline{k} \mid \underline{c}_1) \mid$

$c_2.\text{crash}.n_2'(w'', r'', n'').(TM\langle b, n'' \rangle \mid !\underline{k} \mid \underline{c}_2)$

Concurrent Commit

$T_1 : \underline{c}_1.(c_1 + x_1.T_1\langle b \rangle)$

$T_2 : \underline{c}_2.(c_2 + x_2.T_2\langle b \rangle)$

$TM : \underline{\text{crash}} \mid !\dots \mid k.\underline{x}_1 \mid k.\underline{x}_2 \mid ! p(x).k.\underline{x} \mid$

$c_1.\text{crash}.n_1'(w'', r'', n'').(TM\langle b, n'' \rangle \mid !\underline{k} \mid \underline{c}_1) \mid$

$c_2.\text{crash}.n_2'(w'', r'', n'').(TM\langle b, n'' \rangle \mid !\underline{k} \mid \underline{c}_2)$

Concurrent Commit

$T_1 : \underline{c}_1.(c_1 + x_1.T_1\langle b \rangle)$

$T_2 : \underline{c}_2.(c_2 + x_2.T_2\langle b \rangle)$

TM : crash | !... | k.x₁ | k.x₂ | ! p(x).k.x |

$c_1.\text{crash}.n_1'(w'', r'', n'').(TM\langle b, n'' \rangle | !\underline{k} | \underline{c}_1) |$

$c_2.\text{crash}.n_2'(w'', r'', n'').(TM\langle b, n'' \rangle | !\underline{k} | \underline{c}_2)$

Concurrent Commit

$T_1 : \underline{c}_1.(c_1 + x_1.T_1\langle b \rangle)$

$T_2 : \underline{c}_2.(c_2 + x_2.T_2\langle b \rangle)$

$TM : \underline{\text{crash}} \mid !\dots \mid k.\underline{x}_1 \mid k.\underline{x}_2 \mid ! p(x).k.\underline{x} \mid$

$c_1.\text{crash}.n_1'(w'', r'', n'').(TM\langle b, n'' \rangle \mid !\underline{k} \mid \underline{c}_1) \mid$

$c_2.\text{crash}.n_2'(w'', r'', n'').(TM\langle b, n'' \rangle \mid !\underline{k} \mid \underline{c}_2)$

Concurrent Commit

$T_1 : \underline{c}_1.(c_1 + x_1.T_1\langle b \rangle)$

$T_2 : \underline{c}_2.(c_2 + x_2.T_2\langle b \rangle)$

$TM : \underline{\text{crash}} \mid !\dots \mid \underline{k}.x_1 \mid \underline{k}.x_2 \mid ! p(x).k.x \mid$

$c_1.\text{crash}.n_1'(w'', r'', n'').(TM\langle b, n'' \rangle \mid !\underline{k} \mid \underline{c}_1) \mid$

$c_2.\text{crash}.n_2'(w'', r'', n'').(TM\langle b, n'' \rangle \mid !\underline{k} \mid \underline{c}_2)$

Concurrent Commit

$T_1 : \underline{c}_1.(c_1 + \mathbf{x}_1.T_1\langle b \rangle)$

$T_2 : \underline{c}_2.(c_2 + \mathbf{x}_2.T_2\langle b \rangle)$

$TM : \underline{\text{crash}} \mid !\dots \mid k.\underline{\mathbf{x}}_1 \mid k.\underline{\mathbf{x}}_2 \mid ! p(\mathbf{x}).k.\underline{\mathbf{x}} \mid$

$c_1.\text{crash}.n_1'(w'', r'', n'').(TM\langle b, n'' \rangle \mid !\underline{k} \mid \underline{c}_1) \mid$

$c_2.\text{crash}.n_2'(w'', r'', n'').(TM\langle b, n'' \rangle \mid !\underline{k} \mid \underline{c}_2)$

Concurrent Commit

$T_1 : \underline{c}_1.(c_1 + x_1.T_1\langle b \rangle)$

$T_2 : \underline{c}_2.(c_2 + x_2.T_2\langle b \rangle)$

$TM : \underline{\text{crash}} \mid !\dots \mid k.\underline{x}_1 \mid k.\underline{x}_2 \mid ! p(x).k.\underline{x} \mid$

$c_1.\text{crash}.n_1'(w'', r'', n'').(TM\langle b, n'' \rangle \mid !\underline{k} \mid \underline{c}_1) \mid$

$c_2.\text{crash}.n_2'(w'', r'', n'').(TM\langle b, n'' \rangle \mid !\underline{k} \mid \underline{c}_2)$

Exceptions Enhancement

- Avoiding Unnecessary Exceptions

TM := (... | !p(x,cancel).(k.x + cancel))

G := ... (vcancel)(p<x,cancel>.(... | cancel.!k)

- Allowing Needed Exceptions

Commit while Starting

$T_1 : \underline{c}_1.(c_1 + x_1.T_1\langle b \rangle)$

$T_2 : \underline{b}\langle o_2, g_2, a_2, c_2, x_2 \rangle.\underline{o}.g \dots$

$TM : \underline{\text{crash}} \mid !\dots \mid k.\underline{x}_1 \mid ! p(x).k.\underline{x} \mid$

$c_1.\text{crash}.n_1'(w'', r'', n'').(TM\langle b, n'' \rangle \mid !\underline{k} \mid \underline{c}_1) \mid$

$b(o_2, g_2, a_2, c_2, x_2).p\langle x_2 \rangle.\underline{o} \dots$

Commit while Starting

$T_1 : \underline{c}_1.(c_1 + x_1.T_1\langle b \rangle)$

$T_2 : \underline{b}\langle o_2, g_2, a_2, c_2, x_2 \rangle.\underline{o}.g \dots$

TM : crash | !... | k.x₁ | ! p(x).k.x |

$c_1.\text{crash}.n_1'(w'', r'', n'').(\text{TM}\langle b, n'' \rangle | !\underline{k} | \underline{c}_1) |$

$\underline{b}\langle o_2, g_2, a_2, c_2, x_2 \rangle.\underline{p}\langle x_2 \rangle.\underline{o} \dots$

Commit while Starting

$T_1 : \underline{c}_1.(c_1 + x_1.T_1\langle b \rangle)$

$T_2 : \underline{b}\langle o_2, g_2, a_2, c_2, x_2 \rangle.\underline{o}.g \dots$

$TM : \underline{\text{crash}} \mid !\dots \mid k.\underline{x}_1 \mid ! p(x).k.\underline{x} \mid$

$c_1.\text{crash}.n_1'(w'', r'', n'').(TM\langle b, n'' \rangle \mid !\underline{k} \mid \underline{c}_1) \mid$

$b(o_2, g_2, a_2, c_2, x_2).\underline{p}\langle x_2 \rangle.\underline{o} \dots$

Commit while Starting

$T_1 : \underline{c}_1.(c_1 + x_1.T_1\langle b \rangle)$

$T_2 : \underline{b}\langle o_2, g_2, a_2, c_2, x_2 \rangle.\underline{o}.g \dots$

TM : crash | !... | k.x₁ | ! p(x).k.x |

$c_1.\text{crash}.n_1'(w'', r'', n'').(TM\langle b, n'' \rangle | !\underline{k} | \underline{c}_1) |$

$b(o_2, g_2, a_2, c_2, x_2).p\langle x_2 \rangle.o \dots$

Commit while Starting

$T_1 : \underline{c}_1.(c_1 + x_1.T_1\langle b \rangle)$

$T_2 : \underline{b}\langle o_2, g_2, a_2, c_2, x_2 \rangle.\underline{o}.g \dots$

$TM : \underline{\text{crash}} \mid !\dots \mid k.\underline{x}_1 \mid ! p(x).k.\underline{x} \mid$

$c_1.\text{crash}.n_1'(w'', r'', n'').(TM\langle b, n'' \rangle \mid !\underline{k} \mid \underline{c}_1) \mid$

$b(o_2, g_2, a_2, c_2, x_2).p\langle x_2 \rangle.o \dots$

Commit while Starting

$T_1 : \underline{c}_1.(\underline{c}_1 + x_1.T_1\langle b \rangle)$

$T_2 : \underline{b}\langle o_2, g_2, a_2, c_2, x_2 \rangle.\underline{o}.g \dots$

$TM : \underline{crash} \mid !\dots \mid k.\underline{x}_1 \mid ! p(x).k.\underline{x} \mid$

$c_1.crash.n_1'(w'', r'', n'').(TM\langle b, n'' \rangle \mid !\underline{k} \mid \underline{c}_1) \mid$

$b(o_2, g_2, a_2, c_2, x_2).p\langle x_2 \rangle.o \dots$

Commit while Starting

$T_1 : \underline{c}_1.(c_1 + x_1.T_1\langle b \rangle)$

$T_2 : \underline{b}\langle o_2, g_2, a_2, c_2, x_2 \rangle.\underline{o}.g \dots$

TM : crash | !... | $k.\underline{x}_1$ | $p(x).k.\underline{x}$ | ! $p(x).k.\underline{x}$ |

$c_1.\text{crash}.n_1'(w'', r'', n'').(TM\langle b, n'' \rangle | !\underline{k} | \underline{c}_1) |$

$b(o_2, g_2, a_2, c_2, x_2).p\langle x_2 \rangle.\underline{o} \dots$

Commit while Starting

$T_1 : \underline{c}_1.(c_1 + x_1.T_1\langle b \rangle)$

$T_2 : \underline{b}\langle o_2, g_2, a_2, c_2, x_2 \rangle.\underline{o.g} \dots$

$TM : \underline{\text{crash}} \mid !\dots \mid \underline{k.x}_1 \mid p(x).\underline{k.x}_2 \mid ! p(x).\underline{k.x} \mid$
 $c_1.\text{crash}.n_1'(w'', r'', n'').(TM\langle b, n'' \rangle \mid !\underline{k} \mid \underline{c}_1) \mid$
 $b(o_2, g_2, a_2, c_2, x_2).p\langle x_2 \rangle.\underline{o} \dots$

Commit while Starting

$T_1 : \underline{c}_1.(c_1 + x_1.T_1\langle b \rangle)$

$T_2 : \underline{b}\langle o_2, g_2, a_2, c_2, x_2 \rangle.\underline{o}.g \dots$

$TM : \underline{\text{crash}} \mid !\dots \mid \underline{k}.x_1 \mid p(x).\underline{k}.x_2 \mid ! p(x).\underline{k}.x \mid$
 $c_1.\text{crash}.n_1'(w'', r'', n'').(TM\langle b, n'' \rangle \mid !\underline{k} \mid \underline{c}_1) \mid$
 $b(o_2, g_2, a_2, c_2, x_2).p\langle x_2 \rangle.\underline{o} \dots$