Combining Strong and Eventual Consistency in Distributed TM

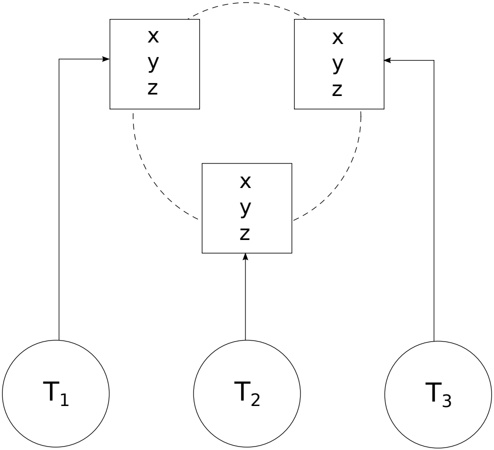
Konrad Siek and Paweł T. Wojciechowski
Poznań University of Technology
{konrad.siek,pawel.t.wojciechowski}@cs.put.edu.pl

10 IX 2014

Distributed Systems Group

http://dsg.cs.put.poznan.pl

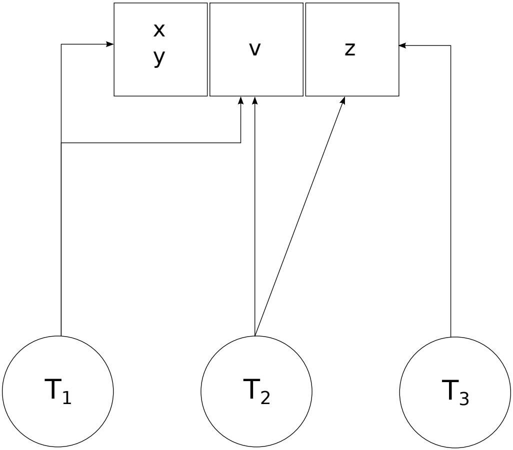
Distributed Transactional Memory



Replicated TM

SRDS'12, ICDCS'13, WTTM'14, SRDS'14

Distributed Transactional Memory



Distributed Transactions

SPAA'13, WTTM'14, HLPP'14 (to appear in IJSS), DISC'14

Pessimistic vs Optimistic TM

Optimistic approach

$T_1 \; [\![\; r(x)1, w(x)2 \;]\!]$

$T_2 \; [\![\quad r(x)1, \quad \searrow w(x)2 \; \circlearrowleft \quad \rightarrow \quad T_2' \; [\![\; r(x)2, w(x)3 \;]\!]$

Pessimistic approach

$T_1 \; [\![\; r(x)1, w(x)2 \;]\!]$

$T_2 \; [\![\qquad\qquad \searrow r(x)2, w(x)3 \;]\!]$

Pessimistic vs Optimistic TM

Optimistic approach

$T_1 \ \llbracket \ r(x)1, w(x)2 \ \rrbracket$

$T_2 \ \llbracket \quad r(x)1, \ \searrow w(x)2 \ \circlearrowleft \quad \rightarrow \quad T_2' \ \llbracket \ r(x)2, w(x)3 \ \rrbracket$

Pessimistic approach

$T_1 \ \llbracket \ r(x)1, w(x)2 \ \rrbracket$

$T_2 \ \llbracket \qquad\qquad \searrow r(x)2, w(x)3 \ \rrbracket$

- Retain the transaction abstraction
- Tolerate high contention
- Safe for irrevocable operations (prevent aborts)

Supremum Versioning Algorithm

SVA in a nutshell:

T_i starts: it gets a version ticket for each resource x, y, z

T_i can access x once T_i's ticket matches x's version counter, otherwise T_i must wait

T_i commits: x, y, z's version counters are incremented (transaction with next ticket can access x, y, z)

Once T_i accesses x for the last time (check *supremum*) x's version counter is incremented

Wojciechowski. *Isolation-only Transactions by Typing and Versioning*. PPDP'05.
Siek, Wojciechowski. *A Formal Design of a Tool for Static Analysis of Upper Bounds on Object Calls.* FMICS'12.
Siek, Wojciechowski. *Atomic RMI: a Distributed Transactional Memory Framework*. HLPP'14.

The joys of early release

Early release on last use

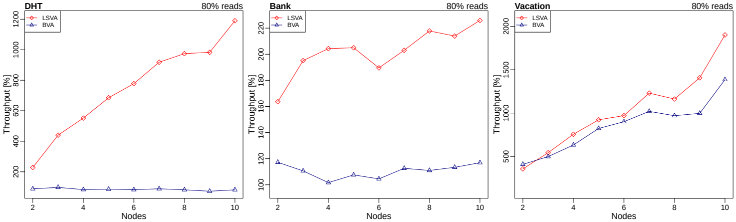
$$T_1 \; [\![\; r(x)1, \textcolor{red}{w(x)2}, r(y)1, w(y)2 \;]\!]$$
$$T_2 \; [\![\qquad\qquad \searrow \textcolor{red}{r(x)2}, w(x)3 \;]\!]$$

The joys of early release

Early release on last use

$$T_1 \; [\![\; r(x)1, \color{red}{w(x)2}, r(y)1, w(y)2 \;]\!]$$
$$T_2 \; [\![\qquad\qquad \searrow \color{red}{r(x)2}, w(x)3 \;]\!]$$

Performance boost:

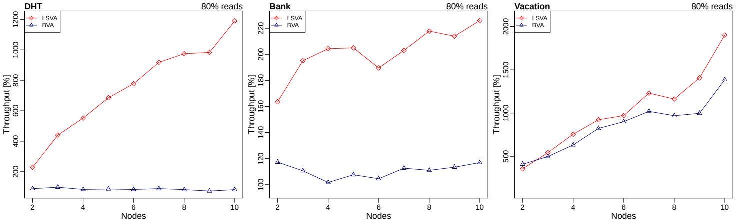


The joys of early release

Early release on last use

$$T_1 \; [\![\; r(x)1, {\color{red} w(x)2}, r(y)1, w(y)2 \;]\!]$$
$$T_2 \; [\![\; \searrow {\color{red} r(x)2}, w(x)3 \;]\!]$$

Performance boost:



Not opaque, but no inconsistent views, because no aborts.

Manual aborts

The case for manual aborts:

- More powerful syntax
- Difficult to implement well in distributed systems
- Necessary for fault tolerance

Siek, Wojciechowski. Brief Announcement: Towards a Fully-Articulated Pessimistic Distributed Transactional Memory. SPAA'13.

Manual aborts

The case for manual aborts:

- More powerful syntax
- Difficult to implement well in distributed systems
- Necessary for fault tolerance

Siek, Wojciechowski. Brief Announcement: Towards a Fully-Articulated Pessimistic Distributed Transactional Memory. SPAA'13.

Cascading abort

$$T_1 \; [\![\; r(x)1, w(x)2, r(y)1, w(y)2, \circlearrowleft$$
$$T_2 \; [\![\qquad \searrow r(x)2, w(x)3 \quad \searrow \circlearrowleft \ldots$$

Manual aborts

The case for manual aborts:

- More powerful syntax
- Difficult to implement well in distributed systems
- Necessary for fault tolerance

Siek, Wojciechowski. Brief Announcement: Towards a Fully-Articulated Pessimistic Distributed Transactional Memory. SPAA'13.

Cascading abort

$$\begin{array}{ll} T_1 & [\![\; r(x)1, \color{red}{w(x)2}, r(y)1, w(y)2, \; \color{red}{\circlearrowleft} \\ T_2 & [\![\; \qquad\qquad \searrow r(x)2, w(x)3 \quad \searrow \circlearrowleft \ldots \end{array}$$

Not opaque and it matters now.

Inconsistent views

Precludes overwriting:

$$T_i \;[\![\; \mathrm{w}(x)0,\quad \mathrm{w}(x)1 \;]\!]$$
$$T_j \qquad [\![\; \searrow \mathrm{r}(x)0 \;\; \searrow \circlearrowleft T'_j \;[\![\; \mathrm{r}(x)1, \mathrm{w}(x)2 \;]\!]$$

Allowed inconsistent view:

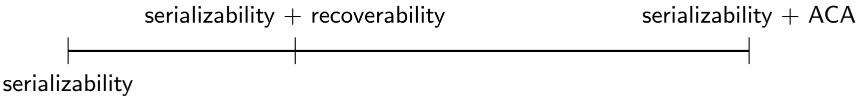
$$T_i \;[\![\; \mathrm{w}(x)0,\quad \mathrm{w}(x)1 \qquad \circlearrowleft$$
$$T_j \qquad\qquad [\![\; \searrow \mathrm{r}(x)0 \;\; \searrow \circlearrowleft T'_j [\![\; \mathrm{r}(x)1, \mathrm{w}(x)2 \;]\!]$$

Safety properties for TMs with early release

- Serializability
- Elastic Opacity
- Virtual World Consistency
- TMS1 & TMS2
- Recoverability
- Avoiding Cascading Aborts
- Strictness
- Rigorousness

Safety properties for TMs with early release

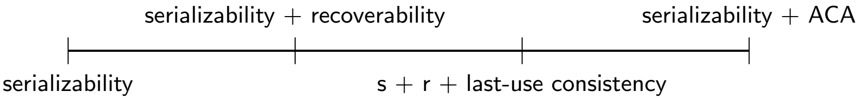
- Serializability
- Elastic Opacity
- Virtual World Consistency
- TMS1 & TMS2
- Recoverability
- Avoiding Cascading Aborts
- Strictness
- Rigorousness



Siek, Wojciechowski. Zen and the Art of Concurrency Control: An Exploration of TM Safety Property Space with Early Release in Mind. WTTM'14.

Safety properties for TMs with early release

- Serializability
- Elastic Opacity
- Virtual World Consistency
- TMS1 & TMS2
- Recoverability
- Avoiding Cascading Aborts
- Strictness
- Rigorousness



Siek, Wojciechowski. Zen and the Art of Concurrency Control: An Exploration of TM Safety Property Space with Early Release in Mind. WTTM'14.

Last-use opacity

Components of opacity:

- Serializability
- Real-time order
- Consistency

Last-use opacity

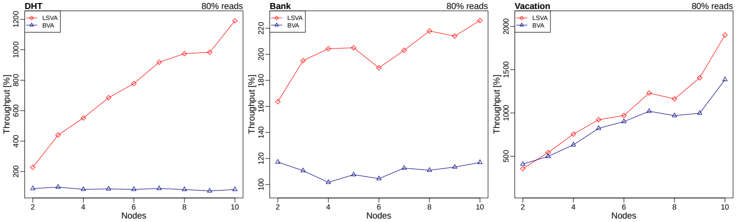
Components of opacity:

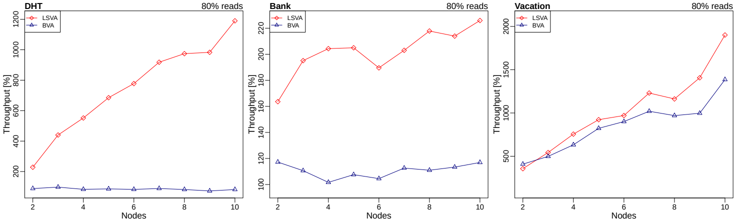
- Serializability
- Real-time order
- Consistency

Components of **last-use opacity**:

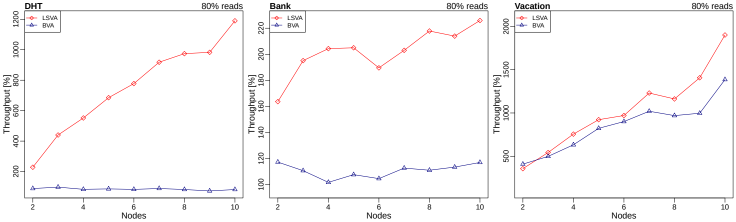
- Serializability
- Real-time order
- **Recoverable last-use consistency**

Siek, Wojciechowski. Relaxing Opacity in Pessimistic Transactional Memory. DISC'14.


DHT
80% reads
LSVA
BVA
Throughput [%]
Nodes
Bank
80% reads
LSVA
BVA
Throughput [%]
Nodes
Vacation
80% reads
LSVA
BVA
Throughput [%]
Nodes



weaken consistency a little → improve efficiency a lot

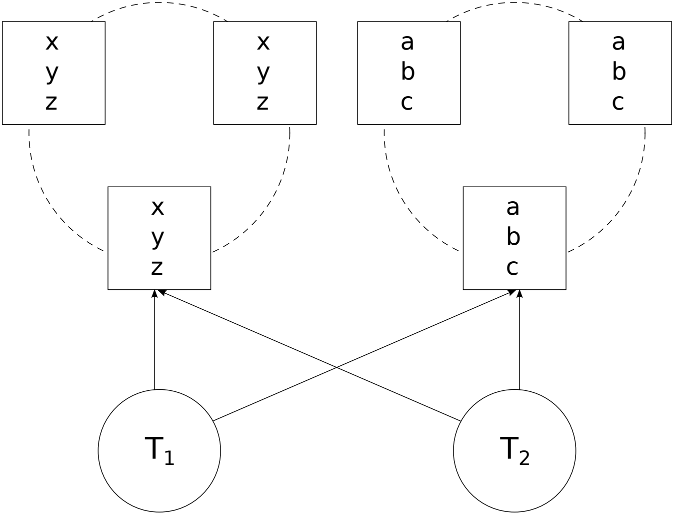


weaken consistency a little → improve efficiency a lot

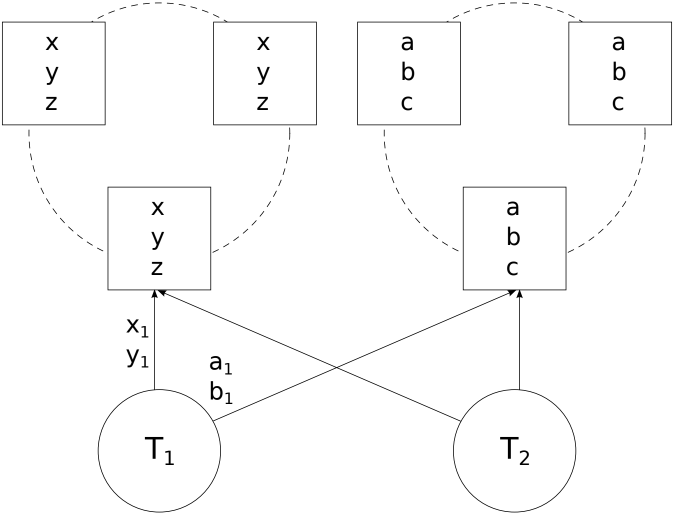
weaken consistency a little more → improve efficiency a lot more?

Wojciechowski, Siek. *Having Your Cake and Eating it Too: Combining Strong and Eventual Consistency.* PaPEC'14.

Eventually Consistent Extension



Eventually Consistent Extension



Transaction Modes

Transaction T_1

$$T_1 \llbracket \ r(x)v, w(x)u \ \rrbracket$$

Transaction Modes

Transaction T_1

$$T_1 \ [\![\ r(x)v, w(x)u \ ]\!]$$

Consistent mode

$$T_1^c \ [\![\ r(x)v, w(x)u \ ]\!]$$

Eventually consistent mode

$$T_1^{ec}[\ r(x)v_{ec}, w(x)u_{ec} \ ]$$

Consistent and EC modes run simultaneously → convergence

Internal consistency of weak transactions

Modification versions

$$\{x = 1, y = 1\} \quad T_1 \; [\![\; r(x)1, w(x)2, r(y)1, w(y)2 \;]\!]$$
$$T_2 \; [\![\qquad \searrow r(x)2, w(x)3 \;]\!] \quad \{x = 3, y = 2\}$$

Internal consistency of weak transactions

Modification versions

$$\{\overset{0}{x}=1,\overset{0}{y}=1\}\quad T_1^{ec}\ [\![\ r(\overset{0}{x})1, w(\overset{1}{x})2, r(\overset{0}{y})1, w(\overset{1}{y})2\ ]\!]$$

$$T_2\ [\![\ \searrow r(\overset{1}{x})2, w(\overset{2}{x})3\ ]\!]\quad \{\overset{2}{x}=3,\overset{1}{y}=2\}$$

Internal consistency of weak transactions

Modification versions

$$\{\overset{0}{x}=1,\overset{0}{y}=1\}\quad T_1^{ec}\ [\![\ r(\overset{0}{x})1, w(\overset{1}{x})2, r(\overset{0}{y})1, w(\overset{1}{y})2\ ]\!]$$

$$T_2\ [\![\ \searrow r(\overset{1}{x})2, w(\overset{2}{x})3\ ]\!]\quad \{\overset{2}{x}=3,\overset{1}{y}=2\}$$

Enforce read isolation

$$T_1\ [\![\ r(\overset{0}{x})1, w(\overset{1}{x})2, r(\overset{0}{y})1, w(\overset{1}{y})2, w(\overset{2}{y})3\ ]\!]$$

$$T_2\ [\![\ \searrow r(\overset{1}{x})2, w(\overset{2}{x})3\ ]\!]$$

$$T_3\ [\![\ \searrow r(\overset{2}{x})3, w(\overset{3}{x})4, r(\overset{2}{y})3, w(\overset{3}{y})4\ ]\!]$$

Internal consistency of weak transactions

Modification versions

$\{\overset{0}{x} = 1, \overset{0}{y} = 1\}$ $\quad T_1^{ec}$ $[\![\ r(\overset{0}{x})1, w(\overset{1}{x})2, r(\overset{0}{y})1, w(\overset{1}{y})2\ ]\!]$

T_2 $[\![\ \searrow r(\overset{1}{x})2, w(\overset{2}{x})3\ ]\!]$ $\quad \{\overset{2}{x} = 3, \overset{1}{y} = 2\}$

Enforce read isolation

T_1 $[\![\ r(\overset{0}{x})1, w(\overset{1}{x})2, r(\overset{0}{y})1, w(\overset{1}{y})2, w(\overset{2}{y})3\ ]\!]$

T_2 $[\![\ \searrow r(\overset{1}{x})2, w(\overset{2}{x})3\ ]\!]$

T_3 $[\![\ \searrow r(\overset{2}{x})3, w(\overset{3}{x})4, r(\overset{2}{y})3, w(\overset{3}{y})4\ ]\!]$

Correct: $\{\overset{1}{x}, \overset{2}{y}\}$, $\{\overset{2}{x}, \overset{2}{y}\}$, $\{\overset{3}{x}, \overset{3}{y}\}$.

Incorrect: $\{\overset{3}{x}, \overset{2}{y}\}$, $\{\overset{*}{x}, \overset{1}{y}\}$.

Consistent snapshot in SVA in practice

Maintaining a consistent snapshot in buffers:

T_i commits: records the latest version of each variable to B^c

T_i release x early:

- records the latest released version of x to B^r
- records variables that were not released early to F

Most recent consistent read snapshot in buffer → EC transactions do not wait to access objects or block other transactions

Maintaining consistent state of non-EC transactions

Handling writes:

$$T_1 \quad [\ r(\overset{0}{x})1, w(\overset{1}{x})2, r(\overset{0}{y})1, w(\overset{1}{y})2, w(\overset{2}{y})3\ ] \qquad \{\overset{1}{x} = 2, \overset{2}{y} = 3\}$$

Maintaining consistent state of non-EC transactions

Handling writes:

$$T_1 \;\; [\; r(\overset{0}{x})1, w(\underline{x})2, r(\overset{0}{y})1, w(\underline{y})2, w(\underline{y})3 \;] \qquad \{\overset{0}{x} = 1, \overset{0}{y} = 1\}\{\underline{x} = 2, \underline{y} = 3\}$$

Buffer $\underline{x}$ only visible to T_1

Maintaining consistent state of non-EC transactions

Handling writes:

$$T_1 \quad [\ r(\overset{0}{x})1, w(\underline{x})2, r(\overset{0}{y})1, w(\underline{y})2, w(\underline{y})3\ ] \qquad \{\overset{0}{x} = 1, \overset{0}{y} = 1\}\{\underline{x} = 2, \underline{y} = 3\}$$

Buffer $\underline{x}$ only visible to T_1

Possibility of “recycling” effort:

If consistency allows it, apply the bufferred writes instead of executing consistent mode from scratch

Eventually Consistent SVA Execution

$$
\begin{array}{llll}
\{\overset{0}{x}=1,\overset{0}{y}=1\} & T_1 & [\![\; r(\overset{0}{x})1, w(\overset{1}{x})2, r(\overset{0}{y})1, w(\overset{1}{y})2 \;]\!] & \\
 & T_2^c & [\![\quad \searrow r(\overset{1}{x})2, w(\overset{2}{x})3 \;]\!] & \{\overset{2}{x}=3,\overset{1}{y}=2\} \\
 & T_2^{ec} & [\; r(\overset{0}{x})1, \color{red}{w(\underline{x})2} \;] \qquad \searrow & \\
 & T_3 & \qquad [\![\; r(\overset{2}{x})3, w(\overset{3}{x})4 \;]\!] &
\end{array}
$$

Conclusions and future work

- eventual consistency extension for pessimistic distributed TM
- minimal extra cost
- eventually consistent transactions read consistent snapshots
- strongly consistent transactions are unaffected
- smaller apparent client latency
- future work:
  - implementation and experimental evaluation
  - safety guarantees of EC transactions

The work is supported by National Science Centre grant *Eventually consistent replication: Algorithms and methods* (30/09/2013–29/09/2016).

?