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

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**

\[
T_1 \left[ \begin{array}{c} r(x)1, w(x)2 \\ r(x)1, w(x)2 \end{array} \right] \\
T_2 \left[ \begin{array}{c} r(x)1, w(x)2 \end{array} \right] \quad \rightarrow \quad T'_2 \left[ \begin{array}{c} r(x)2, w(x)3 \end{array} \right]
\]

**Pessimistic approach**

\[
T_1 \left[ \begin{array}{c} r(x)1, w(x)2 \end{array} \right] \\
T_2 \left[ \begin{array}{c} r(x)2, w(x)3 \end{array} \right]
\]
Pessimistic vs Optimistic TM

Optimistic approach

\[
T_1 \left[ r(x)1, w(x)2 \right] \\
T_2 \left[ r(x)1, w(x)2 \right] \Rightarrow T_2' \left[ r(x)2, w(x)3 \right]
\]

Pessimistic approach

\[
T_1 \left[ r(x)1, w(x)2 \right] \\
T_2 \left[ r(x)2, w(x)3 \right]
\]

- 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.
The joys of early release

Early release on last use

\[ T_1 \left[ r(x)1, w(x)2, r(y)1, w(y)2 \right] \]
\[ T_2 \left[ r(x)2, w(x)3 \right] \]
The joys of early release

Early release on last use

\[ T_1 \left[ r(x)1, w(x)2, r(y)1, w(y)2 \right] \]

\[ T_2 \left[ r(x)2, w(x)3 \right] \]

Performance boost:
The joys of early release

Early release on last use

\[ T_1 \left[ r(x)1, w(x)2, r(y)1, w(y)2 \right] \]
\[ T_2 \left[ r(x)2, w(x)3 \right] \]

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

Manual aborts

The case for manual aborts:

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


Cascading abort

\[ T_1 \uparrow r(x)1, w(x)2, r(y)1, w(y)2, \leftarrow \]
\[ T_2 \uparrow r(x)2, w(x)3 \rightarrow \leftarrow \ldots \]
Manual aborts

The case for manual aborts:

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


Cascading abort

\[
T_1 \left[ r(x)1, w(x)2, r(y)1, w(y)2, \rightarrow \right] \\
T_2 \left[ \rightarrow r(x)2, w(x)3 \rightarrow \rightarrow \ldots \right]
\]

Not opaque and it matters now.
Inconsistent views

Precludes overwriting:

\[
T_i \left[ \begin{array}{c}
w(x)0, \\
w(x)1
\end{array} \right] \\
T_j \left[ \begin{array}{c}r(x)0 \\
\end{array} \right] \Rightarrow T_j' \left[ \begin{array}{c}r(x)1, \\
w(x)2
\end{array} \right]
\]

Allowed inconsistent view:

\[
T_i \left[ \begin{array}{c}
w(x)0, \\
w(x)1
\end{array} \right] \Rightarrow \\
T_j \left[ \begin{array}{c}r(x)0 \\
\end{array} \right] \Rightarrow T_j' \left[ \begin{array}{c}r(x)1, \\
w(x)2
\end{array} \right]
\]

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

Serializability + recoverability

Serializability + ACA

Serializability

Safety properties for TMs with early release

- Serializability
- Elastic Opacity
- Virtual World Consistency
- TMS1 & TMS2

- Recoverability
- Avoiding Cascading Aborts
- Strictness
- Rigorousness

serializability + recoverability

serializability + ACA

serializability

s + r + last-use consistency

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

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?

Eventually Consistent Extension
Eventually Consistent Extension
Transaction Modes

Transaction $T_1$

\[ T_1 \left[ r(x)v, w(x)u \right] \]
Transaction Modes

Transaction $T_1$

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

**Consistent mode**

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

**Eventually consistent mode**

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

Consistent and EC modes run simultaneously $\rightarrow$ convergence
Internal consistency of weak transactions

Modification versions

\[ \{x = 1, y = 1\} \quad T_1 \left[ r(x)1, w(x)2, r(y)1, w(y)2 \right] \]

\[ T_2 \left[ r(x)2, w(x)3 \right] \quad \{x = 3, y = 2\} \]
Internal consistency of weak transactions

Modification versions

\[
\{ x = 1, y = 1 \} \quad T_1^{ec} \left[ r(x)^0, w(x)^1, r(y)^0, w(y)^2 \right] \\
T_2 \left[ r(x)^1, w(x)^2 \right] \quad \{ x = 3, y = 2 \}
\]
Internal consistency of weak transactions

Modification versions
\[
\{x = 1, y = 1\} \quad T_1^{ec} \quad \left[ \begin{array}{c}
 r(x)^1, w(x)^2, r(y)^1, w(y)^2 \\
 r(x)^2, w(x)^3 \\
 r(x)^2, w(x)^3
\end{array} \right] \\
T_2 \quad \left[ \begin{array}{c}
 r(x)^2, w(x)^3 \\
 r(x)^3, w(x)^4, r(y)^3, w(y)^4
\end{array} \right] \quad \{x = 3, y = 2\}
\]

Enforce read isolation
\[
T_1 \quad \left[ \begin{array}{c}
 r(x)^1, w(x)^2, r(y)^1, w(y)^2, w(y)^3 \\
 r(x)^2, w(x)^3 \\
 r(x)^3, w(x)^4, r(y)^3, w(y)^4
\end{array} \right]
\]

\[
T_2 \quad \left[ \begin{array}{c}
 r(x)^2, w(x)^3 \\
 r(x)^3, w(x)^4, r(y)^3, w(y)^4
\end{array} \right]
\]

\[
T_3 \quad \left[ \begin{array}{c}
 r(x)^3, w(x)^4, r(y)^3, w(y)^4
\end{array} \right]
\]
Internal consistency of weak transactions

Modification versions

\[ \{ x = 1, y = 1 \} \quad T_1^{ec} \left[ r(x)^1, w(x)^2, r(y)^1, w(y)^2 \right] \]
\[ T_2 \left[ r(x)^2, w(x)^3 \right] \quad \{ x = 3, y = 2 \} \]

Enforce read isolation

\[ T_1 \left[ r(x)^1, w(x)^2, r(y)^1, w(y)^2, w(y)^3 \right] \]
\[ T_2 \left[ r(x)^2, w(x)^3 \right] \]
\[ T_3 \left[ r(x)^3, w(x)^4, r(y)^3, w(y)^4 \right] \]

Correct: \( \{ x^1, y^2 \}, \{ x^2, y^2 \}, \{ x^3, y^3 \} \).
Incorrect: \( \{ x^3, y^2 \}, \{ x^1, y \} \).
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:

\[ T_1 \ [ \ r(x)^0 1, w(x)^1 2, r(y)^0 1, w(y)^1 2, w(y)^2 3 ] \quad \{ x = 2, y = 3 \} \]
Maintaining consistent state of non-EC transactions

Handling writes:

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

Buffer \( x \) only visible to \( T_1 \)
Maintaining consistent state of non-EC transactions

Handling writes:

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

Buffer \( x \) only visible to \( T_1 \)

Possibility of “recycling” effort:
If consistency allows it, apply the buffered writes instead of executing consistent mode from scratch
Eventually Consistent SVA Execution

\[
\begin{align*}
\{ x = 1, y = 1 \} & \quad T_1 \quad [ r(x)^0, w(x)^1, r(y)^0, w(y)^1 ] \\
T_2^c & \quad [ r(x)^1, w(x)^2, r(y)^0, w(y)^2 ] & \{ x = 3, y = 2 \} \\
T_2^{ec} & \quad [ r(x)^0, w(x)^2 ] \\
T_3 & \quad [ r(x)^2, w(x)^3 ] & [ r(x)^3, w(x)^4 ]
\end{align*}
\]
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).