Towards a Fully-Articulated Pessimistic Distributed Transactional Memory

Brief Announcement

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

23 VII 2013

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

Run simultaneously in case there are no conflicts

In case of conflicts, rollback and retry

\[
\begin{align*}
\{x = 1\} & \quad T_1 \left[ r(x)1, w(x)2 \right] \\
| & \\
T_2 \left[ r(x)1, w(x)2 \right] & \supset\ldots T_2' \left[ r(x)2, w(x)3 \right] \quad \{x = 3\}
\end{align*}
\]
Distributed TM

Distributed Transactions
The Problem of Irrevocable Operations

Irrevocable operations \( T_i[\ldots, \text{ir}, \ldots] \)

- do not operate on shared data
- visible effects on the system
- effect cannot be withdrawn (barring compensation)

Examples: network messages, system calls, I/O operations
The Problem of Irrevocable Operations

Irrevocable operations \( T_i[ ..., \text{ir}, ..., ] \)
- do not operate on shared data
- visible effects on the system
- effect cannot be withdrawn (barring compensation)

Examples: network messages, system calls, I/O operations

\[
\begin{align*}
\{ x = 1 \} & \ T_1 \ [ \ r(x)1, w(x)2 \ ] \\
\mid & \ T_2 \ [ \ r(x)1, \text{ir}, w(x)2 \} \ T'_2 \ [ \ r(x)2, \text{ir}, w(x)3 \} \{ x = 3 \}
\end{align*}
\]
The Problem of Irrevocable Operations

Some workarounds

- forbid irrevocable operations
- buffer irrevocable operations and execute them on commit
- run irrevocable transactions one-at-a-time
- multiple versions of objects
Pessimistic Approach

Defer execution to prevent conflicts

\[
\{ x = 1 \} \quad T_1 \left[ \begin{array}{c}
  r(x)_1, w(x)_2
\end{array} \right]
\]

\[
T_2 \left[ \begin{array}{c}
  r(x)_2, w(x)_3
\end{array} \right] \quad \{ x = 3 \}
\]
Pessimistic Approach

Defer execution to prevent conflicts

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

No rollbacks/aborts, irrevocable operations are not re-run

\[ \{x = 1\} \quad T_1 \left[ r(x)1, w(x)2 \right] \]
\[ \quad | \quad T_2 \quad \rightarrow \quad r(x)2, ir, w(x)3 \quad \{x = 3\} \]
Rollback is still needed for
- expressiveness
- efficiency (i.e. limiting network traffic)
- necessary for fault tolerance
Supremum Versioning Algorithm

Transactions know which objects they use and how many times (suprema)

start:

lock all used objects
assign object's next version to transaction
release locks

access $x$:

wait until $x$ is released by transaction with the previous version of $x$
access $x$
if last use of $x$: release $x$

commit:

release all objects
SVA Characteristics

Early release on last use

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

\( \downarrow T_2 \downarrow [ r(x)2, w(x)3 ] \) \( \{ x = 3, y = 2 \} \)

No aborts, no issues with irrevocable operations
SVA + Rollback

start:
  lock all used objects
  assign object’s next version to transaction
  release locks

access \( x \):
  wait until \( x \) is released by transaction with the previous version of \( x \)
  if first use of \( x \): make copy of \( x \)
  access \( x \)
  if last use of \( x \): release \( x \)

commit:
  wait until transaction with the previous version of \( x \) commits
  if previous transaction rolls back: also roll back
  release all objects

rollback:
  wait until transaction with the previous version of \( x \) commits
  restore all objects from copies and release them
Cascading rollback

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

\[
| \ T_2 \ [ \ r(x)2, w(x)3 \ \Rightarrow \ldots \]
\]
SVA+R Characteristics

Cascading rollback
\[ \{x = 1, y = 1\} \quad T_1 \quad [ \quad r(x)1, w(x)2, r(y)1, w(y)2, \quad \text{abort} \quad ] \]
\[ \quad | \quad T_2 \quad [ \quad r(x)2, w(x)3 \quad ] \quad \Rightarrow \quad \ldots \quad \]

Cascading rollback with irrevocable operations
\[ \{x = 1, y = 1\} \quad T_1 \quad [ \quad r(x)1, w(x)2, r(y)1, w(y)2, \quad \text{abort} \quad ] \]
\[ \quad | \quad T_2 \quad [ \quad r(x)2, ir, w(x)3 \quad ] \quad \Rightarrow \quad \ldots \quad \]
Fixing Cascading Rollback in SVA+R

Cascading rollback conditions in SVA:

- There are two or more transactions that access some object $x$
- The first of those transactions releases $x$ early
- Some younger transaction accesses $x$
- The first transaction aborts
Fixing Cascading Rollback in SVA+R

Cascading rollback conditions in SVA:
- There are two or more transactions that access some object $x$
- The first of those transactions releases $x$ early
- Some younger transaction accesses $x$
- The first transaction aborts

Transactions containing irrevocable operations cannot access objects that were released early (by transactions which may abort)

| $T_1$ [ $r(x)1, w(x)2, r(y)1, w(y)2, \text{ abort}$ ] |
| $T_2$ [ $r(x)2, \text{ir}, w(x)2$ ] |
Properties

- **Opacity (Safety)**
  
  There is some equivalent sequential history that preserves the real-time order of the transactional history and every transaction in the transactional history is legal in the sequential history.

  - Real-time order from version order
  - Legality from exclusive access to committed objects
Properties

- **Opacity** *(Safety)*
  There is some equivalent sequential history that preserves the real-time order of the transactional history and every transaction in the transactional history is legal in the sequential history.
  - Real-time order from version order
  - Legality from exclusive access to committed objects
  - ... or uncommitted objects equivalent to committed objects
Properties

- **Opacity** *(Safety)*
  There is some equivalent sequential history that preserves the real-time order of the transactional history and every transaction in the transactional history is legal in the sequential history.
  - Real-time order from version order
  - Legality from exclusive access to committed objects
  - ... or uncommitted objects equivalent to committed objects

- **Strong Progressiveness** *(Liveness)*
  When two transactions conflict on some object, one of them will not be forced to abort.
  - Impossibile for all transactions to roll back due to cascading rollback conditions and version order
Properties

- **Opacity** (Safety)
  There is some equivalent sequential history that preserves the real-time order of the transactional history and every transaction in the transactional history is legal in the sequential history.
  - Real-time order from version order
  - Legality from exclusive access to committed objects
  - ... or uncommitted objects equivalent to committed objects

- **Strong Progressiveness** (Liveness)
  When two transactions conflict on some object, one of them will not be forced to abort.
  - Impossibile for all transactions to roll back due to cascading rollback conditions and version order

- **Deadlock-freedom**

- Probably not **Livelock-freedom**

- Probably susceptible to **Parasitic Transactions**
Conclusions

- Transactional Memory for Distributed Systems
Conclusions

- Transactional Memory for Distributed Systems
- Irrevocable operations and Pessimistic TMs
Conclusions

- Transactional Memory for Distributed Systems
- Irrevocable operations and Pessimistic TMs
- Incorporating Rollback into Pessimistic Distributed TM
Conclusions

- Transactional Memory for Distributed Systems
- Irrevocable operations and Pessimistic TMs
- Incorporating Rollback into Pessimistic Distributed TM
- Safety and Progress
Related Papers:

