OSARE
Opportunistic Speculation in Actively Replicated Transactional Systems

Roberto Palmieri, Francesco Quaglia, Paolo Romano

Roberto Palmieri – Workshop on Transactional Memory (WTM 2012)
Bern, Switzerland, April 10, 2012
Why Active Replication

• Each replica keeps all data and executes the same transactions in the same order

• PRO (+)
  – Full failure masking
  – No coordination for processing read-only transactions
  – No network synchronization while processing
  – Prone to target performance issues

• CONS (-)
  – Agreement on common execution order
Literature solution for actively replicated transactional systems
Optimistic Approach (OPT)

- Based on Optimistic Atomic Broadcast as GCS
- It processes in optimistic manner:
  - At most one conflicting transaction
  - Any non-conflicting transactions

```
if(opt-delivery order == to-delivery order)
  Commit(m)
if(opt-delivery order <> to-delivery order)
  Abort & Restart(m)
if(m is non-conflicting transaction)
  Commit(m)
```

Roberto Palmieri – Workshop on Transactional Memory (WTM 2012) - Bern, Switzerland, April 10, 2012
Critiques to OPT

• A-priori knowledge on transaction read/write sets
• Limited overlapping in case of fine-grain transactions

Coordination delay

Local Transaction Execution Time
Unbalancing Ratio

Traditional Scenarios
≈ 2 msec
≈ 1/10 msec

Modern (STM) Scenarios
≈ 2 msec
≈ 10/100 μsec

Resources underutilization

Processing
Coordination phase
Target: Maximize the overlap

- Coordination delay Vs Local transaction processing
- Local transaction processing Vs Local transaction processing

Opt-delivery (m1) → Coordination phase m1 → Coordination phase m2 → Coordination phase m3 → to-delivery (m3)

Multiple or same Serialization Orders

Roberto Palmieri – Workshop on Transactional Memory (WTM 2012) - Bern, Switzerland, April 10, 2012
How: Speculative Processing

- Basic ideas:
  - Activate all transactions as soon as they are optimistic delivered
  - Explore (in depth and/or in breadth) multiple serialization orders

\[
\begin{align*}
T_A: & \quad \text{exec } T_A \\
& \quad T_{commit} \rightarrow T_A \\
& \quad T_B \rightarrow T_A \\
T_B: & \quad \text{exec } T_B \\
& \quad T_{commit} \rightarrow T_B \\
& \quad T_A \rightarrow T_B \\
T'_A: & \quad \text{exec } T'_A \\
& \quad T_B \rightarrow T'_A \\
& \quad T_A \rightarrow T'_A \\
T'_B: & \quad \text{exec } T'_B \\
& \quad T_A \rightarrow T'_B \\
& \quad T_B \rightarrow T'_B \\
\end{align*}
\]

Commit sequence:

- \( T_A \): commit(T_A)
- \( T_B \): commit(T_B)

Abort sequence:

- \( T_A \): abort(T'_A)
- \( T_B \): abort(T'_B)
Changing the Perspective of Speculation

OSARE
Speculating According to an Opportunistic Paradigm
Opportunistic Speculation in Active Replication (OSARE) [SRDS2011]

• Simple data structures -> Polynomial Overhead
• Targets:
  – Follow the optimistic delivery order
  – Explore multiple serialization orders only when concurrency increases
• How:
  – Exploit the so called *snapshot miss* event to expand the coverage of alternative SOs
OSARE: Key Ideas

• Speculation degree reflects network concurrency
• The probability to invert the optimistic delivery order wrt the final one increases with the arrival rate of network messages
Snapshot Miss Event

• A snapshot miss event occurs when:
  – $T_i^s$ writes data item X for which $T_j^t$ has already issued a read operation

$T_j^t$ missed the snapshot of $T_i^s$

A new instance of $T_j^t$ -> $T_j^{(t+1)}$ is spawned
The snapshot miss events + the optimistic delivery order define the actual transaction serialization order that is stored within each transaction.
Algorithm sketch: Read

- Read on data item X by $T_i^j$
- Select version V in according to the serialization order of $T_i^j$
- Update read-from set of $T_i^j$
Algorithm sketch: Complete

- $T_i^j$ completes its execution
- Make available written data items to others speculative transactions
- For each transaction $T_m^n$ that missed any data item written by $T_i^j$
  - Spawn new speculative transaction $T_m^{(n+1)}$ and force $T_m^{(n+1)}$ to read the missed data item
- wave on $T_m^n$
Algorithm sketch: Wave

When a transaction $T_{m^n}$ misses a snapshot

For each transaction $T_{i^f}$ belonging to the read-form set of $T_{m^n}$

Spawn new speculative transaction $T_{i^{(f+1)}}$

Wave (recursively) propagates the snapshot miss to all the transactions that read from $T_{m^n}$
OSARE Trajectory

- Largest set of explored speculative serialization orders: $O(2^n)$
Trace-based Simulator

- A discrete event simulator
- Realistic data access patterns
- Realistic optimistic atomic broadcast traces (coming from Appia Toolkit)
- Realistic probability to mismatch between optimistic and final deliveries
Simulation results

- **Stamp Benchmarks**
- **OSARE Vs:**
  - AGGRO (aggressive at-most-one serialization order)
  - OPT (at-most-one conflicting transaction)
  - SM (No optimistic processing)
Thank you for the attention