Archie: A Speculative Replicated Transactional System

Sachin Hirve, Roberto Palmieri, Binoy Ravindran
Systems Software Research Group
Virginia Tech
What do we want for our transactional application?

Strong consistency
High availability
High performance
Great scalability in small/med size
Fault tolerance
Low latency
Programmability
Commonly used building blocks

Full Replication

Total Order (e.g., Paxos)
How does it work when combined?

Client Request (tx1) -> msg (tx1) -> Ordering phase -> execution of (tx1) -> Client Reply

tx1 -> R1 -> R2 -> R3 -> R4 -> tx2
...and when the load increases?
Goal:
Boost performance in the common case

Challenge:
Transaction response time = total order latency
Anticipate the work

- Exploitation of an early notification triggered during the ordering phase (called *optimistic-delivery*)
Reliable Optimistic Delivery

- Avoid any mismatch between the optimistic delivery order and total order
- Maximize the time between the optimistic delivery and the establishment of the total order
MIMOX

• Multi Paxos + Reliable Optimistic Delivery
  – Exploit the leader’s batching time for maximizing the overlapping time
  – Exploit the leader’s knowledge for making the optimistic delivery reliable
    • Tag messages with leader’s expected order
MIMOX: Batch creation
MIMOX: Batch broadcast

Batching window

Client Requests

L

R1

R2

123

123

123

123

1

2

3

Optimistic Delivery

Optimistic Delivery
MIMOX: Performance

Graph 1: Performance of MIMOX with different node counts and message sizes.

- **Nodes**: 3, 5, 7, 9, 11, 13, 15, 17, 19
- **Message Sizes**: 10 bytes, 20 bytes, 50 bytes
- **Y-Axis**: 1000x msgs per sec
- **X-Axis**: Nodes

Graph 2: Bar chart showing the milliseconds (msec) for different message sizes.

- **Nodes**: 3, 5, 7, 9, 11, 13, 15, 17, 19
- **Message Sizes**: 10 bytes, 20 bytes, 50 bytes
Transaction Processing

- Exploit parallelism
- Commit in order
- Minimal overhead for conflict detection and resolution
Key points of ParSpec

• Reduce problem’s complexity by activating MaxSpec transactions at-a-time
  – meta-data’s size is fixed
  – set of possible conflicting transactions is limited

• Parallel execution but in-order speculative commit (x-commit)
ParSpec: details

<table>
<thead>
<tr>
<th>Optimistic Order:</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
<th>T7</th>
<th>T8</th>
<th>MaxSpec = 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>T2 (X): Begin</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T3 (X): Begin</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X: Read bit-array</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Abort bit-array</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read(X)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Wait</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Read(X)</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Write(X)</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>X-commit</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**OR bitwise**
Is this enough for achieving the goal?

• Unfortunately not…

![Diagram showing Ordering phase, Tx1, Validation ReadSet, Commit WriteSet, and a symbol \( \Delta \) with \( \approx \Delta \) relation.]
Solution for Write Set

- Speculative versions are associated with an tentative commit timestamp, which is the expected commit timestamp in case of no mismatch between optimistic and total order
Solution for Read Set

• If the leader is not either crashed or suspected (it is stable), then there is no chance of mismatch between optimistic and total order

• ParSpec commits according to optimistic order and make them visible during the total order
We made it!

Ordering phase

TS++

Tx1
Evaluation

• Testbed – PRObE cluster (19 nodes)
  – AMD Opteron 6272, 64-core, 2.1 GHz CPU
  – 128 GB RAM and 40 Gbps Ethernet

• Benchmarks
  – Bank, TPC-C and Vacation

• Competitors
  – PaxosSTM: DUR-based approach; it suffers from remote aborts
  – SM-DER: Post final delivery single-thread execution
  – HiperTM: SM-DER based single thread execution
  – Archie-FD: Archie but post final delivery parallel processing
Evaluation: TPC-C

Throughput
19 warehouses

Throughput
100 warehouses

Number of warehouses

<table>
<thead>
<tr>
<th>Number of warehouses</th>
<th>3 nodes</th>
<th>11 nodes</th>
<th>19 nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>90</td>
<td>70</td>
</tr>
<tr>
<td>19</td>
<td>30</td>
<td>80</td>
<td>60</td>
</tr>
<tr>
<td>100</td>
<td>40</td>
<td>90</td>
<td>70</td>
</tr>
</tbody>
</table>

Systems Software Research Group
Virginia Tech
Invent the Future
Evaluation: Distributed Vacation

Transactions per sec

Number of relations

SM-DER
HiperTM
PaxosSTM
ARCHIE-FD
ARCHIE

Contention level
Thanks!

Questions?

Research project’s web-site: www.hyflow.org