Designing Distributed Systems Using Approximate Synchrony in Data Center Networks

Dan R. K. Ports, Jialin Li, Vincent Liu, Naveen Kr. Sharma, Arvind Krishnamurthy
University of Washington

Presented by Xiao Zhu
Background

● Replication and Consensus algorithms
  ○ State Machine Replication & Paxos
  ○ Provide high available and consistent services
  ○ Typically designed independently from the underlying network

● Data Centers
  ○ Host distributed applications
  ○ Have desirable properties to be distinguished from the Internet
    ■ Centralized control
    ■ A structured network
    ■ Switch support for QoS
Co-design Distributed systems with Data Center Networks

- Approximately synchrony instead of worst-case assumptions
  - New network-level primitive to provide Mostly-Ordered Multicast
  - New replication protocol: Speculative Paxos

Deploy new network-level primitives that benefit higher layers

Build systems that rely on stronger guarantees available in the network
Mostly-Ordered Multicast (MOM)

- Existing multicast are not ordered
  - Different path lengths
  - Different congestion for each path
- Comparison with totally-ordered multicast (TOM)
  - Relaxed version of TOM
  - Ordering violation handling
Engineer a Data Center Network for MOM

- Topology-aware multicast
  - Ensure all multicast messages traverse the same number of links
- High-priority multicast
  - Assign high QoS priorities to multicasts
- In-network serialization
  - Route all packets through a single root switch
Evaluation of MOM

- Small-scale experiment:

- Large-scale simulation:
Speculative Paxos

● A new state machine replication protocol
  ○ Relies on MOMs to be ordered in the common case
  ○ Each replica speculatively executes requests based on the order, before agreement is reached
  ○ Remains correct even with reorderings: safety + liveness under usual conditions

● Handling Ordering Violations
  ○ Does not rely on MOM for correctness
  ○ Same correctness assumption as Paxos & Viewstamped Replication:
    ■ Requires 2 f + 1 replicas to provide safety
  ○ Synchronization + Reconciliation
SpecPaxos Replica State

- Each replica maintains `<status, log, view number>
  - **Status** indicates whether it can process new operations
    - NORMAL, RECONCILIATION, RECOVERY, RECONFIGURATION
  - **Log** consists of entries tagged with **sequence number** and **state**, also associated with a **summary hash**
    - **State** can be either **COMMITTED** or **SPECULATIVE**
    - summary\(_n\) = \(H(\text{summary}_{n-1} \mid \text{operation}_n)\)
  - The system moves through a series of views
    - Round-robin leader selection based on view number
Paxos v.s. SpecPaxos

Paxos latency: 4 message delays

SpecPaxos latency: 2 message delays
No bottleneck replica: each processes only 2 messages
SpecPaxos Synchronization

- Leader initiates synchronization by sending a \(<\text{SYNC}, v, s>\) message to the other replicas
  - Periodically: every \(y\) milliseconds or every \(k\) requests
  - \(v\): current view number
  - \(s\): the highest sequence number in its log

- Replicas respond to \text{SYNC} message by sending \(<\text{SYNC-REPLY}, v, s, h(s)>\) to leader
  - Message will be ignored if \(v\) is not the current view number, or the replica is not in the \text{NORMAL} state

- Leader checks the \text{SYNC-REPLY} message and sends commit message \(<\text{COMMIT}, v, s, h(s)>\) to other replicas
  - When the leader received \(f+[f/2]+1\) \text{SYNC-REPLY} messages for a sequence number \(s\)
  - Initiates a reconciliation when:
    - The received \text{SYNC-REPLY} messages don’t have the same hash
    - Or a replica receives a \text{COMMIT} message with a different hash than its current log entry
SpecPaxos Reconciliation

- Replicas execute requests speculatively
  - Might have to roll back operations
- Reconciliation repairs divergent state when
  - Messages are dropped or reordered by the network
  - Replica fails
- Follows the same structure as view changes in Viewstamped Replication, but with a more complex log merging procedure
  - Replica increments view number and sets status to RECONCILIATION
  - Sends `<START-RECONCILIATION, v>` message to the other replicas
  - Sends `<RECONCILE, v, v, log>` message to leader when receiving START-RECONCILIATION messages for view v from f other replicas
  - Leader merges logs upon receiving RECONCILE messages from f other replicas
Implementation

- SpecPaxos protocol as a library for clients and replicas
  - 10,000 lines of C++ code
  - Also supports leader-based Paxos (w/ or w/o batching) and Fast Paxos

**Client interface**

- `invoke(operation) → result`

**Replica interface**

- `speculativelyExecute(seqno, operation) → result`
- `rollback(from-seqno, to-seqno, list<operations>)`
- `commit(seqno)`
Evaluation

● Setup
  ○ 12-switch fat-tree testbed
  ○ 1Gbps/10Gbps ethernet
  ○ 3 replicas (2.27 GHz Xeon L5640)

● For reordering sensitivity test
  ○ Three nodes with Xeon L5640 processors connected via a single 1 Gbps switch
  ○ 20 concurrent closed-loop clients
Latency/Throughput Comparison

![Latency/Throughput Comparison Graph](image-url)
Reordering Sensitivity

![Graph showing throughput (operations/sec) vs. simulated packet reordering rate for different algorithms. The graph compares SpecPaxos, Paxos + batching, Fast Paxos, and Paxos. Each algorithm has a distinct line, indicating their performance across varying reordering rates.]
Applications

- Timestamp Server
- Lock Manager
- Transactional Key-Value Store

<table>
<thead>
<tr>
<th>Application</th>
<th>Total LoC</th>
<th>Rollback LoC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timestamp Server</td>
<td>154</td>
<td>10</td>
</tr>
<tr>
<td>Lock Manager</td>
<td>606</td>
<td>75</td>
</tr>
<tr>
<td>Key-value Store</td>
<td>2011</td>
<td>248</td>
</tr>
</tbody>
</table>

![Bar chart showing max throughput for different Paxos variants](chart.png)
Related Work

- Weak Synchrony
  - Ethernet LAN
  - Optimistic Atomic Broadcast

- Paxos variants

<table>
<thead>
<tr>
<th></th>
<th>Latency (Msg Delays)</th>
<th>Message Complexity</th>
<th>Messages at Bottleneck Replica</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paxos</td>
<td>4</td>
<td>$2n$</td>
<td>$2n$</td>
</tr>
<tr>
<td>Paxos + batching</td>
<td>4+</td>
<td>$2 + \frac{2n}{b}$</td>
<td>$2 + \frac{2n}{b}$</td>
</tr>
<tr>
<td>Fast Paxos</td>
<td>3</td>
<td>$2n$</td>
<td>$2n$</td>
</tr>
<tr>
<td>Speculative Paxos</td>
<td>2</td>
<td>$2n + \frac{2n}{s}$</td>
<td>$2 + \frac{2n}{s}$</td>
</tr>
</tbody>
</table>

- Speculation
  - Speculative Byzantine fault tolerant replication (e.g. Zyzzyva, Eve, etc)
  - Speculation on the client side
Conclusion

- Co-design distributed systems with the data center network
  - New network-level multicast mechanisms to provide MOMs in DCNs
  - Speculative Paxos as an efficient replication protocol
- Higher throughput and lower latency than standard protocols in data center environments
Concerns

- What if the switch fails
- Commodity switches may not support QoS
- Client may become the bottleneck
Thank you!

Q & A