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 CSE
Today’s most popular applications are **distributed systems** in the **data center**
Today’s most popular applications are \textit{distributed systems} in the \textit{data center}

Modern data center:
\begin{itemize}
  \item \~50,000 commodity servers
  \item constant server failures
\end{itemize}
How do we program the data center?

Use distributed algorithms to tolerate failures, inconsistencies.

Example: Paxos state machine replication.
Distributed systems and networks are typically designed independently.
Distributed systems and networks are typically designed independently.

Asynchronous network (Internet):
- packets may be arbitrarily
  - dropped
  - delayed
  - reordered
Distributed systems and networks are typically designed independently.
Distributed systems and networks are typically designed independently.

Data center networks are different!
Data Center Networks Are Different

Data center networks are more **predictable**
- known topology, routes, predictable latencies

Data center networks are more **reliable**

Data center networks are **extensible**
- single administrative domain makes changes possible
- software-defined networking exposes sophisticated line-rate processing capability
Data Center Networks Are Different

Data center networks are more **predictable**
- known topology, routes, predictable latencies

We should co-design distributed systems and data center networks!

Data center networks are **extensible**
- single administrative domain makes changes possible
- software-defined networking exposes sophisticated line-rate processing capability
Co-Designing Networks and Distributed Systems

- Design the *data center network* to support *distributed applications*
- Design *distributed applications* around the properties of the *data center network*
This Talk

A concrete instantiation:

improving replication performance using

*Speculative Paxos* and *Mostly-Ordered Multicast*
This Talk

A concrete instantiation:

improving replication performance using *Speculative Paxos* and *Mostly-Ordered Multicast*

- new replication protocol
- new network primitive
This Talk

A concrete instantiation:

improving replication performance using

Speculative Paxos and Mostly-Ordered Multicast

3x throughput and 40% lower latency than conventional approach
Outline

1. Co-designing Distributed Systems and Data Center Networks

2. Background: State Machine Replication & Paxos

3. Mostly-Ordered Multicast and Speculative Paxos

4. Evaluation
State Machine Replication

Used to tolerate failures in datacenter applications

- keep critical management services online (e.g., Google's Chubby, Zookeeper)
- persistent storage in distributed databases (e.g., Spanner, H-Store)

Strongly consistent (linearizable) replication, i.e., all replicas execute same operations in same order

... even when up to half replicas fail
... even when messages are lost
Example: Paxos

Client
Leader Replica
Replica
Replica
Example: Paxos
Example: Paxos

Client

Leader Replica

Replica

Replica

request prepare
Example: Paxos

Client

Leader Replica

Replica

Replica

request prepare prepareok
Example: Paxos

- Client
- Leader
- Replica
- Replica
- Replica

request → prepare → prepareok → exec
Example: Paxos

Client

Leader Replica

Replica

Replica

request | prepare | prepareok | reply

exec | commit
Example: Paxos

Latency: 4 message delays
Example: Paxos

Latency: 4 message delays

Throughput: bottleneck replica processes 2n msgs
Outline

1. Co-designing Distributed Systems and Data Center Networks
2. Background: State Machine Replication & Paxos
3. Mostly-Ordered Multicast and Speculative Paxos
4. Evaluation
Improving Paxos Performance

Paxos requires a leader replica to order requests

Can we use the network instead?
Improving Paxos Performance

Paxos requires a leader replica to order requests

Can we use the network instead?

Engineer the network to provide Mostly-Ordered Multicast (MOM)
- best-effort ordering of multicasts

New replication protocol: Speculative Paxos
- commits most operations in a single round trip
Mostly-Ordered Multicast

Concurrent messages are ordered:
If any node receives message A then B, then all other receivers process them *in the same order*

- best effort — not guaranteed

Practical to implement

- can be violated in event of network failure
- but not satisfied by existing multicast protocols!
Mostly-Ordered Multicast
Mostly-Ordered Multicast

- Different path lengths, congestion cause reordering
Mostly-Ordered Multicast

- Different path lengths, congestion cause reordering
Mostly-Ordered Multicast

- Different path lengths, congestion cause reordering
Mostly-Ordered Multicast

- Different path lengths, congestion cause reordering
Mostly-Ordered Multicast

- Different path lengths, congestion cause reordering

- **MOM approach**: Route multicast messages to a root switch equidistant from receivers
Mostly-Ordered Multicast

- Different path lengths, congestion cause reordering

- **MOM approach**: Route multicast messages to a root switch equidistant from receivers
MOM Design Options

1. Topology-Aware Multicast
   route packets to a randomly-chosen root switch

less network support

better ordering
MOM Design Options

1. Topology-Aware Multicast
   route packets to a randomly-chosen root switch

2. High-Priority Multicast
   use higher QoS priority to avoid link congestion

less network support

better ordering
**MOM Design Options**

1. **Topology-Aware Multicast**
   route packets to a randomly-chosen root switch

2. **High-Priority Multicast**
   use higher QoS priority to avoid link congestion

3. **Network Serialization**
   route packets through a *single* root switch
Speculative Paxos

New state machine replication protocol
Relies on MOM to order requests
in the normal case
But not required:
• remains correct even with reorderings:
  safety + liveness under usual conditions
Speculative Paxos
Speculative Paxos

- Client
- Replica
- Replica
- Replica

request
Speculative Paxos

replicas immediately speculatively execute request & reply!

Client

Replica

Replica

Replica

request

spec-reply(result, hash)

spec-exec

spec-exec

spec-exec
Speculative Paxos

replicas immediately speculatively execute request & reply!
client checks for matching responses from 3/4 superquorum
Speculative Paxos

replicas immediately speculatively execute request & reply!
client checks for matching responses from 3/4 superquorum

latency: 2 message delays (vs 4)
Speculative Paxos

replicas immediately speculatively execute request & reply!
client checks for matching responses from 3/4 superquorum

- replicas each processes only 2 msgs
- no bottleneck replica

latency: 2 message delays (vs 4)
Speculative Execution

Replicas execute requests speculatively
- might have to roll back operations

Clients know their requests succeeded
- they check for matching hashes in replies
- means clients don’t need to speculate

Similar to Zyzzyva [SOSP’07]
Handling Ordering Violations

What if replicas don’t execute requests in the same order?

Replicas periodically run *synchronization* protocol

If divergence detected: *reconciliation*

- replicas pause execution, select leader, send logs
- leader decides ordering for operations and notifies replicas
- replicas rollback and re-execute requests in proper order
Handling Ordering Violations

What if replicas don’t execute requests in the same order?

Replicas periodically run *synchronization* protocol

If divergence detected: *reconciliation*

- replicas pause execution, select leader, send logs
- leader decides ordering for operations and notifies replicas
- replicas rollback and re-execute requests in proper order

Note: 3/4 superquorum requirement ensures new leader can always be sure which requests succeeded even if 1/2 fail.  [cf. Fast Paxos]
Outline

1. Co-designing Distributed Systems and Data Center Networks
2. Background: State Machine Replication & Paxos
3. Mostly-Ordered Multicast and Speculative Paxos
4. Evaluation
Evaluation Setup

12-switch fat tree testbed
1 Gb / 10 Gb ethernet
3 replicas (2.27 GHz Xeon L5640)

MOM scalability experiments:
2560-host simulated fat tree data center network
background traffic from Microsoft data center measurements
SpecPaxos Improves Latency and Throughput
(emulated datacenter network with MOMs)

better ↑

latexy
(us)

throughput (ops / second)  better →
SpecPaxos Improves Latency and Throughput
(emulated datacenter network with MOMs)

- Latency (μs):
  - Paxos: 600 μs
  - SpecPaxos: 300 μs

- Throughput (ops/second):
  - Paxos: 25,000 ops/second
  - SpecPaxos: 100,000 ops/second

Better latency and throughput with SpecPaxos.
SpecPaxos Improves Latency and Throughput
(emulated datacenter network with MOMs)

3x throughput and 40% lower latency than Paxos
SpecPaxos Improves Latency and Throughput
(emulated datacenter network with MOMs)

better ↑

latency (us)

throughput (ops / second)

Paxos
Fast Paxos
Paxos + batching
SpecPaxos
Paxos + batching

better →
SpecPaxos Improves Latency and Throughput
(Emulated datacenter network with MOMs)

Better latency than Fast Paxos and same throughput as batching!
MOMs Provide Necessary Support

Throughput

Speculative Paxos

Paxos

Simulated packet reordering rate
<table>
<thead>
<tr>
<th></th>
<th>Testbed (12 switches)</th>
<th>Simulation (119 switches, 2560 hosts)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Regular Multicast</strong></td>
<td>1-10%</td>
<td>1-2%</td>
</tr>
<tr>
<td><strong>Topology-Aware MOM</strong></td>
<td>0.001%-0.05%</td>
<td>0.01%-0.1%</td>
</tr>
<tr>
<td><strong>Network Serialization</strong></td>
<td>~0%</td>
<td>~0%</td>
</tr>
</tbody>
</table>
Application Performance

Transactional key-value store (2PC + OCC)
Synthetic workload based on Retwis Twitter clone

< 250 LOC required to implement rollback

Measured transactions/sec that meet 10 ms SLO

![Bar chart showing max throughput for different Paxos variants]

<table>
<thead>
<tr>
<th>Variant</th>
<th>Max Throughput (Transactions/second)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paxos</td>
<td>1800</td>
</tr>
<tr>
<td>Paxos+Batching</td>
<td>4500</td>
</tr>
<tr>
<td>Fast Paxos</td>
<td>3000</td>
</tr>
<tr>
<td>SpecPaxos</td>
<td>5700</td>
</tr>
</tbody>
</table>
Summary

New approach to building distributed systems based on co-designing with the data center network

Dramatic performance improvement for replication by combining

- MOM network primitive for best-effort ordering
- Speculative Paxos: efficient replication protocol

This is only the first step for co-designing distributed systems and data center networks!