Tolerating Latency in Replicated State Machines through Client Speculation

April 22, 2009

Benjamin Wester\textsuperscript{1}, James Cowling\textsuperscript{2}, Edmund B. Nightingale\textsuperscript{3}, Peter M. Chen\textsuperscript{1}, Jason Flinn\textsuperscript{1}, Barbara Liskov\textsuperscript{2}

\textit{University of Michigan}\textsuperscript{1}, \textit{MIT CSAIL}\textsuperscript{2}, \textit{Microsoft Research}\textsuperscript{3}
Replicated State Machines (RSM)

- Agree on request
- All non-faulty replies are identical
RSMs have high latency

1. Need many replies
2. Agreement
3. Geographic Distribution
Hide the Latency

• Use speculative execution inside RSM
• Speculate before consensus is reached
  – Without faults, any reply predicts consensus value
  – Let client continue after receiving one reply
Overview

- Introduction
- Improving RSMs with speculation
- Application to PBFT
- Performance
- Conclusion
Speculative Execution in RSM

- Continue processing while waiting
Critical path: first reply

- Completion latency less relevant
- First reply latency sets critical path
  - Speed
  - Accuracy
- Other desirable properties
  - Throughput
  - Stability under contention
  - Smaller number of replicas
Requests while speculative

1. Hold request
   - Bad performance
2. Distributed commit/rollback
   - State tracking complex

```python
while not check_lottery():
    submit_tps()
    buy_corvette()
```

Predict `win? = yes`

What do we do with this?
Resolve speculations on the replicas

- Explicitly encode dependencies as **predicates**
- No special request handling needed
- Replicas need to log past replies
- Local decision at replicas matches client

```python
while !check_lottery():
    submit_tps()
    buy_corvette()
```

Predict \( \text{win?} = \text{yes} \)

```python
if \( \text{win?} = \text{yes} \):
    \text{buy}
```
Overview

- Introduction
- Improving RSMs with speculation
- Application to PBFT
- Performance
- Conclusion
Practical BFT-CS

[Castro and Liskov 1999]
Additional Details

- Tentative execution
  - PBFT/PBFT-CS complete in 4 phases
- Read-only optimization
  - Accurate answer from backup replica
- Failure threshold
  - Bound worst-case failure
- Correctness
Overview

• Introduction
• Improving RSMs with speculation
• Application to PBFT
• **Performance**
• Conclusion
Benchmarks

• Shared counter
  – Simple checkpoint
  – No computation

• NFS: Apache httpd build
  – Complex checkpoint
  – Significant computation
1. Primary-local
2. Primary-remote
3. Uniform

Topology

2.5 or 15 ms
Base case: no replication

1. Primary-local
2. Primary-remote
3. Uniform
Shared Counter

Primary-local topology

Run Time (sec) vs. Network Delay (ms)

- PBFT
- PBFT-CS
- No replication
Primary-local topology

Run Time (sec)

Network Delay (ms)

PBFT
PBFT-CS
No replication
Zyzzyva

[Kotla et al. 07]
Shared Counter

Uniform & Primary-remote topology

- PBFT
- PBFT-CS
- No replication

Run Time (sec) vs Network Delay (ms)
Shared Counter

Uniform & Primary-remote topology

Run Time (sec)

Network Delay (ms)

PBFT
PBFT-CS
No replication
Zyzzyva

NSDI'09
Benjamin Wester
University of Michigan CSE
NFS: Apache build

Primary-local topology

Run Time (min) vs Network Delay (ms)

- PBFT
- PBFT-CS
- No replication

NSDI'09
Benjamin Wester
University of Michigan CSE
NFS: Apache build

Uniform topology

- PBFT
- PBFT-CS
- No replication

Run Time (min) vs. Network Delay (ms)
NFS: Apache build

Primary-remote topology

Run Time (min)

Network Delay (ms)

PBFT
PBFT-CS
No replication
NFS: With Failure

Primary-local topology

Run Time (min)

Network Delay (ms)

PBFT
PBFT-CS
No replication
PBFT-CS (1% fail)
Throughput (Shared Counter)

LAN topology

Number of Clients

KOps/sec
Conclusion

• Integrate client speculation within RSMs
• Predicated requests: performance without complexity
• Clients less sensitive to latency between replicas
• 5x speedup over non-speculative protocol

Makes WAN deployments more practical