BFT Protocols under Fire

Atul Singh  
MPI-SWS/Rice

Tathagata Das  
IIT Kharagpur

Petros Maniatis  
Intel Research Berkeley

Peter Druschel  
MPI-SWS

Timothy Roscoe  
ETH Zurich
Byzantine Faults Are Real

• Byzantine: No assumption about the behavior of a faulty component
  • Software bugs, viruses, hardware failures

• Increasing evidence of Byzantine faults
  • E.g., in recent study of database bug reports, majority of errors lead to incorrect results

• Byzantine fault tolerance (BFT) based on state machine replication
  • Abstraction of a correct service even under faults
Byzantine Faults Are Real

- Byzantine: No assumption about the behavior of a faulty component
  - Software bugs, viruses, hardware failures

- Increasing evidence of Byzantine faults
  - E.g., in recent study of database bug reports, majority of errors lead to incorrect results

- Byzantine fault tolerance (BFT) based on state machine replication
  - Abstraction of a correct service even under faults
Flurry of BFT Publications

- Great strides in improving performance
Flurry of BFT Publications

- Great strides in improving performance
Flurry of BFT Publications

- Great strides in improving performance
- But
  - No “apples-to-apples” comparison
  - Performance evaluation tends to focus on benign conditions
- Which protocol is best under a given set of conditions?

PBFT [OSDI’99]
Q/U [SOSP’05]
BAR [SOSP’05]
HQ [OSDI’06]
BFT2F [NSDI’07]
Zyzzyva [SOSP’07]
Shepherd [SOSP’07]
A2M [SOSP’07]
Towards “apples-to-apples” comparison

- Implementation based
  - Use different languages, crypto, transport etc.
  - Compare them under different conditions, workloads

- Analytical performance modeling based
  - Account for crypto cost, network cost
  - Possible only for relatively simple cases
  - But hard to capture dynamic behavior -- multiple phases, concurrency, retransmissions, complex reconfigurations, etc.
BFTSim: The Approach

- Specify protocol in OverLog, a declarative language [P2]
- Specify cost of message processing and crypto operations
- Specify network topology [ns-2]
- Simulate execution
BFTSim: The Approach

- Specify protocol in OverLog, a declarative language [P2]
- Specify cost of message processing and crypto operations
- Specify network topology [ns-2]
- Simulate execution

Hypothesis: sufficient to predict performance trends of a native implementation
Structure

• Overview
• Example BFT protocol: Zyzzyva
• Design of BFTSim
• Validation of BFTSim
• Evaluation with BFTSim
• Conclusion
Zyzzyva [Kotla et al. ’07]

4 Replicas (f=1)
Zyzzyva [Kotla et al. ’07]

4 Replicas (f=1)

Client

Proposes Ordering

Send speculative response

Done if “all” respond
Zyzzyva [Kotla et al. ’07]

Client

Proposes Ordering

4 Replicas (f=1)

Send speculative response

Done if “all” respond
Zyzzyva [Kotla et al. ’07]

Client

Proposes Ordering

Send speculative response

Done if “all” respond

do 2nd phase

4 Replicas (f=1)
Zyzzyva [Kotla et al. ’07]

Client

4 Replicas (f=1)
Zyzzyva [Kotla et al. ’07]

Client

4 Replicas (f=1)
Zyzzyva [Kotla et al. ’07]

Client

4 Replicas (f=1)
Zyzzyva [Kotla et al. ’07]

Client

4 Replicas (f=1)

retransmit
Zyzzyva [Kotla et al. ’07]

Client

4 Replicas (f=1)

retransmit
Zyzzyva [Kotla et al. ’07]

Client

4 Replicas (f=1)

retransmit
Zyzzyva [Kotla et al. ’07]

Client

4 Replicas (f=1)

retransmit

Proposes Ordering
Structure

• Overview
• Example BFT protocol: Zyzzyva
• Design of BFTSim
• Validation of BFTSim
• Evaluation with BFTSim
• Conclusion
Implementations in BFTSim

- OverLog P2 [SOSP’05]

\[
\text{action} \ :- \ \text{event}, \\
\text{precondition1}, \\
\text{precondition2}, \\
\text{precondition3}.
\]
Implementations in BFTSim

- OverLog P2 [SOSP’05]

  action :- event,
  precondition1,
  precondition2,
  precondition3.
Implementations in BFTSim

• OverLog P2[SOSP’05]

\[
\text{action} \quad :\quad \text{request}(A, \text{Client}, \text{Timestamp}, \text{Operation}), \\
\text{precondition1,} \\
\text{precondition2,} \\
\text{precondition3.}
\]
Implementations in BFTSim

- OverLog P2 [SOSP’05]

```prolog
action :- request(A, Client, Timestamp, Operation), prevReply(A, Client, TimestampOld, Result), precondition2, precondition3.
```
Implementations in BFTSim

- OverLog P2[SOSP’05]

\[
\text{action} \leftarrow \text{request}(A, \text{Client}, \text{Timestamp}, \text{Operation}), \\
\text{prevReply}(A, \text{Client}, \text{TimestampOld}, \text{Result}), \\
\text{Timestamp} > \text{TimestampOld}, \\
\text{precondition3}.
\]
Implementations in BFTSim

- OverLog P2 [SOSP'05]
  
  `action :- request(A, Client, Timestamp, Operation), prevReply(A, Client, TimestampOld, Result), Timestamp > TimestampOld, isPrimary(A, View).`
Implementations in BFTSim

- OverLog P2 [SOSP’05]

\[
\text{orderReq}(A, ...) \leftarrow 
\begin{align*}
\text{request}(A, \text{Client}, \text{Timestamp}, \text{Operation}), \\
\text{prevReply}(A, \text{Client}, \text{TimestampOld}, \text{Result}), \\
\text{Timestamp} > \text{TimestampOld}, \\
\text{isPrimary}(A, \text{View}).
\end{align*}
\]
Implementations in BFTSim

\[
\text{orderReq}(A, .):= \\
\text{request}(A, \text{Client}, \text{Timestamp}, \text{Operation}), \\
\text{reply}(A, \text{Client}, \text{TimestampOld}, \text{Result}), \\
\text{Timestamp} > \text{TimestampOld}, \\
\text{isPrimary}(A, \text{View}).
\]
Implementations in BFTSim

```
delay(HMAC, send)  \quad \text{orderReq}(A, .):- \quad \text{request}(A, \text{Client}, \text{Timestamp}, \text{Operation}), \quad \text{reply}(A, \text{Client}, \text{TimestampOld}, \text{Result}), \quad \text{Timestamp} > \text{TimestampOld}, \quad \text{isPrimary}(A, \text{View}).
```

Compute intensive operations
Implementations in BFTSim

\[ \text{delay}(\text{HMAC}, \text{send}) \]

\[ \text{orderReq}(A, .) \leftarrow \text{request}(A, \text{Client}, \text{Timestamp}, \text{Operation}), \right. \]
\[ \text{reply}(A, \text{Client}, \text{TimestampOld}, \text{Result}), \]
\[ \text{Timestamp} > \text{TimestampOld}, \]
\[ \text{isPrimary}(A, \text{View}). \]

- Implemented four protocols

<table>
<thead>
<tr>
<th>Protocol</th>
<th>BFTSim</th>
<th>Original</th>
</tr>
</thead>
<tbody>
<tr>
<td>PBFT</td>
<td>130 rules</td>
<td>15k</td>
</tr>
<tr>
<td>Q/U</td>
<td>90 rules</td>
<td>10k</td>
</tr>
<tr>
<td>Zyzzyva/5</td>
<td>150 rules</td>
<td>15k</td>
</tr>
</tbody>
</table>

Compute intensive operations
Architecture of BFTSim

OverLog Program

OverLog Engine

Network Simulator (ns2)
Architecture of BFTSim

OverLog Program

OverLog Engine

Network Simulator (ns2)

Receive

Atul Singh

NSDI 2008
Architecture of BFTSim

OverLog Program

OverLog Engine

Receive

Verify

Network Simulator (ns2)
Architecture of BFTSim

OverLog Program

OverLog Engine

Receive

Verify

Network Simulator (ns2)
Architecture of BFTSim

OverLog Program

OverLog Engine

State Manipulation

Verify

Receive

Network Simulator (ns2)
Architecture of BFTSim

OverLog Program

OverLog Engine

State Manipulation

Verify

Receive

Network Simulator (ns2)
Architecture of BFTSim

- OverLog Program
- OverLog Engine
- OverLog
- Engine
- State Manipulation
- Verify
- Sign
- Receive
- Network Simulator (ns2)
Architecture of BFTSim

OverLog Program

OverLog Engine

State Manipulation

Verify

Sign

Receive

Send

Network Simulator (ns2)
Structure

- Overview
- Example BFT protocol: Zyzzyva
- Design of BFTSim
- **Validation of BFTSim**
- Evaluation with BFTSim
- Conclusions
Can BFTSim match real implementations?

- Compared against either published results or authors’ implementation
- Validation in 3 settings
  - Baseline (f=1, LAN network, no faults)
  - Introduce a faulty (mute) replica
  - Higher replication factor
Baseline validation: PBFT

Throughput (Ops/s)

Latency (sec)

- Published
Baseline validation: PBFT

Throughput (Ops/s)

Latency (sec)

BFTSim

Published
Baseline validation: Zyzzyva

Throughput (Ops/s) vs. Latency (sec)

- BFTSim
- Published
Baseline validation: Q/U

Throughput (Ops/s)

Latency (sec)

BFTSim

Authors' implementation
Summary of validation

• BFTSim results match published results closely
  • Error <= 10%
  • Able to match performance trends

• BFTSim considers three performance factors
  • Crypto cost
  • Message processing cost
  • Network topology
Structure

- Overview
- Example BFT protocol: Zyzzyva
- Design of BFTSim
- Validation of BFTSim
- Evaluation with BFTSim
- Conclusions
What is BFTSim good for?

- Comparison under identical conditions
- Evaluation under a wide range of conditions
  - Varying payloads, lossy links, heterogeneous network connectivity, misconfigured client timers
- Full results in the paper
- Explore protocol optimizations
- Make BFT protocols more accessible
Comparison under Identical Conditions

![Graph showing latency (sec) vs throughput (Ops/s) for different systems.]

- **Q/U**
- **PBFT B=1**
- **Zyzzyva B=1**

Throughput (Ops/s):

0 20000 40000 60000

Latency (sec):

0.001 0.002 0.003 0.004 0.005 0.006
Comparison under Identical Conditions

![Graph showing latency and throughput comparison between Q/U, PBFT B=100, and Zyzzyva B=100.](image)

- **Throughput (Ops/s)**: 0, 20000, 40000, 60000, 80000, 100000
- **Latency (sec)**: 0, 0.002, 0.004, 0.006, 0.008, 0.01, 0.012

Legend:
- **Q/U**
- **PBFT B=100**
- **Zyzzyva B=100**
Varying Payload

- Varying the payload from 2Bytes to 8KBytes

![Graph showing throughput and latency with varying payloads for Q/U, PBFT B=100, and Zyzzyva B=100.](image)

Throughput (Ops/s) vs. Latency (sec) for different payload sizes.
Varying Payload

- Varying the payload from 2Bytes to 8KBytes

![Graph showing performance drops for Q/U, PBFT, and Zyzzyva with varying payload.]
Varying Payload

- Varying the payload from 2Bytes to 8KBytes

Throughput (Ops/s)

Latency (sec)

Q/U is better

Q/U is better

•

•

PBFT

Zyzzyva

Varying the payload from 2Bytes to 8KBytes

•

•

PBFT

Zyzzyva

Varying the payload from 2Bytes to 8KBytes

•

•

PBFT

Zyzzyva

Varying the payload from 2Bytes to 8KBytes

•

•

PBFT

Zyzzyva

Varying the payload from 2Bytes to 8KBytes

•

•

PBFT

Zyzzyva

Varying the payload from 2Bytes to 8KBytes

•

•

PBFT

Zyzzyva

Varying the payload from 2Bytes to 8KBytes

•

•

PBFT

Zyzzyva

Varying the payload from 2Bytes to 8KBytes

•

•

PBFT

Zyzzyva

Varying the payload from 2Bytes to 8KBytes

•

•

PBFT

Zyzzyva

Varying the payload from 2Bytes to 8KBytes

•

•

PBFT

Zyzzyva

Varying the payload from 2Bytes to 8KBytes

•

•

PBFT

Zyzzyva

Varying the payload from 2Bytes to 8KBytes

•

•

PBFT

Zyzzyva
Heterogenous Network Connectivity

Replicas

Clients

slow link
Heterogenous Network Connectivity: Q/U

Performance degrades

Characteristics of the slow link

Throughput (Ops/s)

1 Gbps 0.04 ms
100 Mbps 1 ms
10 Mbps 10 ms
1 Mbps 100 ms
Heterogenous Network Connectivity: PBFT

Characteristics of slow link

Throughput (Ops/s)

0 20000 40000 60000 80000

1 Gbps 100 Mbps 10 Mbps 1 Mbps

0.04 ms 1 ms 10 ms 100 ms

Improves slightly
Heterogenous Network Connectivity: Zyzzyva

Clients do second phase

Throughput (Ops/s)

<table>
<thead>
<tr>
<th>Characteristics of slow link</th>
<th>Throughput (Ops/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Gbps 0.04 ms</td>
<td>80000</td>
</tr>
<tr>
<td>100 Mbps 1 ms</td>
<td>50000</td>
</tr>
<tr>
<td>10 Mbps 10 ms</td>
<td>40000</td>
</tr>
<tr>
<td>1 Mbps 100 ms</td>
<td>30000</td>
</tr>
</tbody>
</table>

Clients do second phase
Heterogenous Network Connectivity: Zyzzyva5

Performance unchanged

Characteristics of slow link:
- 1 Gbps: 0.04 ms
- 100 Mbps: 1 ms
- 10 Mbps: 10 ms
- 1 Mbps: 100 ms

Throughput (Ops/s): 0

Throughput (Ops/s): 100000

Throughput (Ops/s): 80000

Throughput (Ops/s): 60000

Throughput (Ops/s): 40000

Throughput (Ops/s): 20000

Throughput (Ops/s): 0
Exploratory ‘what-if’ questions
Exploratory ‘what-if’ questions

- Q/U’s performance not as expected
  - No contention in our experiments: ideal for Q/U
  - Q/U’s performance still worse compared to Zyzzyva5
- Cause: Q/U sends larger messages
  - Replicas send their history in each response
  - Clients collect histories, combine them and send in the next request
- Idea: optimize message size
Potential Optimization for Q/U

Throughput (Ops/s)

Latency (ms)

- Q/U
- Q/U Optimized Messages
Conclusions

• BFTSim enables fair comparison
• BFTSim enables extensive performance evaluation
• BFTSim makes BFT protocols more accessible

Future work

• More evaluation and more protocols
• Model checking for safety/liveness property

Available at http://bftsim.mpi-sws.org
Backup slides
PBFT [Castro Liskov ’99]

Client

4 Replicas (f=1)
PBFT [Castro Liskov ‘99]

Client

4 Replicas (f=1)
PBFT [Castro Liskov ‘99]

4 Replicas (f=1)

Client

Proposes Ordering
PBFT [Castro Liskov ‘99]

Client

Proposes Ordering

Promise to accept

4 Replicas (f=1)
PBFT [Castro Liskov ‘99]

Client

Proposes Ordering

Promise to accept

Commit if enough promises

4 Replicas (f=1)
PBFT [Castro Liskov ‘99]

Client

Proposes Ordering

Promise to accept

Commit if enough promises

4 Replicas (f=1)
Q/U [Malek et al. ’05]

Client

6 Replicas (f=1)

Atul Singh

NSDI 2008
Q/U [Malek et al. '05]

Collect Replica History, send in next request

Client

6 Replicas (f=1)

NSDI 2008
Q/U [Malek et al. ’05]

Collect Replica History, send in next request

Client

6 Replicas (f=1)

+Optimistic single-phase execution
Zyzzyva [Kotla et al. ’07]

Client

4 Replicas (f=1)
Zyzzyva [Kotla et al. ’07]

Client

Proposes Ordering

Send speculative response

Done if “all” respond

4 Replicas (f=1)
Zyzzyva [Kotla et al. ’07]

Client

Proposes Ordering

Send speculative response

Done if “all” respond

Else, do second phase

4 Replicas (f=1)
Zyzzyva [Kotla et al. ’07]

Client

Proposes Ordering

4 Replicas (f=1)

Send speculative response

Zyzzyva5

Done if “all” respond

Else, do second phase

Zyzzyva5

Atul Singh

NSDI 2008