Attested Append-Only Memory: Making Adversaries Stick to their Word

Presented by Chuan He
Talk Outline

• Introduction and Motivation

• Attested Append-Only Memory (A2M)

• A2M Protocols

• Evaluation

• Conclusion
Motivation

- You want to build a service
  - Easy on a single machine
  - Replicate service on multiple machines

- Replicated services must appear as single server
  - Linearizability: Completed client requests appear to have been processed in a single, totally ordered, serial schedule consistent with the order they were submitted
Motivation

- You want to build a service
  - Easy on a single machine
  - Replicate service on multiple machines

- Replicated services must appear as single server
  - Equivocation: Different lies to different people
Servers Equivocating to Clients
Servers Equivocating to Servers

- faulty
- non-faulty
- client

Diagram showing a client sending requests to servers and the servers responding with equivocating responses. The timeline indicates the progression of time with requests and responses marked as `<1,req_a>` and `<1,req_b>`.
Questions

• Does preventing equivocation help at all?
  – Can we improve upon the 1/3 Byzantine fault bound?

• How do we prevent equivocation?
  – Is there any minimal system support?
Talk Outline

• Introduction and Motivation

• Attested Append-Only Memory (A2M)

• A2M Protocols

• Evaluation

• Conclusion
Attested Append-Only Memory (A2M)

- A set of numbered logs
- Each log entry contains
  - Sequence number
  - Stored value
  - Crypto digest

**lookup / end**
- Get a log entry
- Attest (sequence number, value, history digest)
- Attest freshness
- Attest the end of log

**append / advance**
- Cannot overwrite
Attested Append-Only Memory (A2M)

- append / advance
  \[ d_H = h(Hx||d_{H-1}) \]
- Important feature
  – Cannot equivocate
Background: PBFT

Quorum: matching messages from different replicas
A2M-PBFT-E (Execution)

Primary

s1

Preprepare

Preprepare

Preprepare

Preprepare

Prepare

Prepare

Prepare

Prepare

Commit

Commit

Commit

Commit

Execute

Execute

Execute

Execute

Quorum = 3

Quorum = 3

Quorum = 3

Quorum = 3

Request log

Request log

Request log

Request log

req, resp, <seq, req, hist>

req, resp, <seq, req, hist>

req, resp, <seq, req, hist>

req, resp, <seq, req, hist>

Reply

Reply

Reply

Reply

time

time

time

time

Request

Request

Request

Request

Client1

Client1

Client1

Client1

Attested by A2M

Attested by A2M

Attested by A2M

Attested by A2M

Attested by A2M
A2M-PBFT-EA (2f + 1 replicas)

Primary
s1

Preprepare

Prepare

Commit

Execute

s2

s3

Quorum = 2

req,resp,<seq,req,hist>

Reply time

Request

Client1

Attested by A2M
Protocol Trade-offs

PBFT

A2M-PBFT-E

A2M-PBFT-EA
Evaluation Setup

• Implemented A2M-PBFT-E and A2M-PBFT-EA
• A2M protocols use signatures or MACs for authentication

• Four replicas in a LAN. Each replica has its own A2M.

• Microbenchmarks
  – Null operation with various request or response sizes
• Macrobenchmarks: NFS
  – Software package compilation
Evaluation - Microbenchmarks

![Graphs showing processing time vs request and response size for different protocols.](image)
## Evaluation - Macrobenchmarks

<table>
<thead>
<tr>
<th>Phase</th>
<th>NFS</th>
<th>-S</th>
<th>-PBFT</th>
<th>-A2M-PBFT-E (sig)</th>
<th>-A2M-PBFT-E (MAC)</th>
<th>-A2M-PBFT-EA (sig)</th>
<th>-A2M-PBFT-EA (MAC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy</td>
<td>0.219</td>
<td>0.709</td>
<td>1.026</td>
<td>0.728</td>
<td>2.141</td>
<td>0.763</td>
<td></td>
</tr>
<tr>
<td>Uncompress</td>
<td>1.015</td>
<td>3.027</td>
<td>4.378</td>
<td>3.103</td>
<td>8.601</td>
<td>3.236</td>
<td></td>
</tr>
<tr>
<td>Untar</td>
<td>2.322</td>
<td>4.448</td>
<td>6.826</td>
<td>4.553</td>
<td>12.896</td>
<td>4.669</td>
<td></td>
</tr>
<tr>
<td>Clean</td>
<td>0.180</td>
<td>0.298</td>
<td>0.640</td>
<td>0.312</td>
<td>0.742</td>
<td>0.311</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>23.725</td>
<td>28.355</td>
<td>41.821</td>
<td>28.854</td>
<td>61.940</td>
<td>29.528</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Mean time to complete the six macrobenchmark phases in seconds.
Varying delay time

<table>
<thead>
<tr>
<th>Additional latency (μs)</th>
<th>NFS- A2M-PBFT-E (MAC)</th>
<th>A2M-PBFT-E (MAC) with batching</th>
<th>A2M-PBFT-EA (MAC)</th>
<th>A2M-PBFT-EA (MAC) with batching</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>28.854</td>
<td>28.763</td>
<td>29.528</td>
<td>29.505</td>
</tr>
<tr>
<td>10</td>
<td>29.598</td>
<td>29.025</td>
<td>31.299</td>
<td>30.188</td>
</tr>
<tr>
<td>50</td>
<td>32.735</td>
<td>30.232</td>
<td>36.242</td>
<td>32.214</td>
</tr>
<tr>
<td>250</td>
<td>48.784</td>
<td>37.237</td>
<td>66.441</td>
<td>45.199</td>
</tr>
<tr>
<td>1000</td>
<td>117.59</td>
<td>65.813</td>
<td>192.53</td>
<td>101.62</td>
</tr>
</tbody>
</table>

Table 2: Mean time to complete the six macrobenchmark phases in seconds for different A2M additional latency costs.
Conclusions

• Present A2M, a small trusted, log-based memory
  – Simple and easily implementable
  – Prevent equivocation

• Improve fault tolerance by forcing servers to commit to a single history of operations
  – Improve fault bounds of BFT state machine replication
  – Achieve linearizability in an untrusted single-server system
  – The benefits are achieved with small performance overhead
Thank you!
Related Work

• Weaken the guarantee
  – fork* consistency [NSDI07]
  – fork consistency [OSDI04]

• Standard trusted hardware like TPM
  – does not improve the fault bound

• Auditing
  – PeerReview [SOSP07], CATS [FAST07]

• Shared file servers
  – SUNDR[OSDI04], Ivy [OSDI02], Plutus[FAST03]

• Separating agreement from execution

• Symmetric faults – hybrid fault model

• Group communication