Beyond Consensus in Permissioned Ledgers: Experiences in using BFT replication on DLTs

Alysson Bessani
Associate Professor

A view of permissioned blockchains

- Decentralized trusted networked services
  - Blockchains are instances of that…
- Distributed trust on the Internet (Cachin’01)
  - Systems that don’t trust any single entity
- Intrusion-tolerant systems (Fraga & Powell’85)
- Requires Byzantine Fault-Tolerant (BFT) consensus
HYPERLEDGER FABRIC

• Open-source, modular, permissioned
• Architecture: not all “peers” are equal
**Hyperledger Fabric**

Ordering Service

- Ordering node state:
  - the ordered transactions not yet in a block,
  - header of the last generated block, and
  - latest configuration block

---

**c·rda**

- Open-source blockchain project targeting (at least initially) the financial market
- Key idea: **there is no shared global ledger**
  - Instead, **there are many distributed ledgers**
• Only participants of a transaction have to execute and validate it
• A transaction is committed only if it achieve
  • Validity consensus: all involved participants need to validate and sign the transaction
  • Uniqueness consensus: requires a notary service

• Notary implements an key-value store that register all state “consumptions”
• Some specific transaction validation might be executed
• Multiple notaries might be used
Consensus

Validator

Client

State Machine Replication

Safety: all replicas execute the same sequence of transactions

Liveness: transactions issued by correct clients are answered

- State machine replication middleware written in Java ("seriously" developed and maintained since 2010)
- Can be configured to tolerate only crashes
- Available under Apache license
- Similar to PBFT in normal case, but it isn’t PBFT


- Leverages trusted computing to constraint adversarial behaviour (i.e., requires TPM or SGX)
- Requires the same number of replicas, comm. steps and message complexity than crash protocols (e.g., Paxos, Raft)
Other protocols: HotStuff [PODC’19] (Libra)

- Linear message/authenticator complexity
- Responsiveness (as all "classical" BFT protocols)
- It’s possibly simpler than other BFT protocols

BFT-SMaRt Performance (gigabit LAN, no disks)

\[ \text{Throughput} = \frac{\text{(1000x tx/sec)}}{f = 1} \]

\[ \text{Throughput} = \frac{\text{(1000x tx/sec)}}{f = 2} \]

\[ \text{Throughput} = \frac{\text{(1000x tx/sec)}}{f = 3} \]

\[ \text{f = number of tolerated failures} \]

\[ \text{Crash: } n = 2f+1, \text{ Byzantine: } n = 3f+1 \]
Consensus is not enough

- A consensus engine also needs:
  - **Durability**: any request completed at a client is reflected in the service after a recovery (more than $f$ replicas can be faulty, but not Byzantine)
  - **Crash recovery**: recovered replicas need to be synched
  - **Reconfiguration**: replica group changes

---

Durability = Stable Logging

<table>
<thead>
<tr>
<th></th>
<th>Throughput (4kB-txs/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td>4772</td>
</tr>
<tr>
<td>Async Disk</td>
<td>4312</td>
</tr>
<tr>
<td>Sync Disk</td>
<td>63</td>
</tr>
<tr>
<td>Sync SSD</td>
<td>1017</td>
</tr>
</tbody>
</table>

---
More features = More Complexity

SMR Complexity (LoCs & Module dependencies)

- Blockchain (?)
- Production-level system
- Decent PhD-level prototype
- Accepted paper
- Rejected paper
- Fault-free execution
- High performance
- Leader change
- Recoveries
- Reconfigurations

BFT-SMaRt

- Techniques for efficient durability
  - Parallel Logging
  - Sequential checkpoints
  - Collaborative state transfer

Client App.

invoke

SMR Client Side

SMR Server Side

Service

setState

execute

logBatch

Keeper

ckp

Stable Storage

setState

getState

execBatch

invokeST

handlerST

Dura-Coordinator
BFT-SMaRt Performance under “sporadic” events

BFT-SMaRt as a Blockchain

- Recently, we’ve been building SMaRtChain, an experimental, feature-minimal blockchain “platform” based on BFT-SMaRt
  - Stable logs as blockchains
  - Improved durability guarantees
  - Fully distributed reconfiguration protocols

- Performance (preliminary numbers):

<table>
<thead>
<tr>
<th>Platform</th>
<th>Throughput (tx/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMaRtChain</td>
<td>~ 13k</td>
</tr>
<tr>
<td>Tendermint</td>
<td>~ 2k</td>
</tr>
<tr>
<td>Fabric (not BFT)</td>
<td>&lt; 1k (3k in the paper)</td>
</tr>
</tbody>
</table>

1kB transactions and networks tolerating a single Byzantine failure
BFT-SMaRt on other Blockchains

- Symbiont Assembly (rewrote BFT-SMaRt in Go)
- Experimental Corda BFT notary
- BFT orderer for Hyperledger Fabric [DSN'18]
It can be observed that when using 10 envelopes/block (d) 4 orderers, 100 envelopes/block.

Figures 6d, 6e, and 6f show the results obtained for 100 replicas, with Virginia standing as WHEAT's additional frontends at 3 frontends scattered across the Americas, with the deployment in a local datacenter, we also conducted a geo-distributed Ordering Cluster. This means that for smaller envelopes/sec capacity of only signatures exhaustion. It can be observed that, across all cluster sizes, the overhead associated with BFT-SM is approximately 50k transactions/second (when there exists tolerated Byzantines OR 2f+1 nodes to tolerate crashes).

Whereas for larger envelope sizes this is due to the overhead values across all combinations of envelope/cluster/block sizes. Interestingly, for a larger number of nodes (7 nodes onward). This is because for larger envelope sizes – the predominant overhead becomes behavior is similar to using 10 envelopes/block, specially from (Figures 6a, 6b, and 6c), the maximum throughput observed is unable to 2 receivers and envelope sizes of 1 and 4 kbytes, the peak 3f+1 nodes to tolerate Byzantines OR 2f+1 nodes to tolerate crashes.

As discussed previously – the predominant overhead becomes the replication protocol. Interestingly, for a larger number of nodes (7 nodes onward). This is because for larger envelope sizes – the predominant overhead becomes behavior is similar to using 10 envelopes/block, specially from (Figures 6a, 6b, and 6c), the maximum throughput observed is unable to 2 receivers and envelope sizes of 1 and 4 kbytes, the peak 3f+1 nodes to tolerate Byzantines OR 2f+1 nodes to tolerate crashes.

It can be observed that when using 10 envelopes/block (d) 4 orderers, 100 envelopes/block.

Fig. 6: BFT-SM Ordering Evaluation (LAN)
Integration with Hyperledger Fabric 1.3

• Check it out: https://github.com/bft-smart/fabric-orderingservice
• Already dockerized; includes recovery, reconfiguration, etc.
• Lessons learned:
  • Redundant signatures during block creation
  • Too many validations on the ordering service
  • Orderer framework is mostly designed for crash fault tolerance
  • It would be great if Fabric (as a project) curates a list of extensions and orderers developed by the community

A R&D Agenda (for BFT SMR)

• Scalability & Elasticity
  • Increase performance dynamically w/ additional replicas
• Geo-replication
  • distributed trust
• Diversity and Fault Independence
  • How to withstand $f$ malicious faults?
Geo-replication: WHEAT & AWARE [SRDS’15,’19]

- Employs a single, well-connected leader (better than multiple leaders)
- Safe weighted replication (to not violate the resilience bound \(f\))
- Reliable self-measurements to adapt the weights at runtime

![Figure 2: Possible quorums for \(n = 5, f = 1, \Delta = 1\) (BFT).](image1)

![Figure 4: Message flow of BFT AWARE (\(f = 1; \Delta = 1\)).](image2)
Diversity Management: Lazarus

UNTRUSTED
BFT-Replicated service

Execution plane
Control plane
R0 R1 R2 Rn
LTU
BFT agreement
LTU LTU LTU
Controller

TRUSTED
Small local trusted component that reboots the host
Logically-centralized orchestrator for reconfigurations

Clients
OSINT
Not controlled by the adversary

Questions?

• Alysson Bessani
  • anbessani@fc.ul.pt
  • www.di.fc.ul.pt/~bessani  @AlyssonBessani
  • To know more:
    • BFT-SMaRt & BFT Fabric Orderer: https://github.com/bft-smart/
    • Sousa, Bessani. From Byzantine Consensus to BFT State Machine Replication: A Latency-optimal Transformation. EDCC’12.
    • Bessani et al. State Machine Replication for the Masses with BFT-SMaRt. DSN’14.
    • Sousa, Bessani. Separating the WHEAT from the Chaff: An empirical design for georeplicated state machines. SRDS’15.