Byzantine Consensus in the Jungle
Geographically-Scalable BFT with Adaptive Weighted Replication

Alysson Bessani

Joint work with João Sousa, Christian Berger, and Hans P. Reiser
Byzantine Fault Tolerance Protocols

- **Performance**
  - The racehorses: PBFT, Zyzzyva, Alyph, …

- **Robustness**
  - Slow but steady: Prime, Aardvark, RBFT, …

- **Resource efficiency**
  - Strong assumptions: MinBFT, CheapBFT, XFT, …

- **Scalability**
  - Blockchainers: HoneyBadger, FastBFT, SBFT, …
BFT-SMaRt


BFT-SMaRt

• Byzantine/ Crash fault tolerant state machine replication library
  • Written in Java, maintained and evolved during more than 10 years
  • Available under Apache license: http://bft-smart.github.io/library/
• Key features: modularity, reconfigurations, robustness, performance
Mod-SMaRt: Normal Phase

Client

P0
P1
P2
P3

Request
Propose
Write
Accept
Reply

VP-Consensus

Validation is done here

Validated and Provable

Proof comprised of these signed messages
Mod-SMaRt: Synchronization Phase

IMPORTANT:
It looks like PBFT, but it is not PBFT 😊
Some Facts about BFT Consensus in WANs

• There’s not much experience with BFT consensus in production on the internet
  • Permissionless blockchains solve eventual consensus
  • (as far as I know) There’s no BFT consensus in production on the Internet
    • Stellar and Ripple is the closest we have...
  • Even CFT systems (Paxos, RAFT) are rarely used in this context

• Decentralization and fault independence requires BFT consensus peers to be deployed on different sites
  • Otherwise, it is difficult to justify the use of BFT?
Some Facts about BFT Consensus

• Node-scalability is not always required for BFT
  • Current consortia typically are small (10s of peers)
    • Libra implements classical state machine replication
    • It aims to 100 validators at launch

• Most permissioned systems tend to isolate consensus in a subset of peers
• 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**
• Notary implements a key-value store that register all state “consumptions”

• Some specific transaction validation might be executed

• Multiple notaries might be used
Geographically-Scalable BFT
Issues with Geo-Replication

• Different administrative domains
• Performance diversity
  • Across replicas
  • Across time
• Throughput can be improved with better networks
• Latency requires protocol optimizations
  • Speed of light is the network limit
  • Latency proportional to the roundtrip to a fast quorum
Classic vs Fast Paxos

<table>
<thead>
<tr>
<th></th>
<th>Classic Paxos</th>
<th>Fast Paxos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comm. steps</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Number of replicas</td>
<td>$2t + 1$</td>
<td>$3t + 1$</td>
</tr>
<tr>
<td>Quorum size</td>
<td>$t + 1$</td>
<td>$2t + 1$</td>
</tr>
</tbody>
</table>

Comparison is made through trace-driven simulations using latencies from 2002 obtained from the internet weather service.

Classic Paxos is faster than Fast Paxos 60% of times

Experimental study conducted with BFT-SMaRt on Planetlab and Amazon EC2

Summary:
- Leader in the best-connected site yields better results than employing a rotating or multiple leader(s) strategy
- Smaller quorums create opportunities for improving latency
Our solution: WHEAT + AWARE


Classical BFT Replication

N=4, f=1

Egalitarian quorums,
Any 3 out-of 4 replica
**WHEAT**: WeigHt-Enabled Active replicaTion

- Use optimizations that lead to significant latency reduction:
  - Single leader in the best-connected site
  - Tentative executions (from PBFT)
  - Employs smaller quorums (weighted replication)

- Weighted replication: safe voting assignment scheme for SMR
  - Uses $\Delta$ extra replica(s) for quorum formation
  - Improves latency by enabling more choice upon quorum formation
  - Needs a to preserve quorum intersection and tolerance to $f$ faulty replicas
Weighted BFT Replication

\[ N=4, f=1, \Delta=1 \text{ (extra)} \]

Weighted quorums,
One set of 3 out-of 5
and any set 4 out-of 5
Weighted BFT Replication

- **Consistency**: All quorums that hold Q votes intersect by at least one correct replica
- **Availability**: There is always a quorum available in the system that holds Q votes
- **Safe minimality**: There exists at least one minimal quorum in the system
Weighted BFT Replication

Define the number of replicas $u$ that hold $V_{\text{max}} > 1$ votes, without violating $f$

<table>
<thead>
<tr>
<th>Mode</th>
<th>$n$</th>
<th>$N_v$</th>
<th>$Q_v$</th>
<th>$u$</th>
<th>$F_v$</th>
<th>$V_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFT mode</td>
<td>$n = 2f + 1 + \Delta$</td>
<td>$N_v = \sum V_i = 2F_v + 1$</td>
<td>$Q_v = F_v + 1$</td>
<td>$u = f$</td>
<td>$F_v = \Delta + f$</td>
<td>$V_{\text{max}} = \frac{\Delta + f}{f} = 1 + \frac{\Delta}{f}$</td>
</tr>
<tr>
<td>BFT mode</td>
<td>$n = 3f + 1 + \Delta$</td>
<td>$N_v = \sum V_i = 3F_v + 1$</td>
<td>$Q_v = 2F_v + 1$</td>
<td>$u = 2f$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Input: $f$ and $\Delta$

Output: $u$ and $V_{\text{max}}$
Size of fast quorums with different $f$ and $\Delta$

- $3f+2$ ($\Delta = 1$)
- $4f+1$ ($\Delta = f$)
- $5f+1$ ($\Delta = 2f$)
- $6f+1$ ($\Delta = 3f$)

The graph shows the relative size of the small quorum for different values of $f$ and $\Delta$. The x-axis represents the percentage of messages, and the y-axis represents the relative size of the small quorum.
AWARE: Adaptive Wide-Area REplication

The benefit of weighted replication depends on choosing an optimal weight configuration.

- The environment of the system (i.e., network characteristics) may change at runtime (e.g., due to a DDoS attack).

AWARE enables a geo-replicated consensus-based system to adapt to its environment!
AWARE Approach

- **Self-Monitoring**
  - AWARE uses reliable self-monitoring as decision-making basis for adapting replicas’ voting weights and leader position at runtime

- **Self-Optimization**
  - AWARE continuously strives for consensus latency gains at runtime
  - Changes weights and leader location to minimize consensus latency
Self-Monitoring: Measuring Latency

- Each replica measures its point-to-point latency to other replicas for consensus protocol messages

- **Non-Leader’s Propose**
  - Periodically an alternately selected dummy leader broadcasts a dummy proposal

- **Write-Response**
  - Replicas immediately respond by sending acknowledgments
Self-Monitoring: Consolidating Measurements

• Replicas periodically disseminate their measurements to others with total order until they have the same latency matrices.

• AWARE maintains synchronized matrices for both PROPOSE and WRITE latencies $\hat{M}^P$ and $\hat{M}^W$ used for decisions later.

<table>
<thead>
<tr>
<th></th>
<th>Oregon</th>
<th>Ireland</th>
<th>Sydney</th>
<th>Sao Paulo</th>
<th>Virginia</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oregon</td>
<td>0</td>
<td>65</td>
<td>69</td>
<td>92</td>
<td>40</td>
</tr>
<tr>
<td>Ireland</td>
<td>65</td>
<td>0</td>
<td>132</td>
<td>93</td>
<td>38</td>
</tr>
<tr>
<td>Sydney</td>
<td>69</td>
<td>132</td>
<td>0</td>
<td>158</td>
<td>105</td>
</tr>
<tr>
<td>Sao Paulo</td>
<td>92</td>
<td>93</td>
<td>158</td>
<td>0</td>
<td>61</td>
</tr>
<tr>
<td>Virginia</td>
<td>40</td>
<td>38</td>
<td>105</td>
<td>61</td>
<td>0</td>
</tr>
</tbody>
</table>
Self-Optimization

• With the same matrices $\hat{M}^P$ and $\hat{M}^W$ the replicas can **solve deterministically** the following optimization problem:

$$\langle \hat{l}, \hat{W} \rangle = \arg\min_{W \in \mathcal{W}, l \in \mathcal{L}} \text{PredictLatency}(l, W, \hat{M}^P, \hat{M}^W)$$

• All correct replicas reach the same, optimal weight distribution and invoke a reconfiguration in the system
Evaluation of WHEAT and AWARE
Setup

- **AWARE** is implemented on top of WHEAT, which is based on BFT-SMaRt
- For evaluation, we use the Amazon AWS cloud, using EC2 instances of t2.micro type with 1 vCPU, 1 GB RAM and 8 GB SSD volume
- We select regions **Oregon, Ireland, Sydney, São Paulo** and **Virginia** for instances (1 client and 1 replica on each instance)
- Clients simultaneously send 1kB-requests across all sites
Clients’ Request Latency

Average latencies of all clients and 20 configurations

Observations

- The best configuration \(<4,0>\) performs about 39% faster than the median \(<3,4>\), 64% faster than the worst \(<2,1>\)
- Tuning voting weights can reduce latency (see configs. with the same leader)
- Leader relocation may be necessary for achieving optimal consensus latency
D. Runtime Behavior of AWARE

We deploy AWARE in our usual setting and observe its behavior during the system's lifespan. Overall, the clients' request latencies show high variations which is caused by variations in quorum formation. According to the model, it predicts a similar request latency, as we can see in the simulation. As expected, consensus speed contributes to total latency.

AWARE attributes one of the replicas to the leader: Sydney, São Paulo (4 vCPU, 8 GB of RAM, 8 GB SSD). This leads to clients observing latency variability in quorum formation. The worst-case latency is over configurations where a leader change is necessary. On a smaller quorum, this leads to clients observing higher request latencies. We notice a positive correlation between the average clients' request latency and the number of times the leader consensus latency changes.

We induce events to evaluate AWARE's behavior in the network. These events are:

1. System Start: Leader: Sydney, Vmax: Sydney, São Paulo
2. Reconfig. to Leader: Oregon, Vmax: Oregon, Ireland
3. Replica Ireland becomes slow
4. São Paulo gets Vmax of Ireland
5. Replica Ireland becomes fast again
6. Ireland gets Vmax of São Paulo
7. Leader Oregon crashes
8. Ireland becomes leader, crashed Oregon still has Vmax
9. São Paulo gets Vmax of Oregon

The reaction to these events is that the clients observe faster request latencies identical to the latency gains observed after the first reconfiguration (Event 2). The latency still stabilizes and the communication links of Ireland are not co-located on the same VM). The worst is when there is an optimal configuration and AWARE might not always choose the actual best configuration but decide for some configuration that is close to the optimum. Nevertheless, the measurement series amortized over configurations are between 100 ms and 148 ms.
AWARE Throughput

Observations

- Low consensus latency indeed has positive effects on throughput for different batch sizes
- The monitoring overhead induced by the Dummy-Propose is noticeable, but still passable, given that AWARE’s main ambition is latency optimization
BFT Ordering with AWARE

Latency across clients before and after optimization*
AWARE with More Nodes: the challenge

- Number of configurations explodes

\[
\left(\frac{3f + 1 + \Delta}{2f}\right) \cdot 2f = \frac{\prod_{i=2f}^{3f+1+\Delta} i}{(f + 1 + \Delta)!}
\]

- Finding the best configuration becomes a huge challenge

Number of weight distribution possibilities

Possible leader location
AWARE w/ More Nodes: simulated annealing

(b) Computation time.

(c) Approximation quality.
AWARE with More Nodes

![Graphs showing latency results for AWARE with different numbers of additional replicas](image-url)
Summary: WHEAT + AWARE

- **Ease of deployment**
  - AWARE provides the needed automation for finding an optimal configuration by tuning voting weights and/or relocating the leader

- **Adjust to varying conditions**
  - AWARE dynamically adjusts to changing conditions by shifting high voting power to replicas that are the fastest in a recent time frame

- **Compensate for faults**
  - AWARE detects (non-malicious) high-weight replicas failures and restores the availability of up to $f (V_{max} - V_{min})$ voting power by redistributing high weights

- **Ultimately, it is a way to deal with heterogeneity**
Questions?

• Alysson Bessani
  • anbessani@fc.ul.pt
  • www.di.fc.ul.pt/~bessani

• 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.
  • Sousa, Bessani. Separating the WHEAT from the Chaff: An Empirical Design for Geo-replicated State Machines. IEEE SRDS’15.