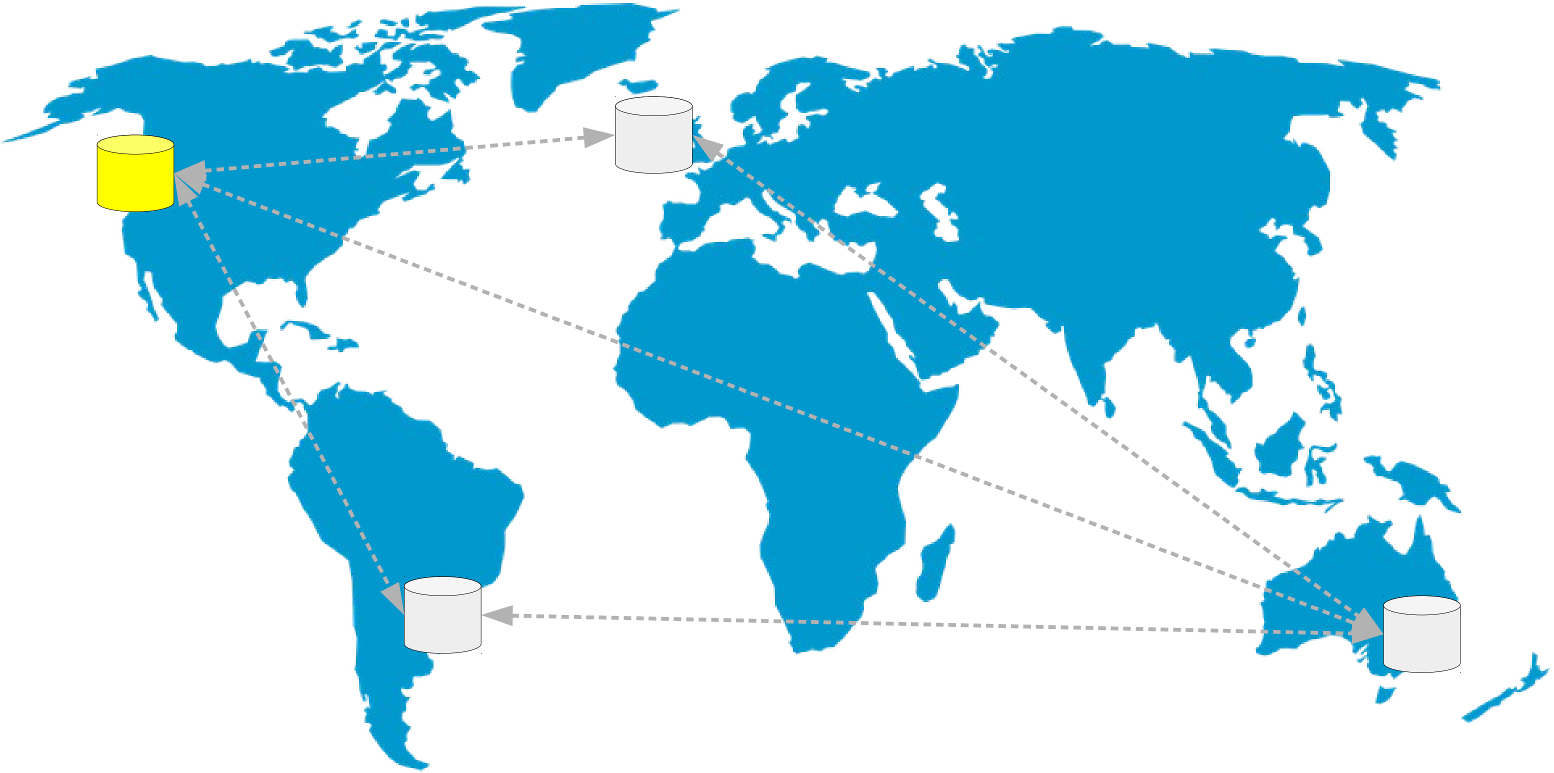
Resilient Wide-Area Byzantine Consensus Using Adaptive Weighted Replication

Christian Berger*, Hans P. Reiser*, João Sousa**, Alysson Bessani**

*University of Passau, Germany
**LASIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal

Wide-Area Byzantine Consensus



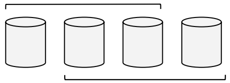
System

N=4, f=1

Legend

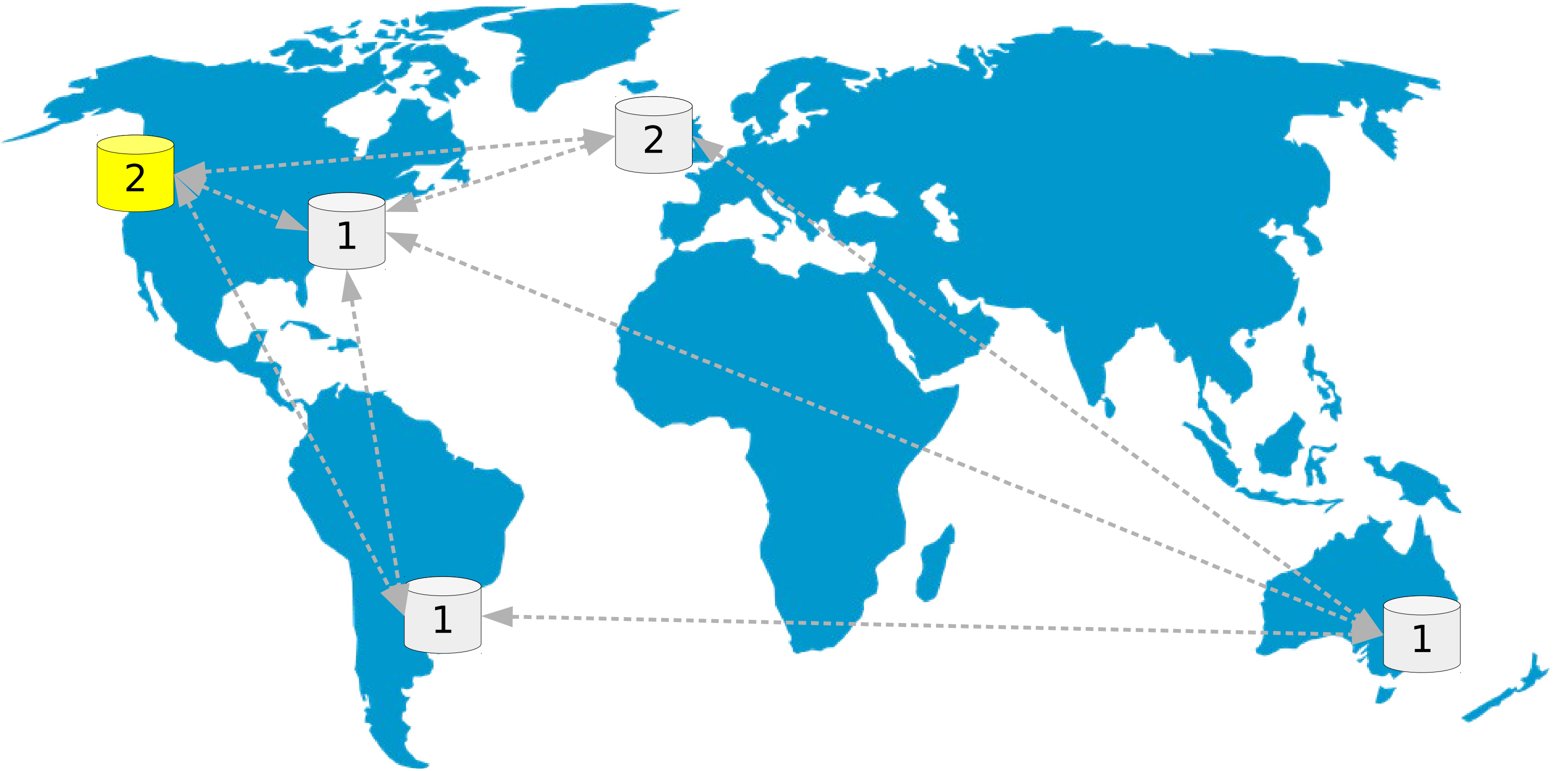
leader

Quorum size



Egalitarian quorums,
Any 3 out of 4 replica

WHEAT: WeigHt-Enabled Active ReplicaTion*



System

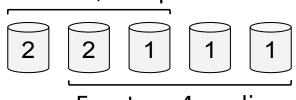
N=5, **f=1,** Δ=1

Legend

leader

replica has voting power x

Quorum size



Weighted quorums

* Sousa, João, and Alysson Bessani. "Separating the WHEAT from the chaff: An empirical design for geo-replicated state machines." *34th IEEE Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 2015.

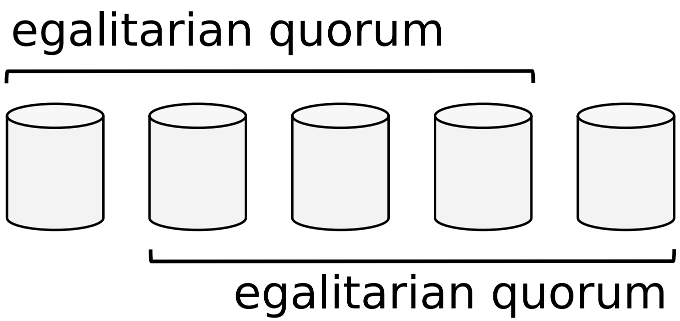
Purpose of doing this? Latency gains!

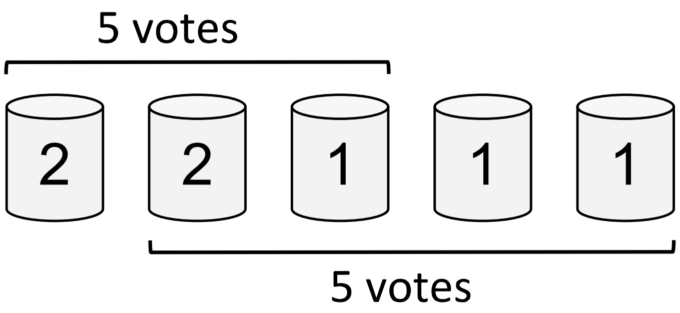
- **To improve latency, we need to make enhancements on protocol level**
- WHEAT
  - utilizes the heterogeneous latencies of links between replicas
  - assigns higher voting power to well-connected replicas
  - benefits from more variety in quorum formation
  - allows replicas to faster make progress by accessing a proportionally smaller quorum of replicas

* Sousa, João, and Alysson Bessani. "Separating the WHEAT from the chaff: An empirical design for geo-replicated state machines." *34th IEEE Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 2015.

Weighted Replication

- Weighted replication is **safe** and **does not violate the resilience bound f**
- Possible quorums for a n=5, f=1 system:



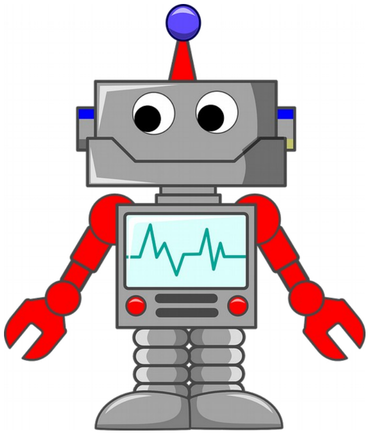


Egalitarian: all quorums contain $\lceil \frac{n+f+1}{2} \rceil$ replicas.

Weighted: a quorum contains at most *n - f* and at least *2f + 1* replicas.

Remaining challenges? Automation needed!

- The benefit of weighted replication depends on **choosing an optimal weight configuration** (a non-trivial problem!)
- The environment of the system (i.e. network characteristics) may **change at runtime** (e.g., due to a DDoS attack)



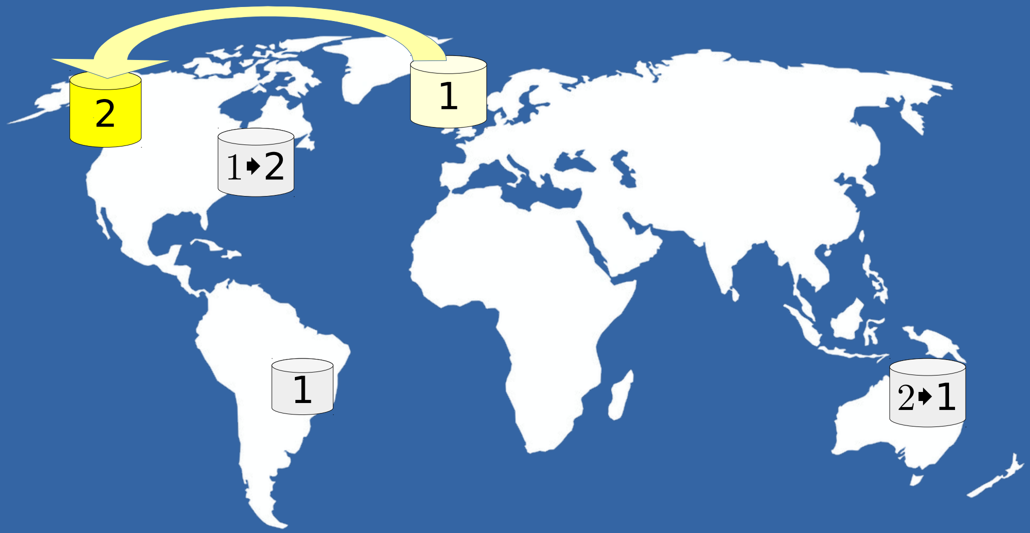
AWARE (**A**daptive **W**ide-**A**rea **RE**plication) enables a geo-replicated system to **adapt to its environment!**

Practical Use?

- Recent blockchain developments (e.g., Libra, Tendermint, Hyperledger) might employ Byzantine consensus in a geographically distributed environment
  - Then they could benefit from **adaptive, weighted replication**

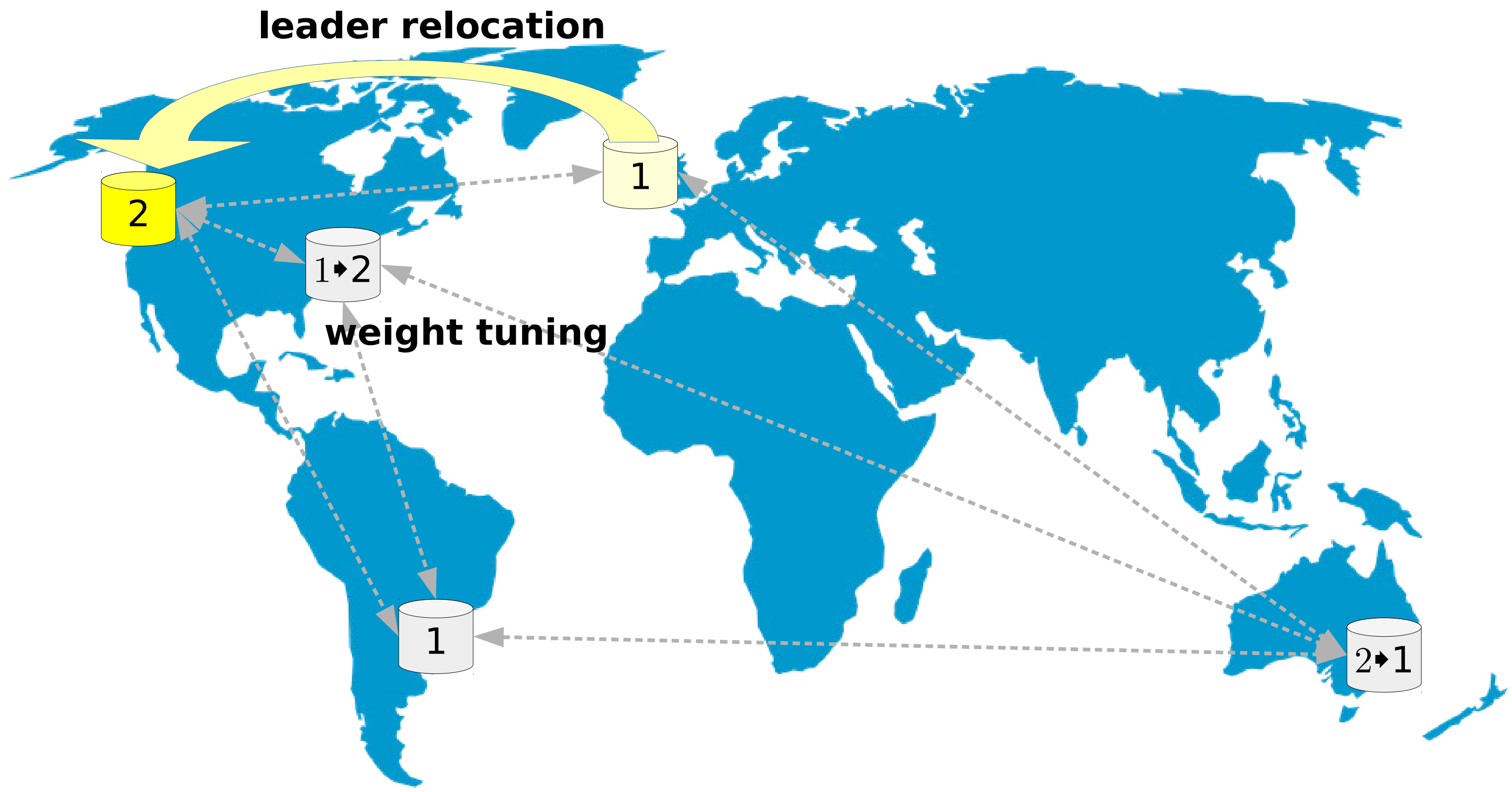
- AWARE can be used as basis for consortium-based blockchains

HYPERLEDGER Tendermint libra



The AWARE protocol

Resilient Wide-Area Byzantine Consensus Using Adaptive Weighted Replication



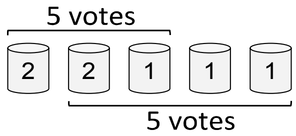
System

N=5, **f=1,** Δ=1

Legend

- leader
- x: replica has voting power x

Quorum size



Adapt to environment

- 1➔2: voting weight tuning
- leader relocation

AWARE Approach

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

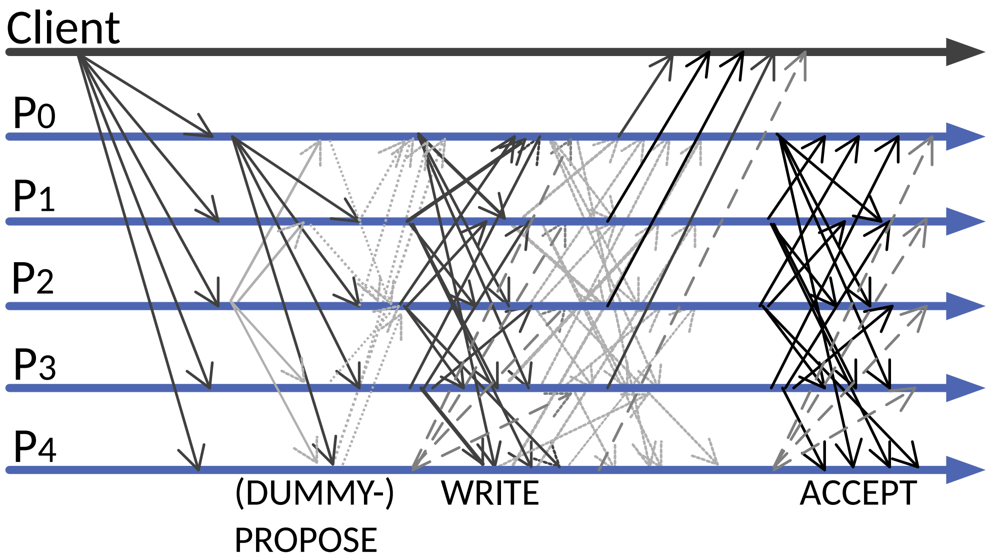
- **Self-Optimization**
  - AWARE continuously strives for latency gains at runtime
  - It optimizes voting weights and leader position to minimize consensus latency

Self-Monitoring: Measuring Latency

- **Measuring latency**: Each replica measures its point-to-point latency to all other replicas from its own perspective 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: Disseminating Measurements

- **Dissemination of measurements**
  - Replicas periodically disseminate their measurements with **total order**
  - Replicas maintain the same latency matrix after some specific consensus instance
  - AWARE maintains **synchronized matrices** for both Propose and Write latencies $\hat{M}^P$ and $\hat{M}^W$ used for decisions later

	Oregon	Ireland	Sydney	Sao Paulo	Virginia
Oregon	0	65	69	92	40
Ireland	65	0	132	93	38
Sydney	69	132	0	158	105
Sao Paulo	92	93	158	0	61
Virginia	40	38	105	61	0

Self-Optimization

- Assume replicas have a synchronized, sanitized latency matrix $\hat{M}$

- When a defined number of consensus is reached, each replica **deterministically** solves the following optimization problem:

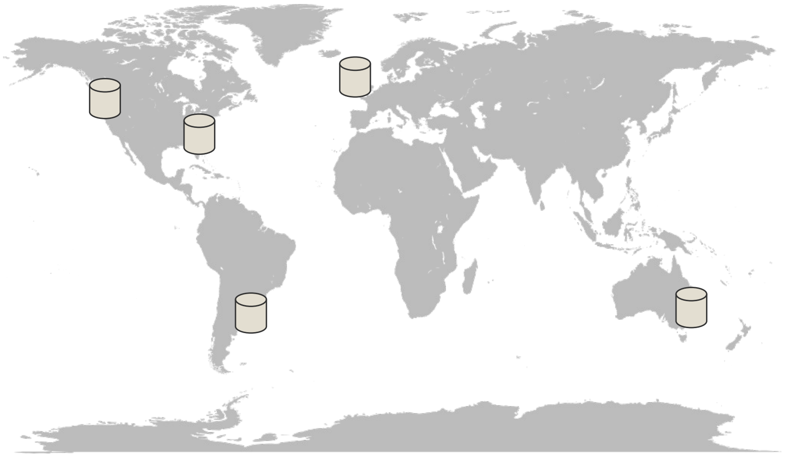
$$\langle \hat{l}, \hat{W} \rangle = \underset{W \in \mathfrak{W}, l \in \mathfrak{L}}{\arg\min} \ PredictLatency(l, W, \hat{M}^P, \hat{M}^W)$$

- All replicas reach the same, optimal weight distribution

Evaluation

Setup

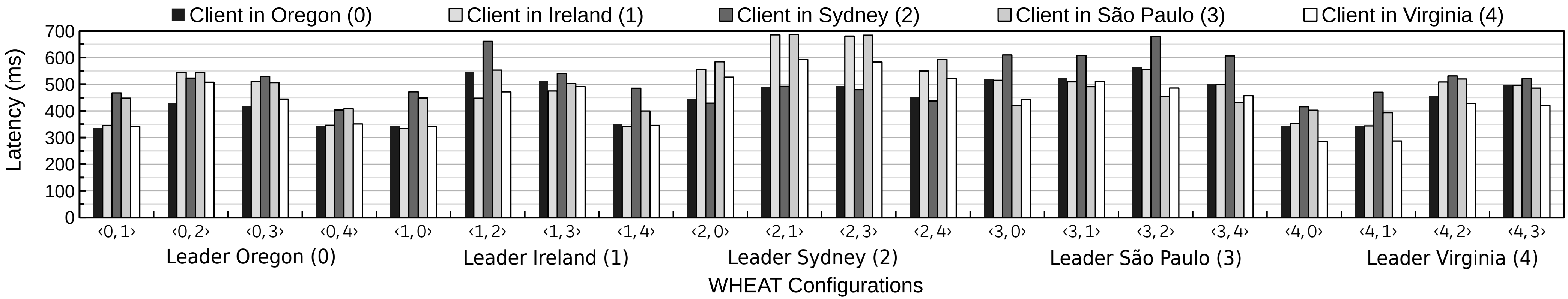
- **AWARE*** is implemented on top of WHEAT, which is based on BFT-SMaRt
- For evaluation, we use the **Amazon AWS cloud**, using EC2 instances of t2.micro type with 1 vCPU, 1 GB RAM and 8 GB SSD volume
- We select regions **Oregon, Ireland, Sydney, São Paulo** and **Virginia** for instances (1 client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



*Code of AWARE prototype is available at https://gitlab.sec.uni-passau.de/cb/aware

Clients' Observed Request Latency

Measured average request latency of 11th to 90th percentile across clients in different regions



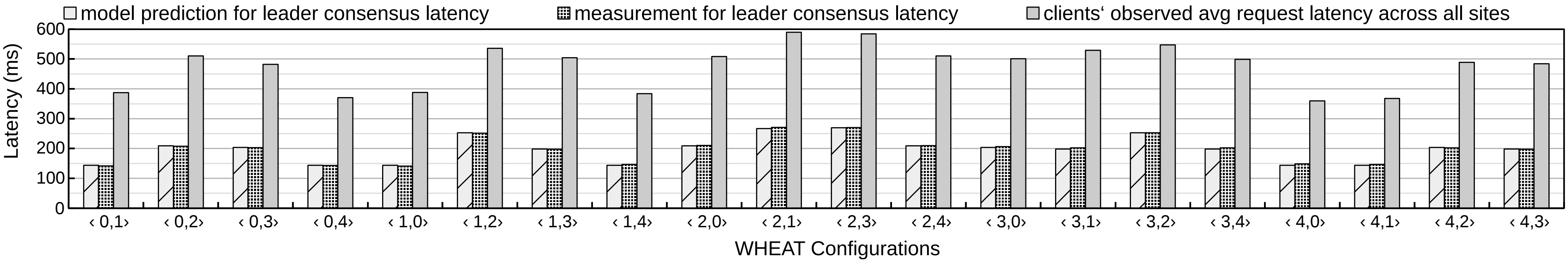
Configuration <L, R> means L is the leader and R is the other replica (besides the leader) with a voting weight of V_{max} = 2

Observations

- The best configuration <4,0> performs about 38.7% faster than the median <3,4>, 63.9% faster than the worst <2,1>
- Tuning voting weights can reduce latency (compare configurations with the same leader)
- Leader relocation may be necessary for achieving optimal consensus latency

Accuracy of Consensus Latency Prediction

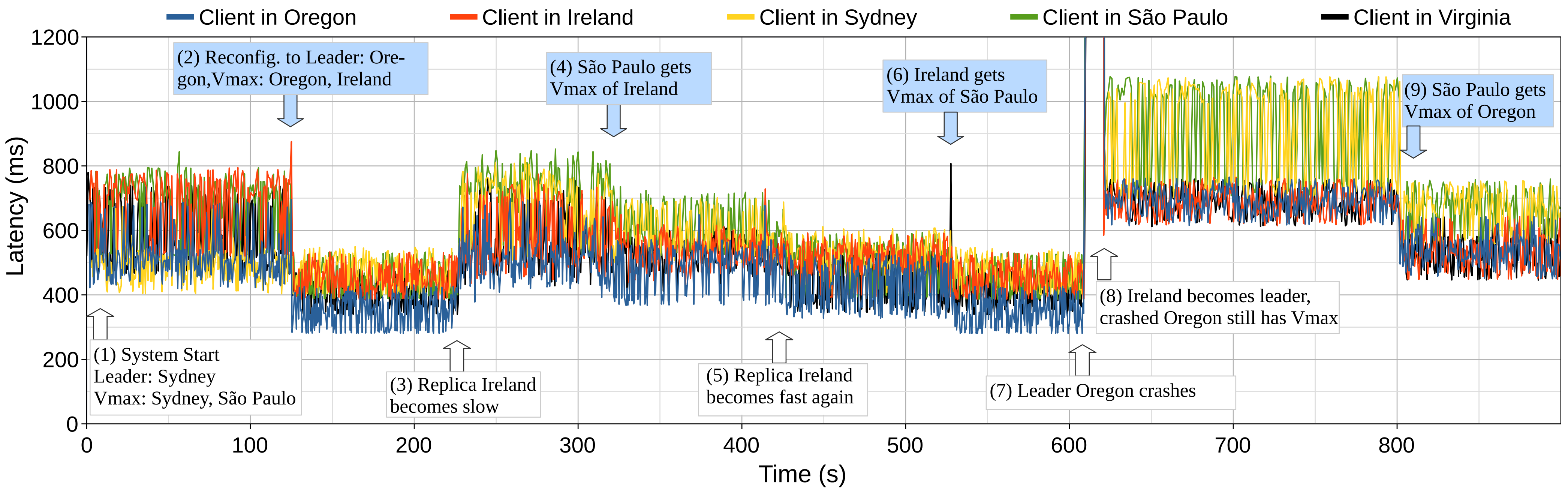
Comparison between predicted consensus latency, measured consensus latency and clients' observed average latency



Observations

- Accuracy of model prediction with respect to observed consensus latency of the leader: Predictions were off by 1.08% on average
- Strong correlation between series of model latency predictions and clients' observed request latency, $\rho(L^{MP}, L^{CR}) = 0.961$

Runtime behavior of AWARE



AWARE’s automated reactions

Summary of Observations

- **Ease of deployment**
  - AWARE provides the needed automation for finding an optimal configuration by tuning voting weights and/or relocating the leader
- **Adjusting 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
- **Compensating for faults**
  - Up to f replicas with high voting power become unavailable and hence restrict quorum variability
  - For non-malicious behavior, AWARE detects this and restores the availability of up to $f\,(V_{max} - V_{min})$ voting power by redistributing high voting weights

Conclusions

Conclusions

- World-spanning Byzantine consensus is getting practical and necessary with recent blockchain developments (e.g., Libra, Tendermint)

- AWARE enriches the idea of weighted replication
  - It provides the needed automation to adapt to changing environmental conditions → **adaptive weighted replication**

- Results show that the **potential for latency gains is substantial**
  - Best configuration performs about 38.7% faster on average in terms of observed latency across clients' sites than the median