U LISBOA | UNIVERSIDADE DE LISBOA

Adapting to Evolving Threats against BFT Systems with Weighted and Diverse Replication

Alysson Bessani

Ciências ULisboa

LASIGE

Byzantine Fault Tolerance Protocols

- **Performance**
  - The race horses: PBFT, Zyzzyva, Alyph, …
- **Robustness**
  - Slow but steady: Prime, Aardvark, RBFT, …
- **Resource efficiency**
  - Strong assumptions: MinBFT, CheapBFT, XFT, …
- **Scalability**
  - Blockchainers: HoneyBadger, FastBFT, SBFT, …

This talk: Adaptiveness

- **Changing/Unknown Network Conditions**
  - Weighted Replication applied to BFT
  - Autonomic weight assignment

- **Threat Landscape (attacks and vulnerabilities)**
  - Diversity for fault independence
  - Monitoring new vulnerabilities and changing the replica set

BFT-SMaRt

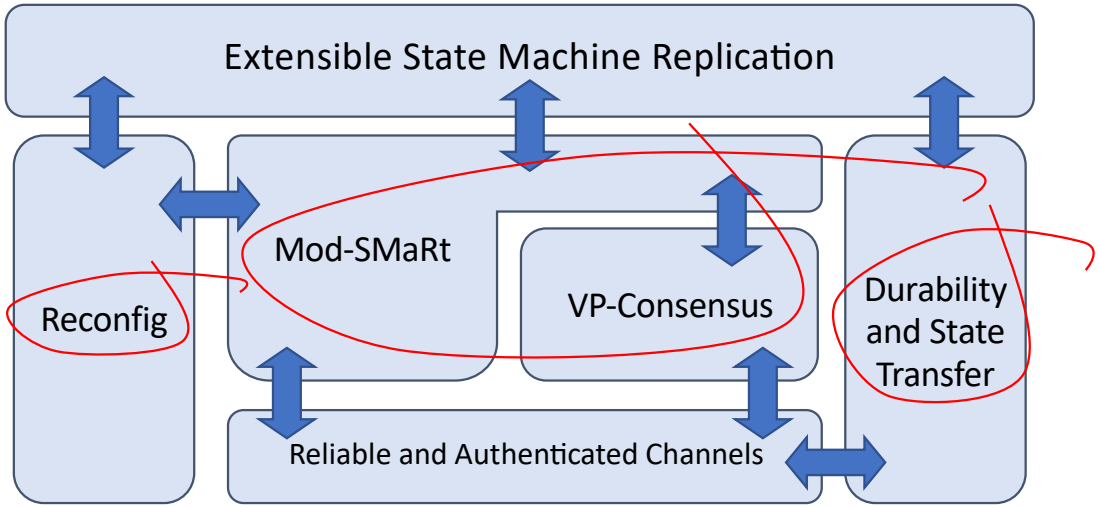
BACKGROUND

Sousa, Bessani. From Byzantine Consensus to BFT State Machine Replication: A Latency-optimal Transformation. EDCC'12.

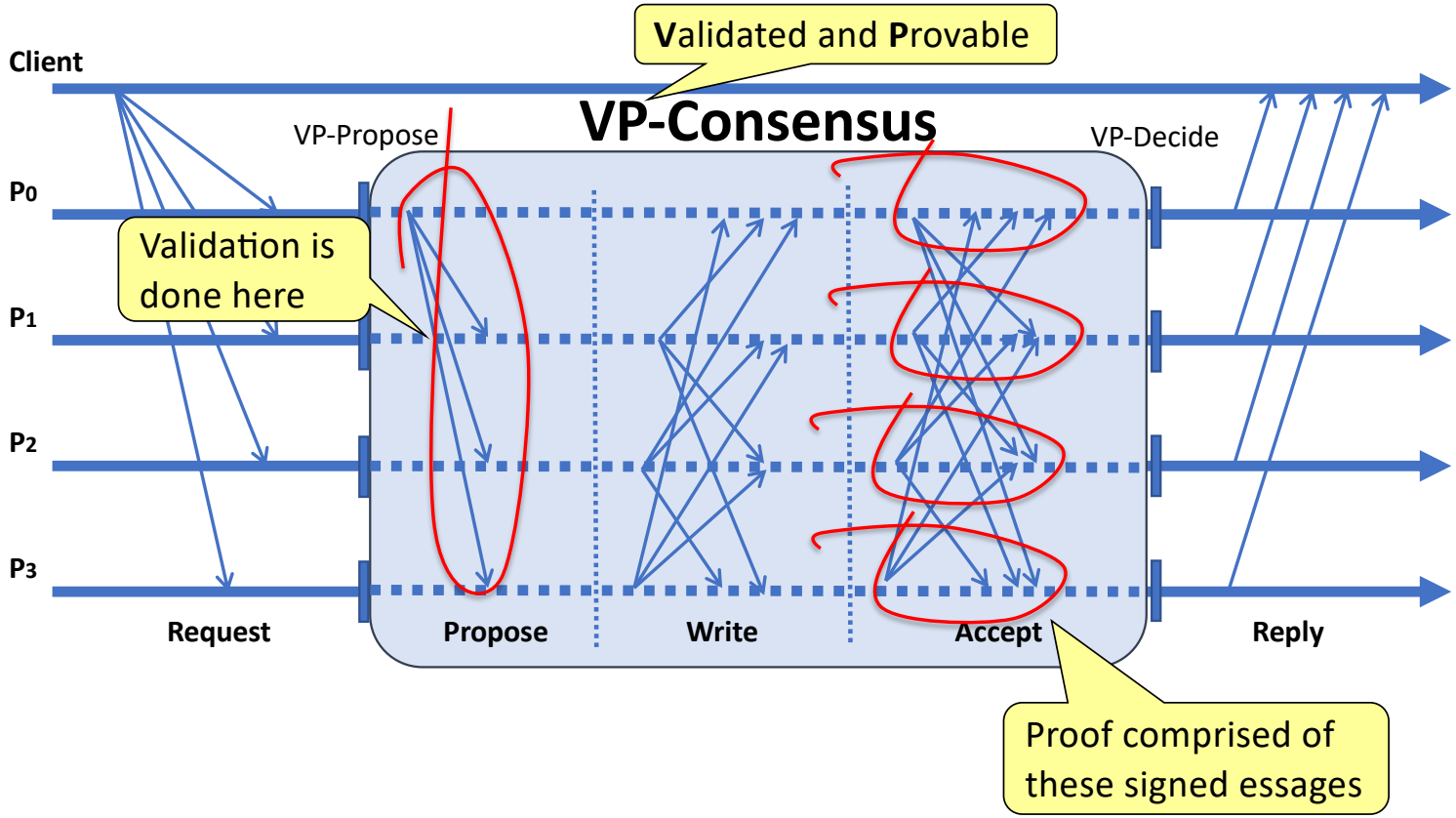
Bessani, Sousa, Alchieri. State Machine Replication for the Masses with BFT-SMaRt. IEEE/IFIP DSN'14.

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

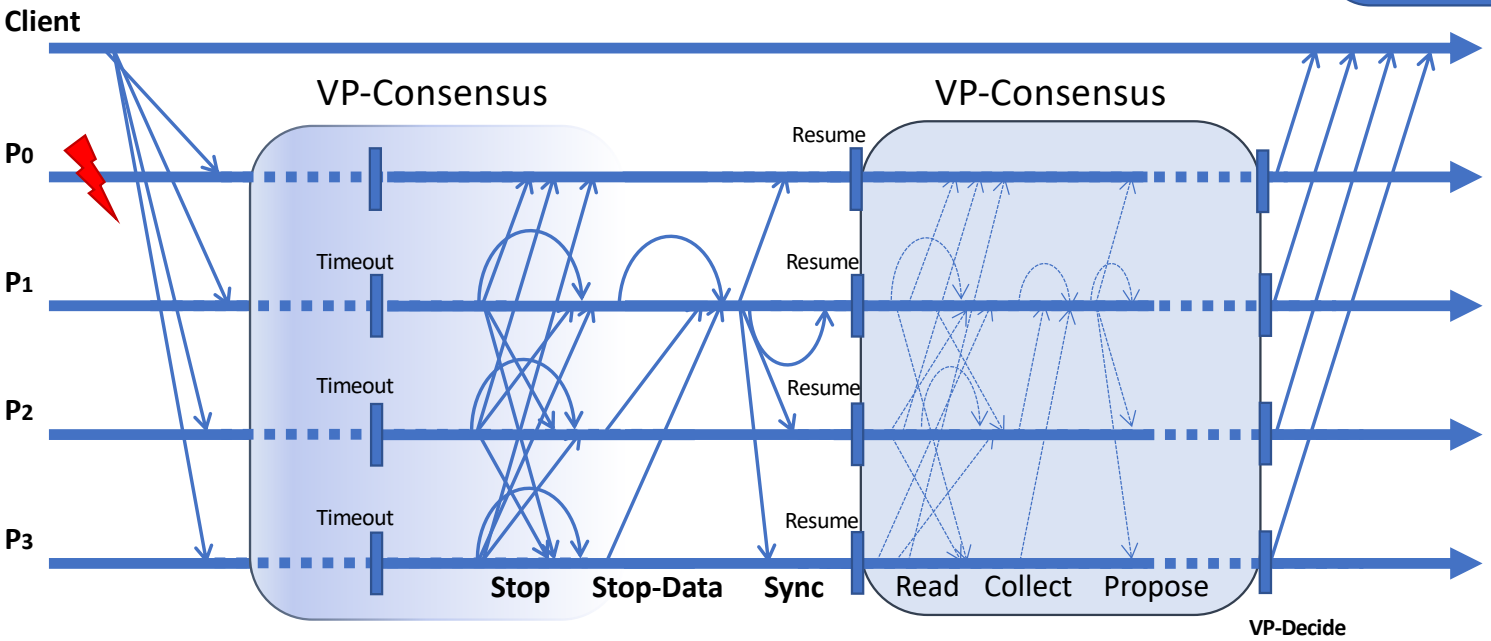


Mod-SMaRt: Normal Phase



Mod-SMaRt: Synchronization Phase

IMPORTANT:
It looks like PBFT, but it is not PBFT ☺



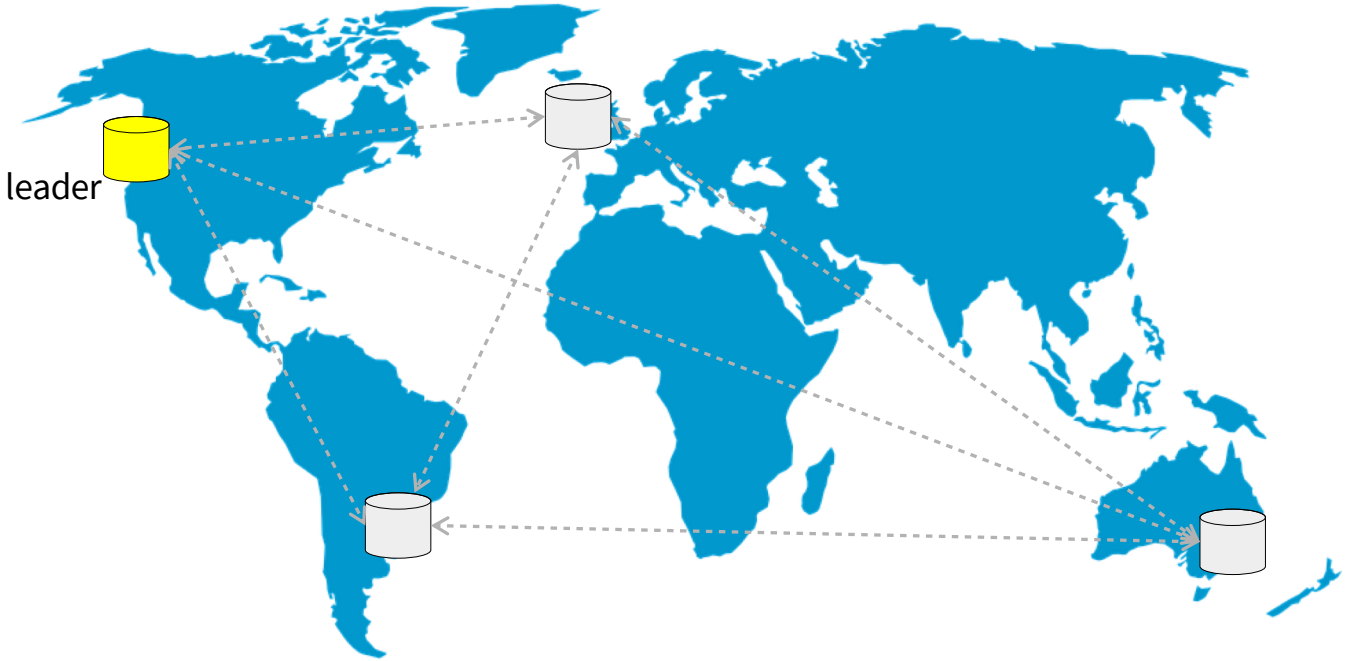
Adaptivity to Network Conditions

PART I

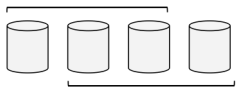
Sousa, Bessani. Separating the WHEAT from the Chaff: An Empirical Design for Geo-replicated State Machines. IEEE SRDS'15.

Berger, Reiser, Sousa, Bessani. Resilient Wide-area Byzantine Consensus using Adaptive Weighted Replication. IEEE SRDS'19.

Classical BFT Replication



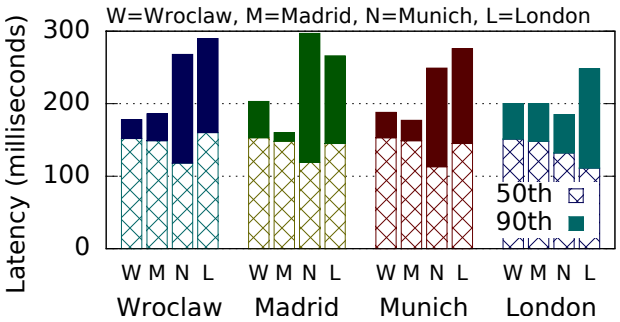
N=4, f=1

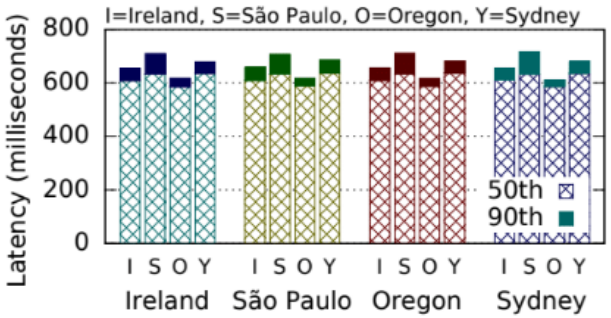


Egalitarian quorums,
Any 3 out-of 4 replica

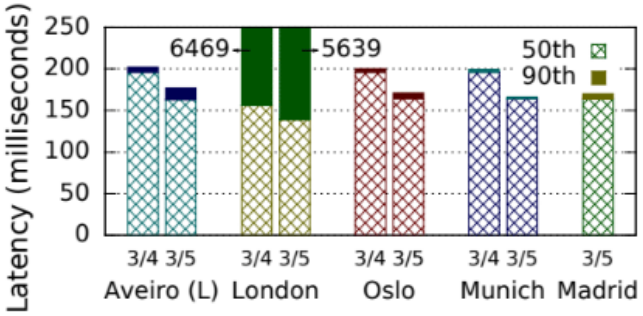
Experimental study conducted with BFT-SMaRt on Planetlab and Amazon EC2

Leader location





Quorum size



Summary:

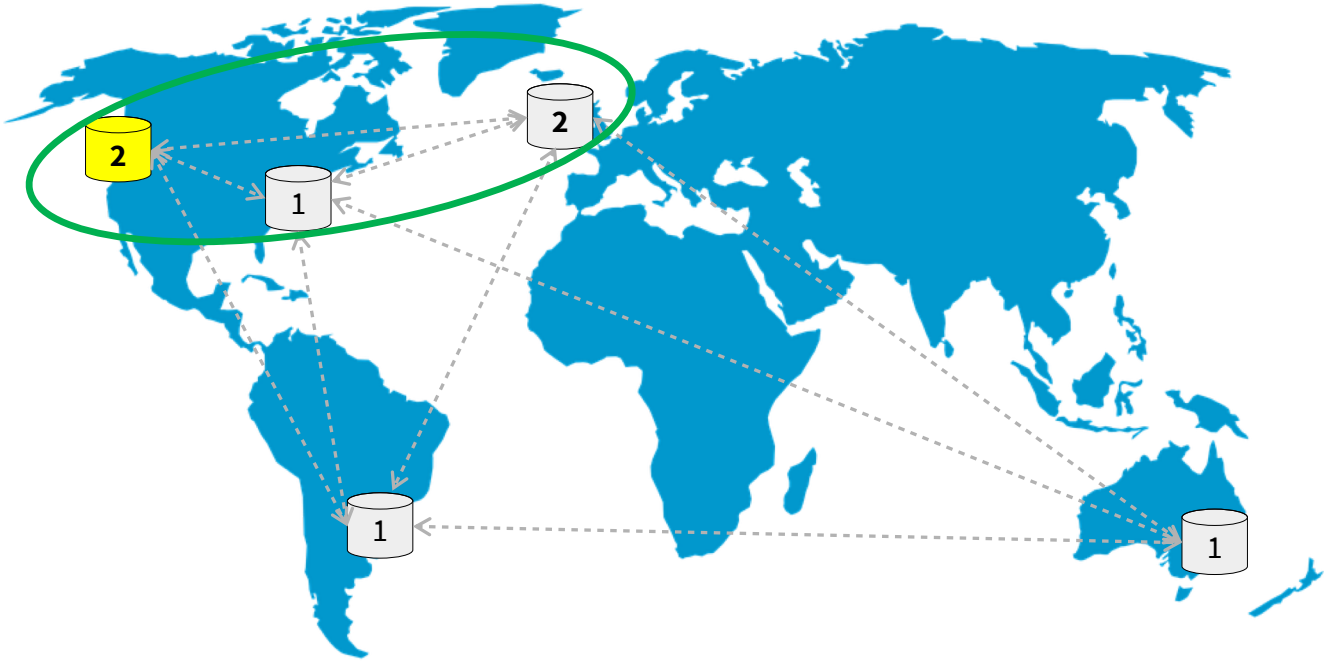
- Leader in the best-connected site yields better results than employing a rotating or multiple leader(s) strategy
- Smaller quorums create opportunities for improving latency

WHEAT: WeigHt-Enabled Active replicaTion

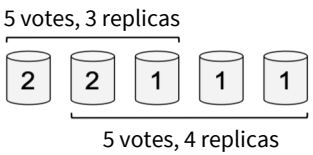
- Use optimizations that lead to significant latency reduction:
  - Single leader in the best-connected site
  - Tentative executions (from PBFT)
  - Employs smaller quorums (weighted replication)

- Weighted replication: safe voting assignment scheme for SMR
  - Uses Δ 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, **Δ=1 (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_{max} > 1 votes, without violating *f*

CFT mode

$$n = 2f + 1 + \Delta$$

$$N_v = \sum V_i = 2F_v + 1$$

$$Q_v = F_v + 1$$

$$u = f$$

BFT mode

$$n = 3f + 1 + \Delta$$

$$N_v = \sum V_i = 3F_v + 1$$

$$Q_v = 2F_v + 1$$

$$u = 2f$$

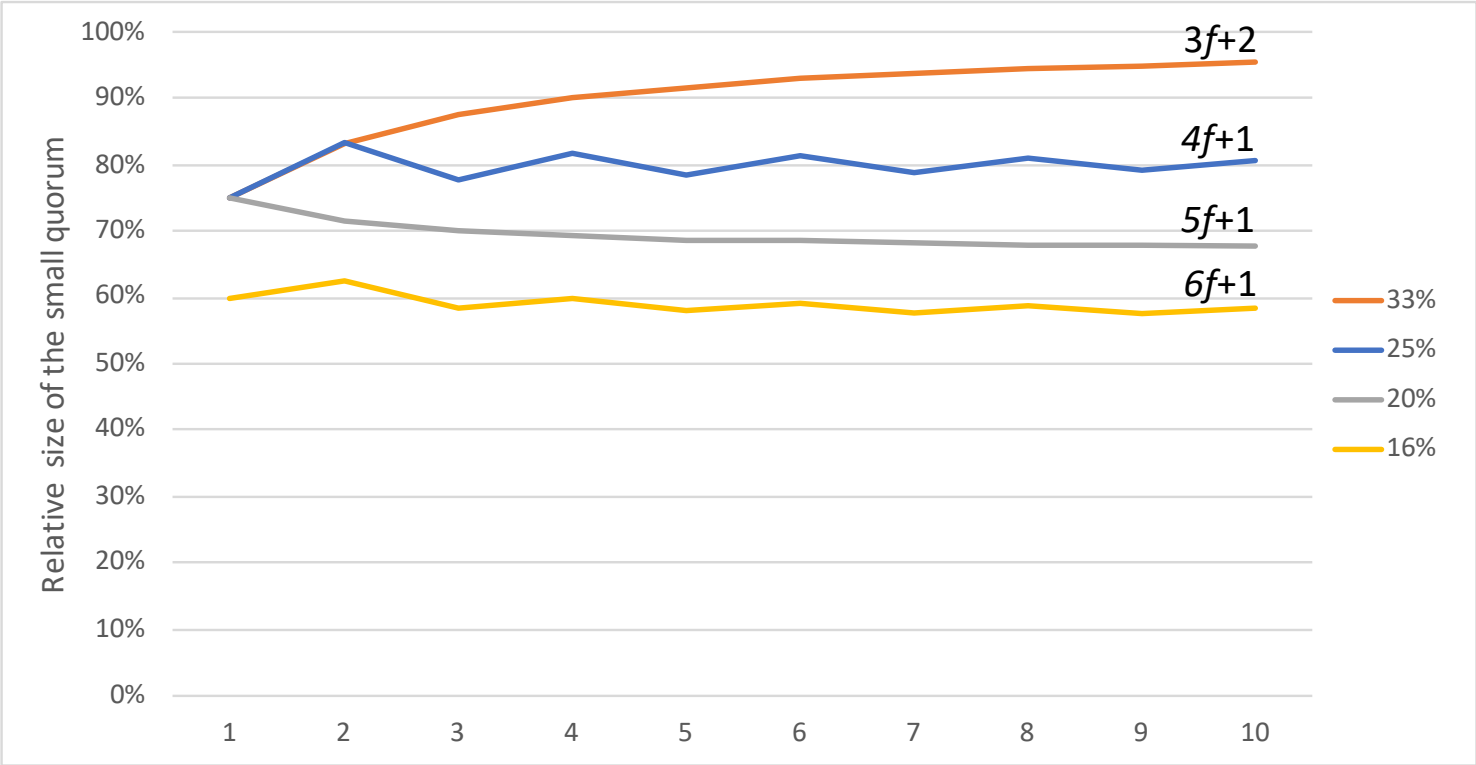
$$F_v = \Delta + f$$

$$V_{max} = \frac{\Delta + f}{f} = 1 + \frac{\Delta}{f}$$

Input:
***f* and Δ**

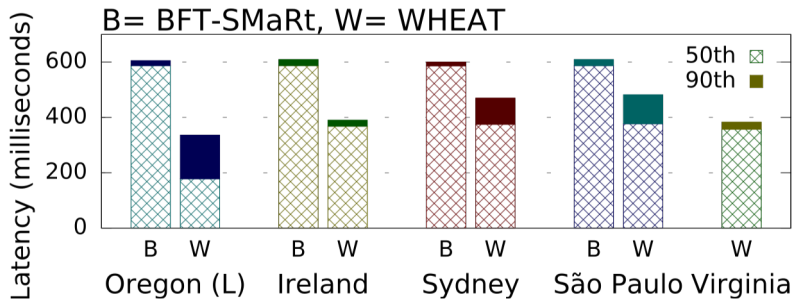
Output:
***u* and V_{max}**

Size of fast quorums with different *f* and Δ



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

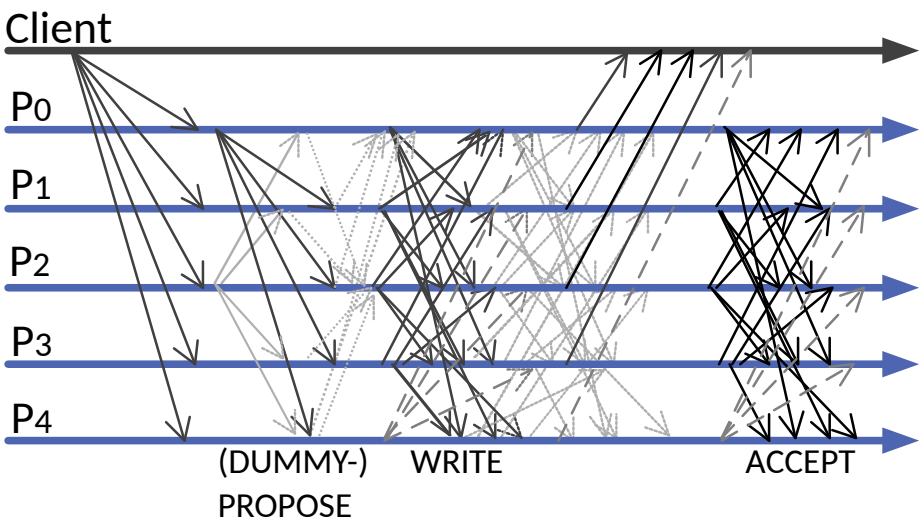
- **Self-Monitoring**
  - AWARE uses reliable self-monitoring as decision-making basis for adapting replicas‘ voting weights and leader position at runtime

- **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

	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

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

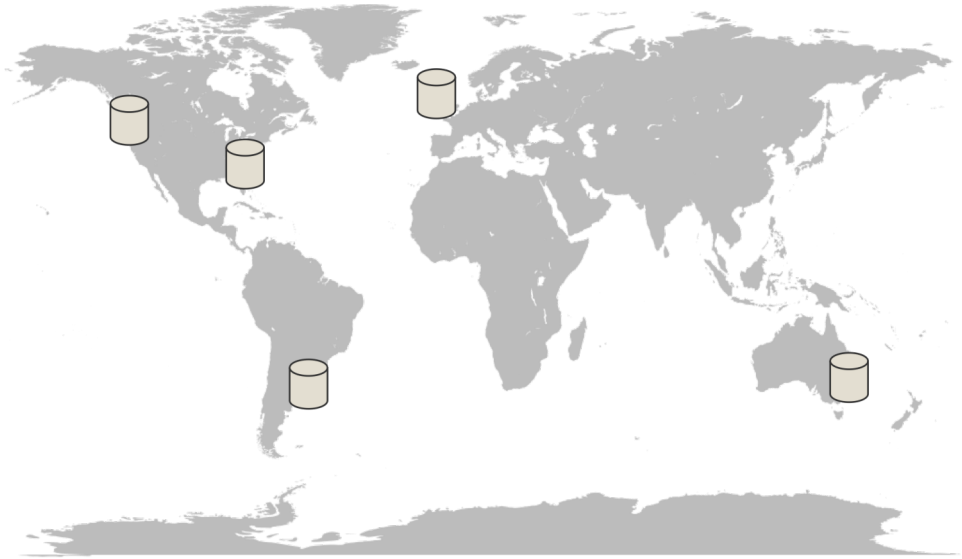
$$\langle \hat{l}, \hat{W} \rangle = \underset{W \in \mathfrak{W}, l \in \mathfrak{L}}{\arg\min} \; PredictLatency(l, W, \hat{M}^P, \hat{M}^W)$$

- All correct replicas reach the same, optimal weight distribution and invoke a reconfiguration in the system

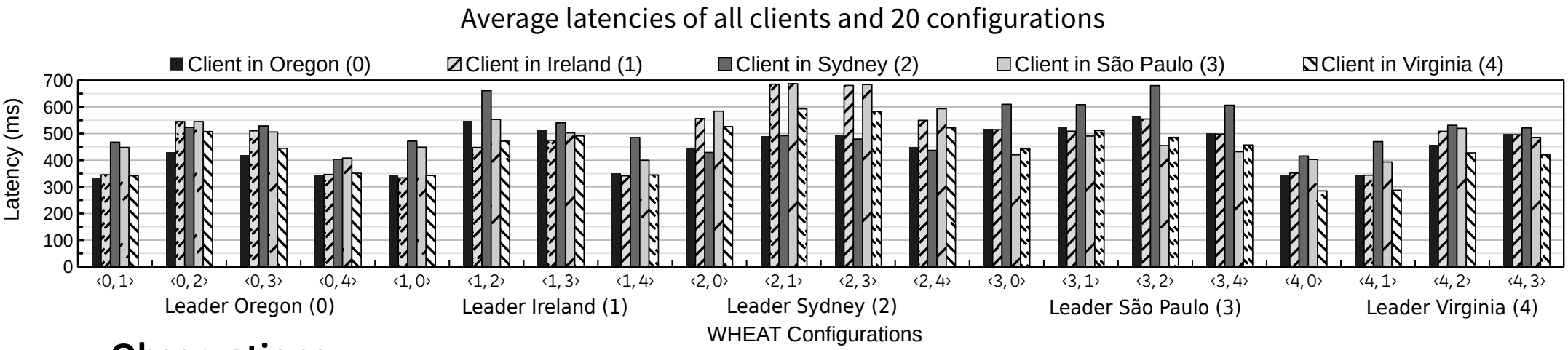
Evaluation of WHEAT and AWARE

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



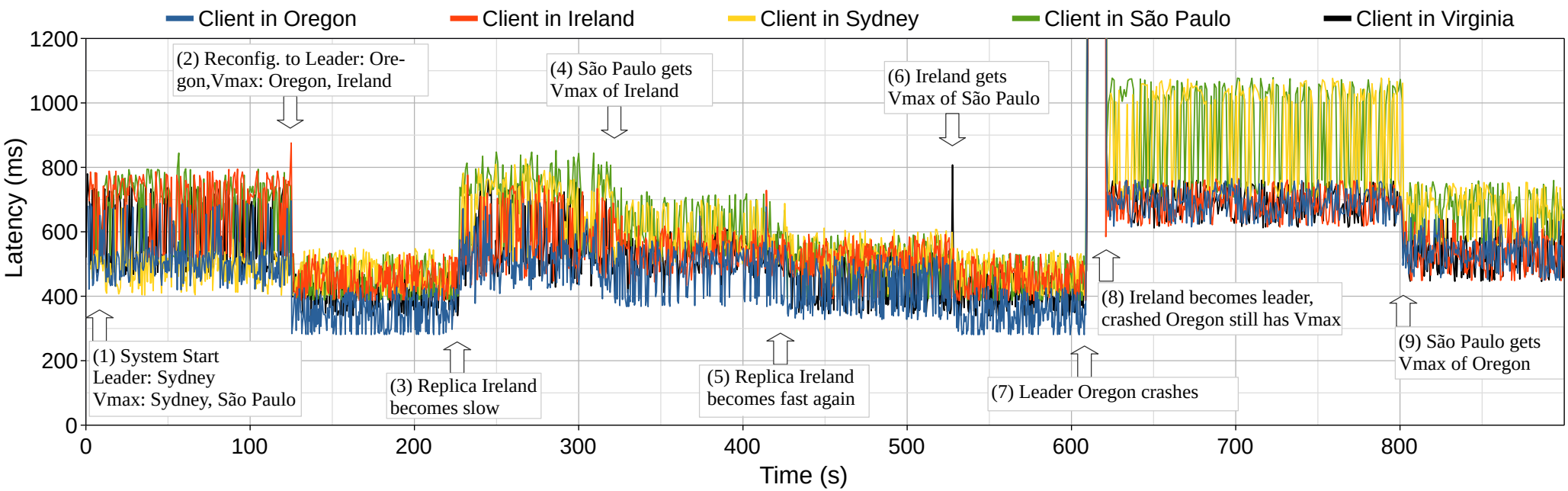
Clients' Request Latency



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

Runtime Behavior of AWARE



Adapting to Evolving Threats with Diverse Replication

PART II

Garcia, Bessani, Neves. Lazarus: Automatic Management of Diversity for BFT Systems. ACM/IFIP Middleware'19.

Problem

“We assume independent node failures.”

“We assume independent node failures.”

“We assume independent node failures.”

“We assume independent node failures.”

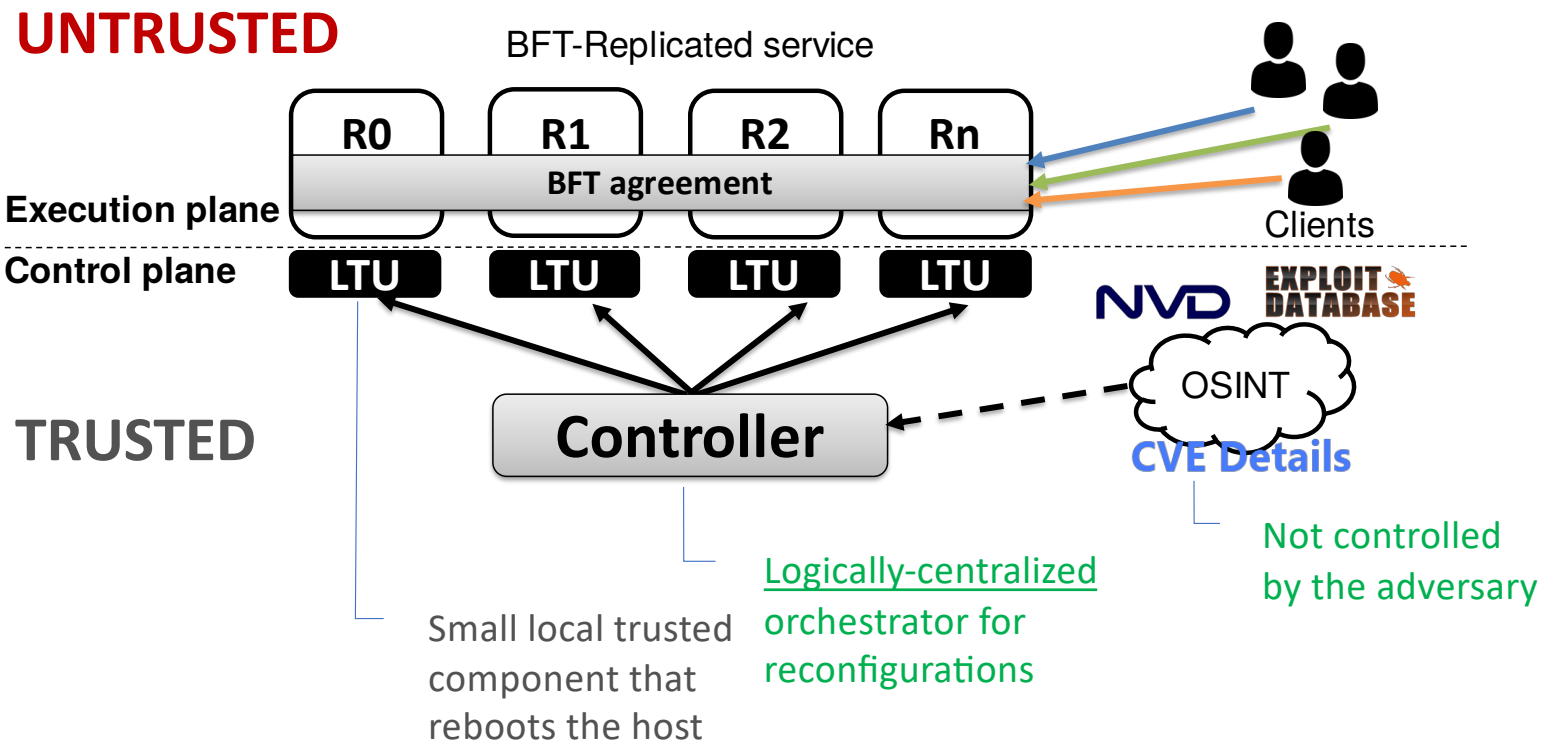
“We assume independent node failures.”

LAZARUS

- LAZARUS is a response to the long-standing open problem of managing the diversity of a BFT system to make it resilient to **malicious (intelligent) adversaries**

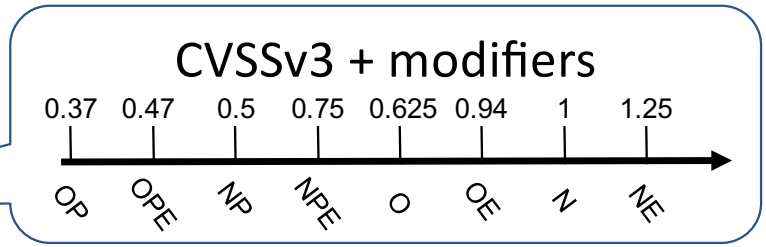
- LAZARUS objective is to address two types of threats:
  - **Newly found vulnerabilities** affecting one or more active replicas
  - **Zero-day vulnerabilities** affecting multiple running replicas

LAZARUS Overview



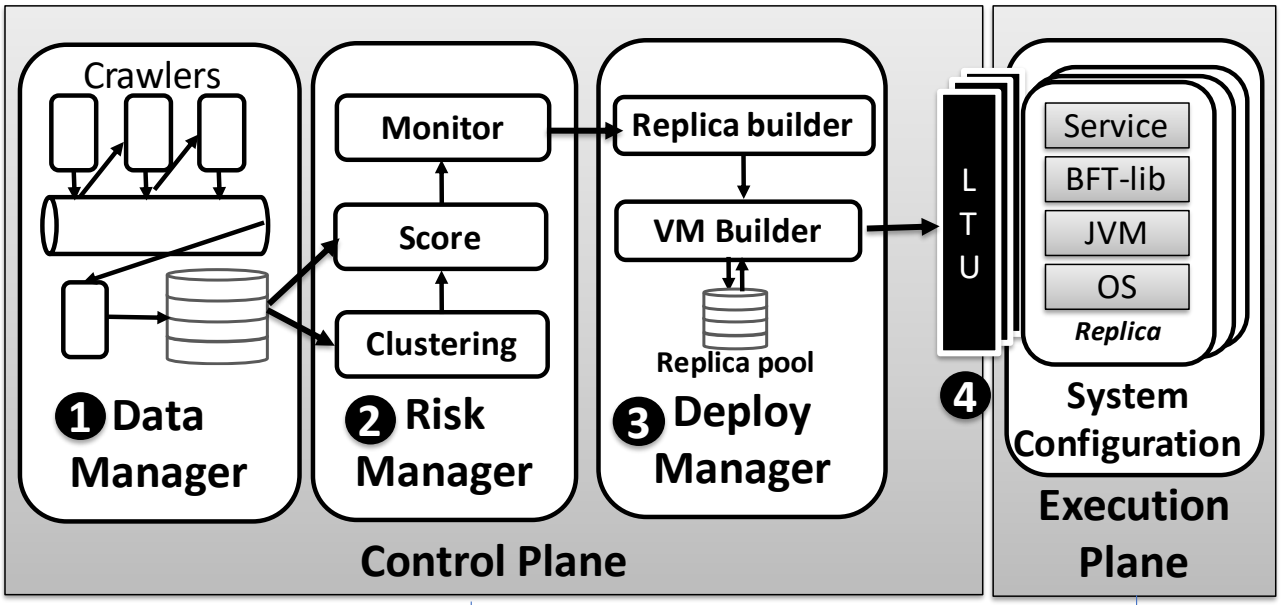
Diversity-aware Reconfigurations

1. Find common vulnerabilities
   - Fetch OSINT from NVD, ExploitDB, vendor sites, etc.
   - For each vulnerability, discover
     - exploit and patch information
     - which replicas in the pool are affected
2. Measure vulnerability severity



3. Measure configuration risk
   - Sum the severity of vulnerabilities affecting each pair of replicas from a config.
4. Select next configuration
   - The one with less risk that requires less changes

LAZARUS Implementation



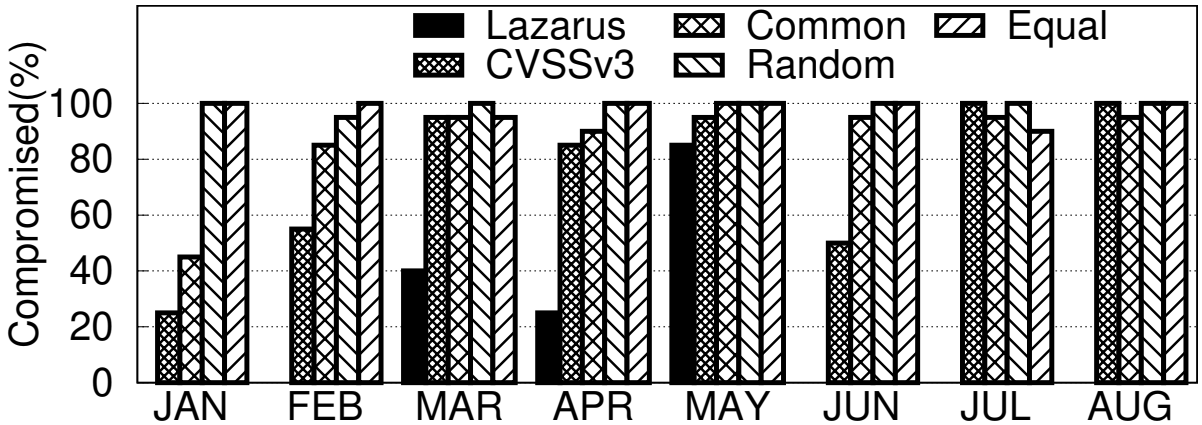
Centralized controller
(can be made BFT)

Replicas packed as
VirtualBox VMs

Effectiveness and Performance of LAZARUS

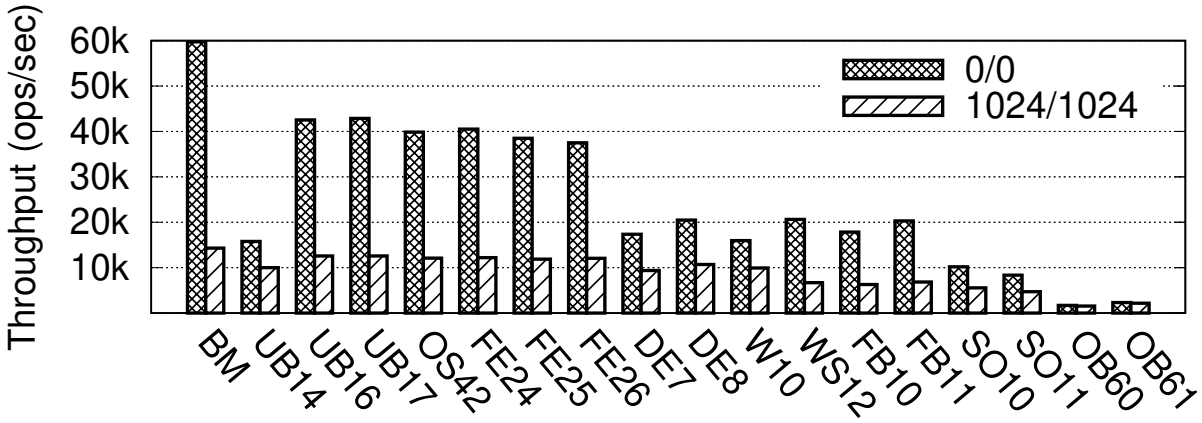
LAZARUS Effectiveness

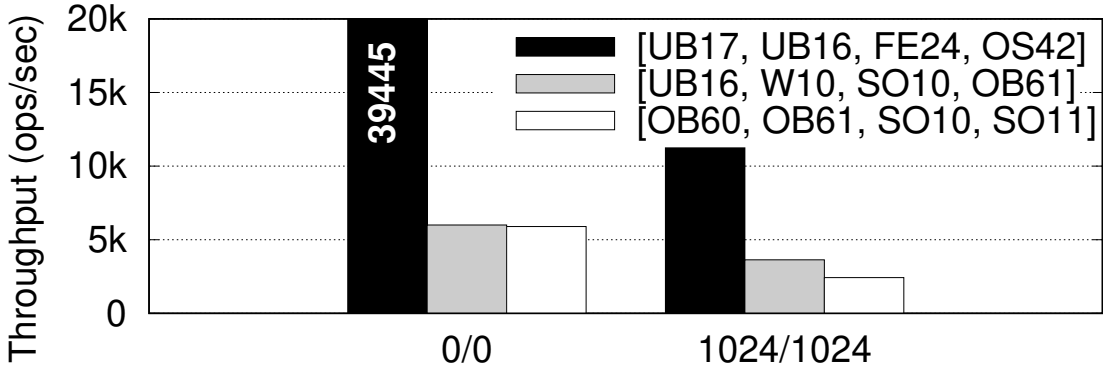
- Using a knowledge base of OS vulnerabilities (NVD, ExpoitDB, etc.) from 2014 to the month before the execution (in 2018)
- We consider a run compromised if a published vulnerability affects more than one host from the configuration
- Each strategy was executed a thousand times



Diversity and Performance

ID	Name	#Cores	Memory
UB14	Ubuntu 14.04	4	15GB
UB16	Ubuntu 16.04	4	15GB
UB17	Ubuntu 17.04	4	15GB
OS42	OpenSuse 42.1	4	15GB
FE24	Fedora 24	4	15GB
FE25	Fedora 25	4	15GB
FE26	Fedora 26	4	15GB
DE7	Debian 7	4	15GB
DE8	Debian 8	4	15GB
W10	Windows 10	4	1GB
WS12	Win. Server 2012	4	1GB
FB10	FreeBSD 10	4	1GB
FB11	FreeBSD 11	4	1GB
SO10	Solaris 10	1	1GB
SO11	Solaris 11	1	1GB
OB60	OpenBSD 6.0	1	1GB
OB61	OpenBSD 6.1	1	1GB





Conclusions

- Distributed trust → geo-distributed systems
- Geo-distributed on the wild → heterogeneity
- Heterogeneity (i.e., diversity) is also important for fault independence

- Our works are “initial steps” on dealing with heterogeneity:
  - Weighted replication is a powerful tool, and **other schemes are possible**
  - Self-adaptiveness of protocols will become mandatory in operation
  - BFT diversity is a hard problem that need to be addressed

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.**
  - **Bessani, Sousa, Alchieri.** ***State Machine Replication for the Masses with BFT-SMaRt.*** **IEEE/IFIP DSN'14.**
  - **Sousa, Bessani.** ***Separating the WHEAT from the Chaff: An Empirical Design for Geo-replicated State Machines.*** **IEEE SRDS'15.**
  - **Berger, Reiser, Sousa, Bessani.** ***Resilient Wide-area Byzantine Consensus using Adaptive Weighted Replication.*** **IEEE SRDS'19.**
  - **Garcia, Bessani, Neves.** ***Lazarus: Automatic Management of Diversity for BFT Systems.*** **ACM/IFIP Middleware'19.**

Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA EDUCAÇÃO E CIÊNCIA
European Commission