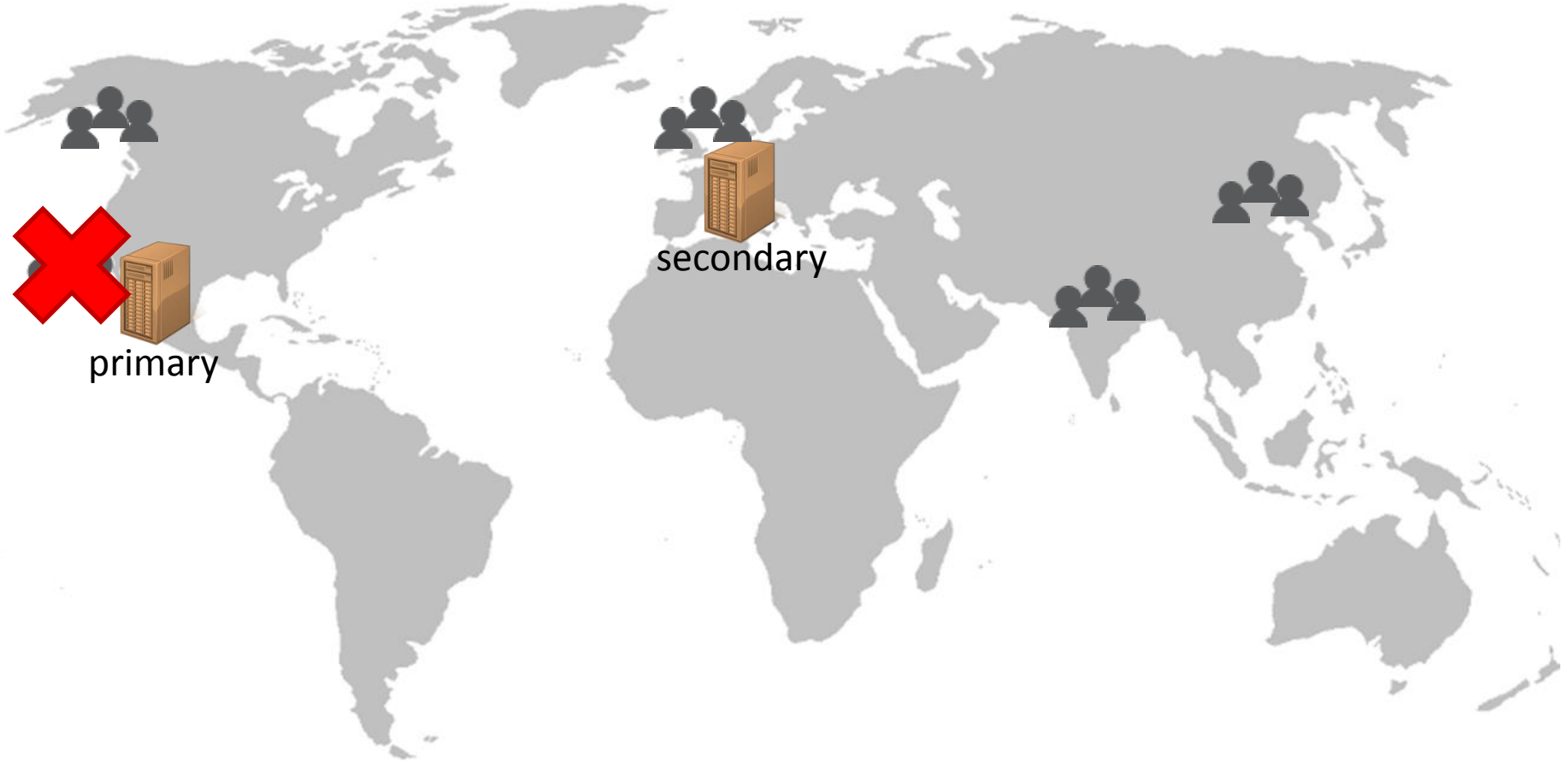
A Self-Configurable Geo-Replicated Cloud Storage System

Masoud Saeida Ardekani, *Inria & UPMC*
Douglas B. Terry, [redacted]

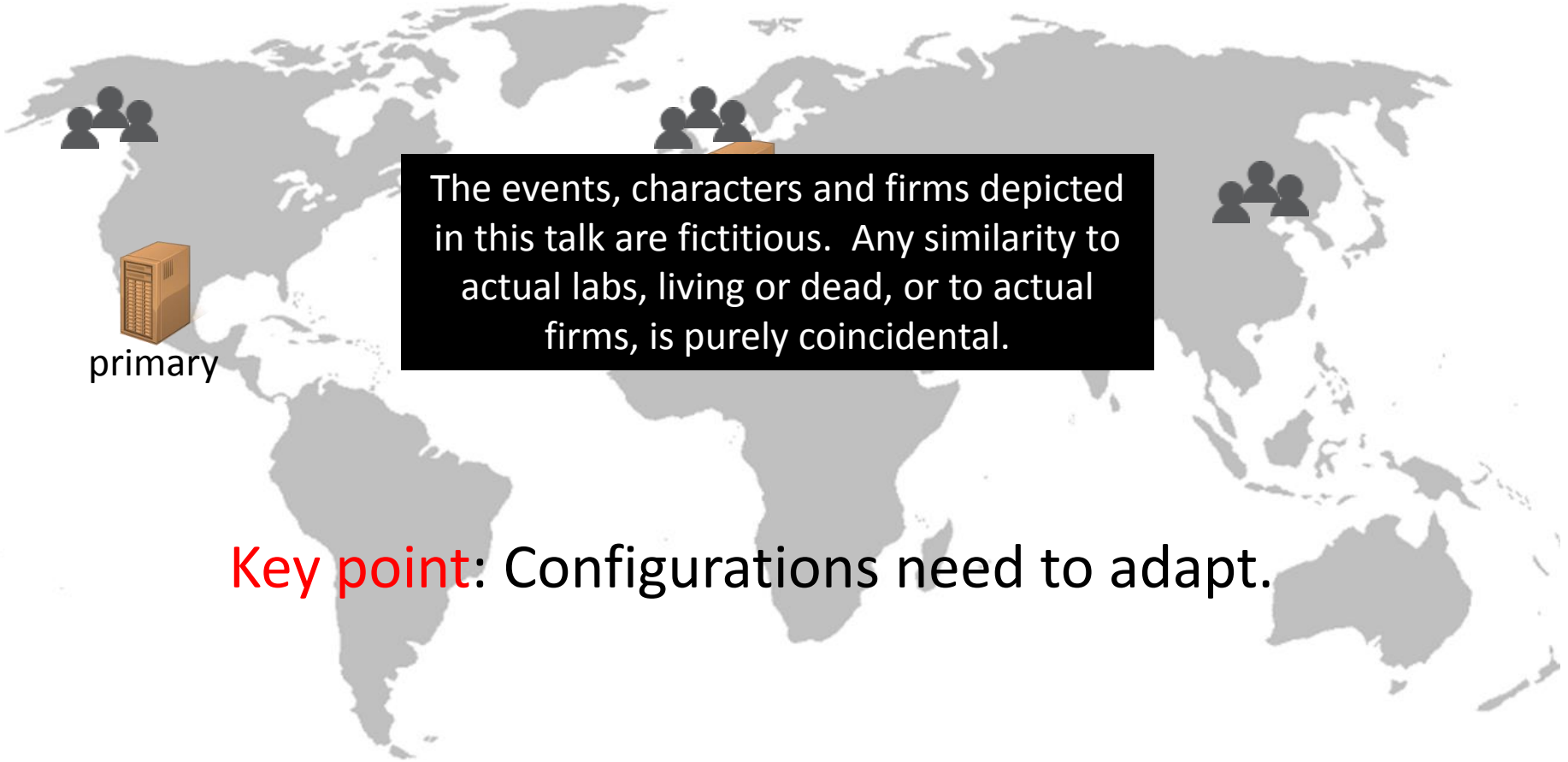
Special thanks to:
Marcos K. Aguilera, Mahesh Balakrishnan, Ramakrishna Kotla

OSDI 2014

Scenario



Scenario



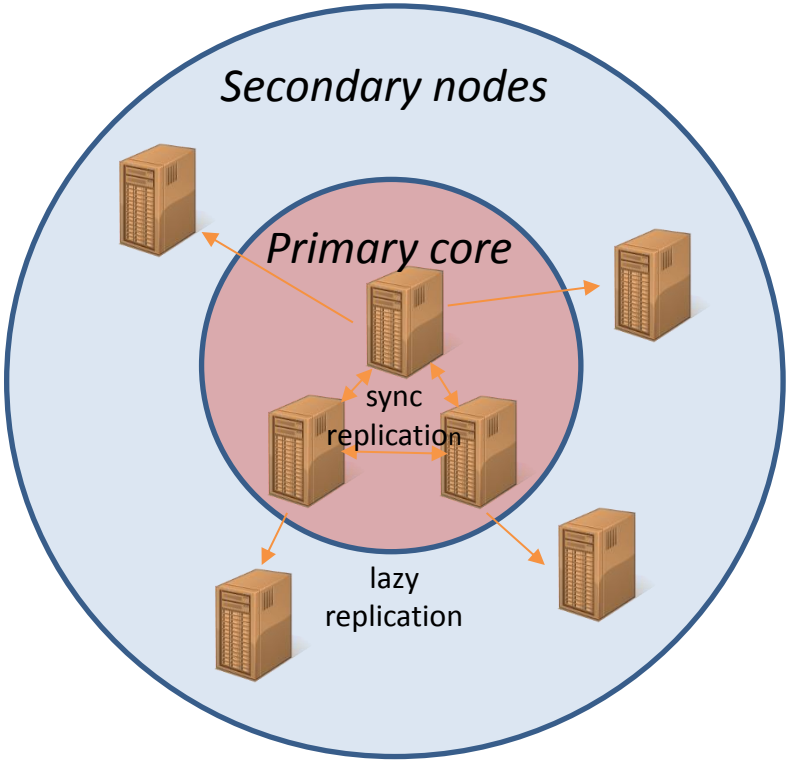
The events, characters and firms depicted in this talk are fictitious. Any similarity to actual labs, living or dead, or to actual firms, is purely coincidental.

Key point: Configurations need to adapt.

Configuration Service

- Selects new configuration to improve overall utility delivered to clients
- Installs new configuration *while* clients continue to read and write data

Storage System Model



Based on Pileus [SOSP 2013]

Configuration =

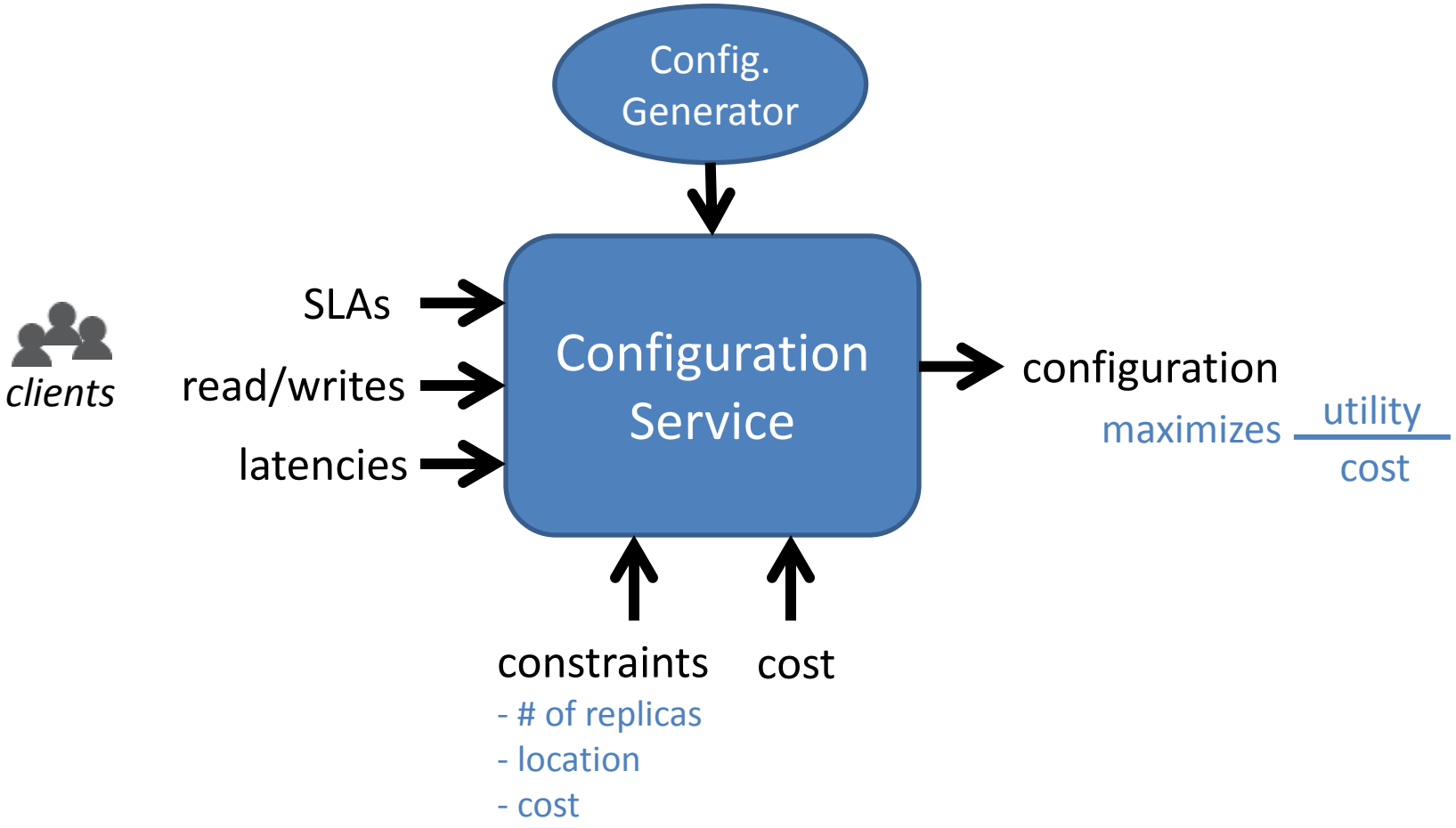
1. Location of primary replicas
2. Location of secondary replicas
3. Synchronization period between primary and secondary replicas

Consistency-based SLAs

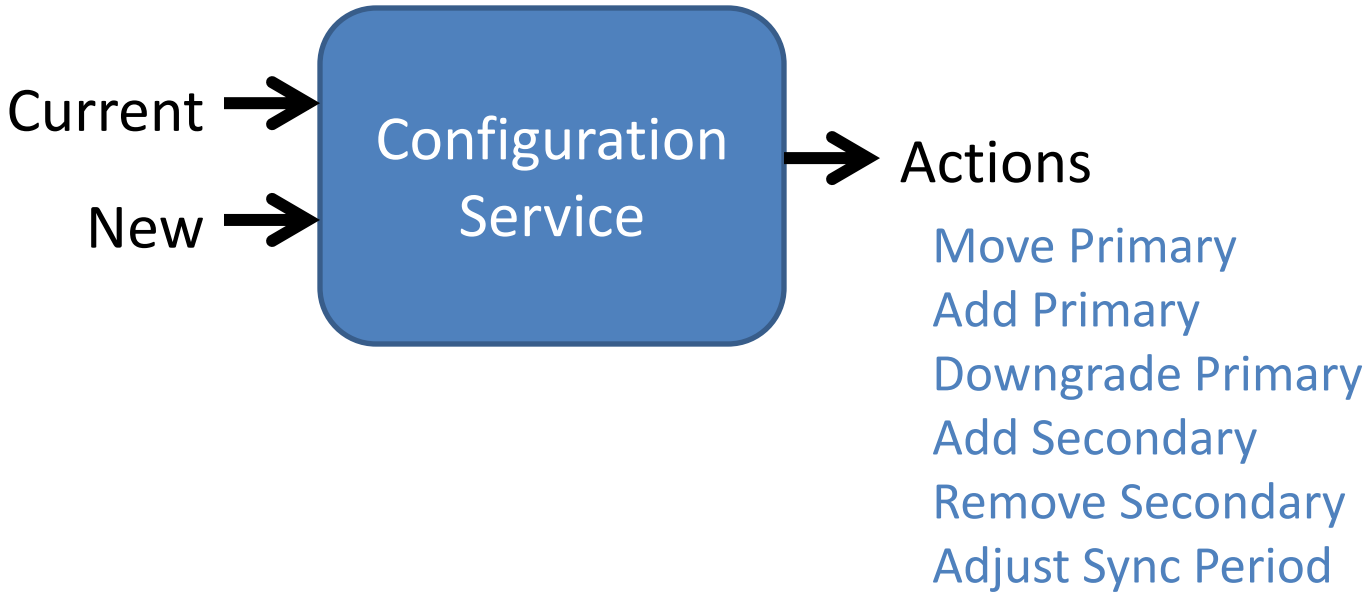
- Applications declare acceptable consistency/latency pairs and utility
- E.g. shopping cart

	consistency	latency	utility
1.	strong	300 ms.	1.0
2.	read my writes	300 ms.	0.5
3.	eventual	300 ms.	0.1

Selecting a Configuration



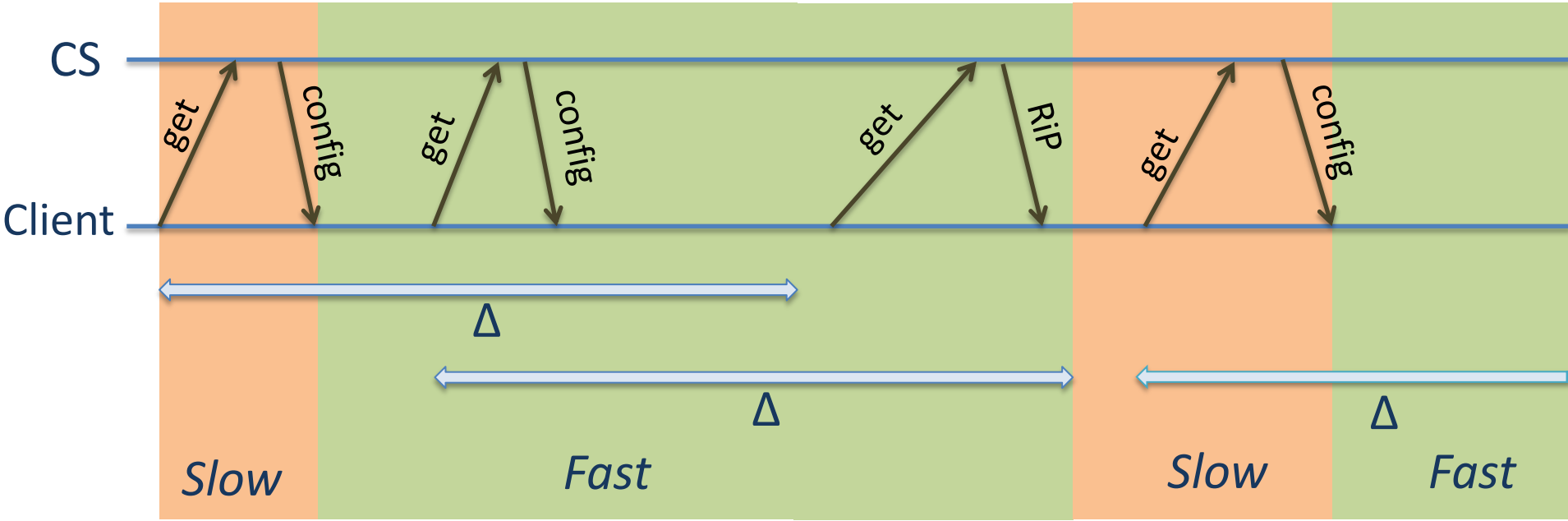
Installing a New Configuration



Example: Move Primary

1. Set Reconfiguration-in-Progress (RiP) flag
2. Wait Δ seconds
3. Add new primary to write-only replicas
4. Clear RiP flag
5. Sync new primary from old
6. Set RiP flag
7. Wait Δ seconds
8. Write new configuration
9. Clear RiP flag

Leasing Configurations



Slow mode = unsure of current configuration

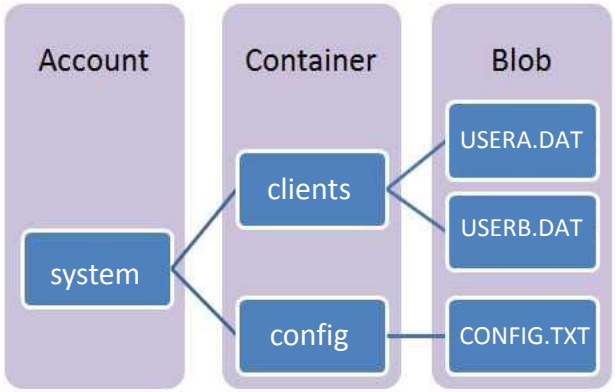
Fast mode = hold lease on configuration for Δ seconds

Client Operations

	Fast	Slow
Read	Read from best replica (ala Pileus)	Do speculative read then check configuration
Write	Write to all primaries	Lock configuration, then write

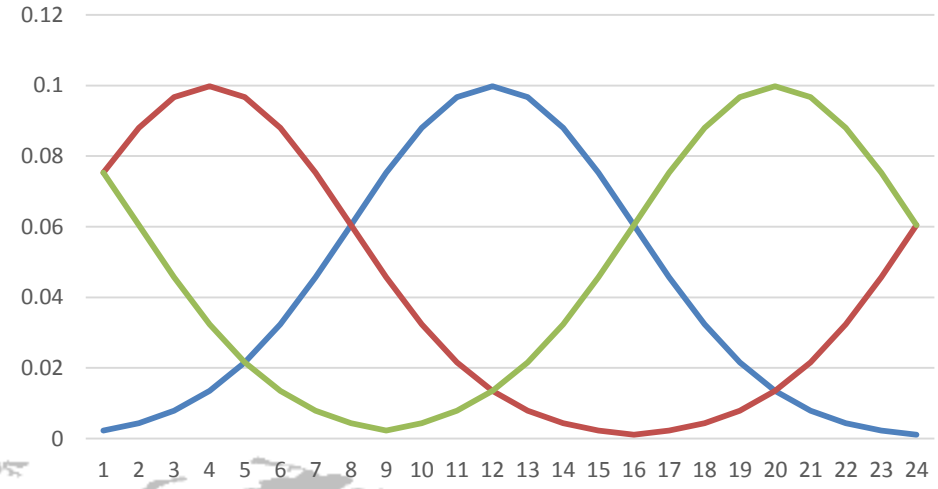
Implementation Notes

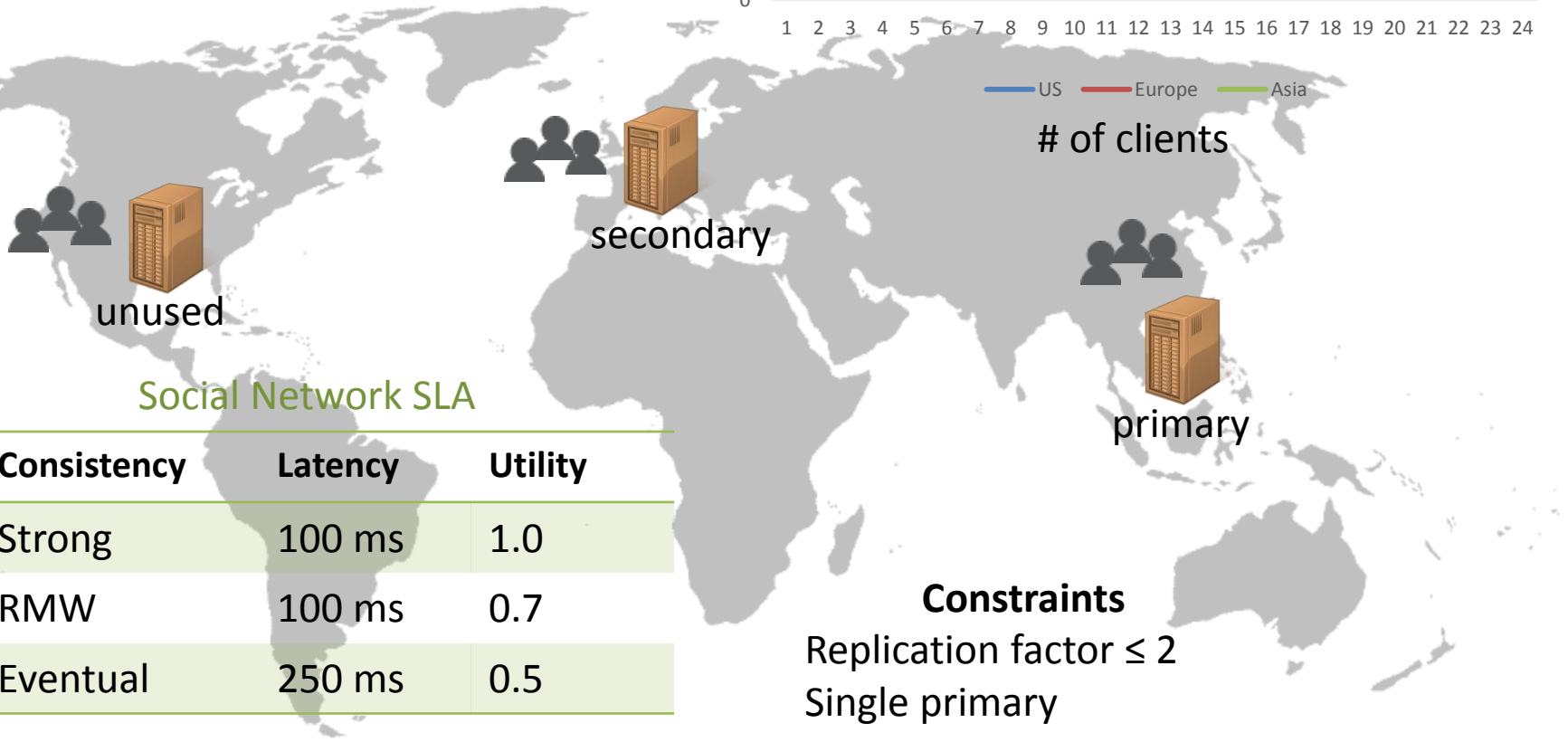
- Built on Azure Blob Store
- C.S. stores configuration blob
  - read periodically by clients
  - RiP flag stored as metadata
- Clients periodically write SLAs, read/write ratios, and latencies to blobs
  - read by C.S. during reconfiguration
- C.S. can run intermittently



Evaluation Setup

YCSB [SoCC'10]
Workload B (95% Read, 5% Write)
1000 Objects (1KB each)



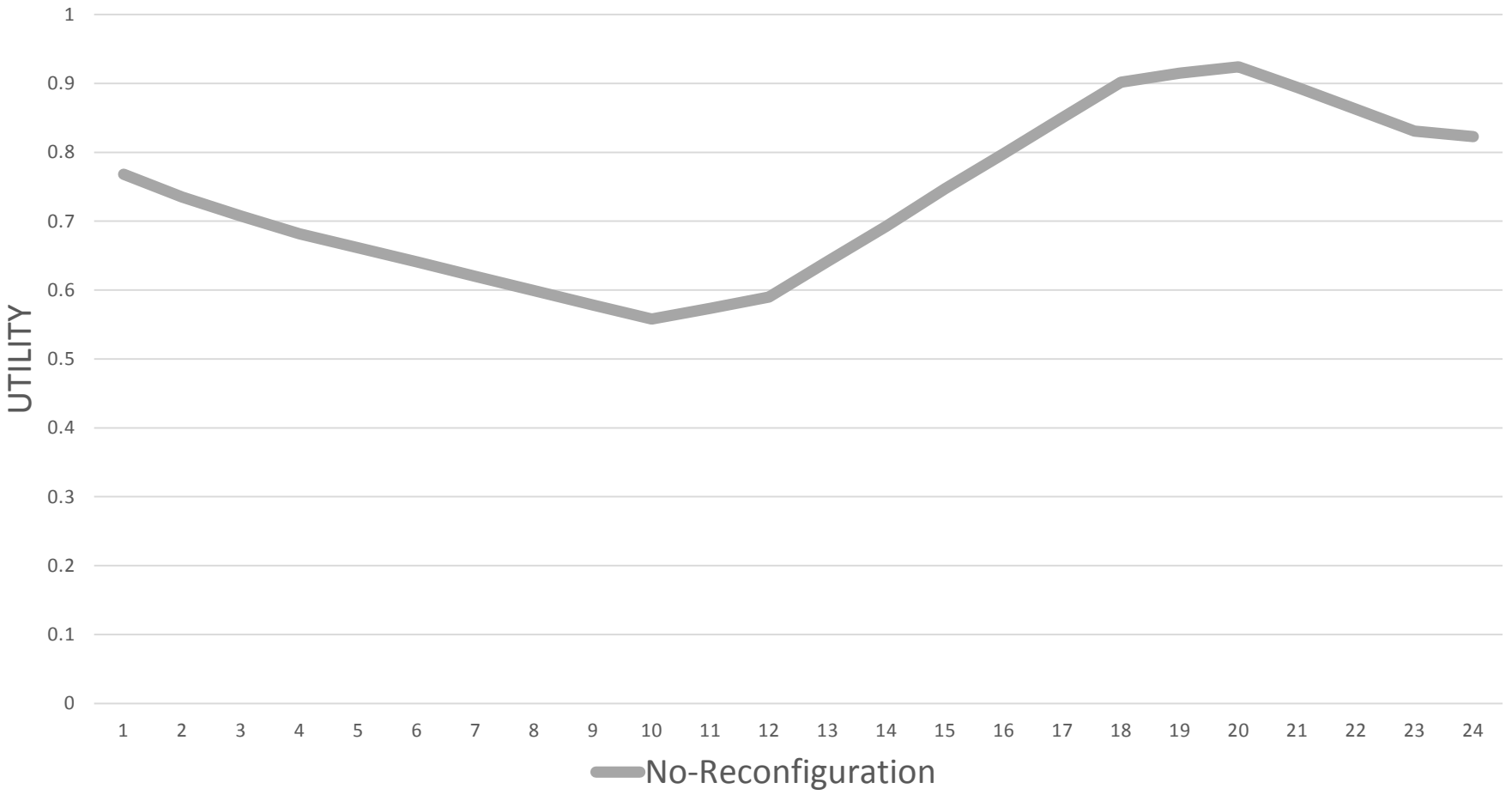


Social Network SLA

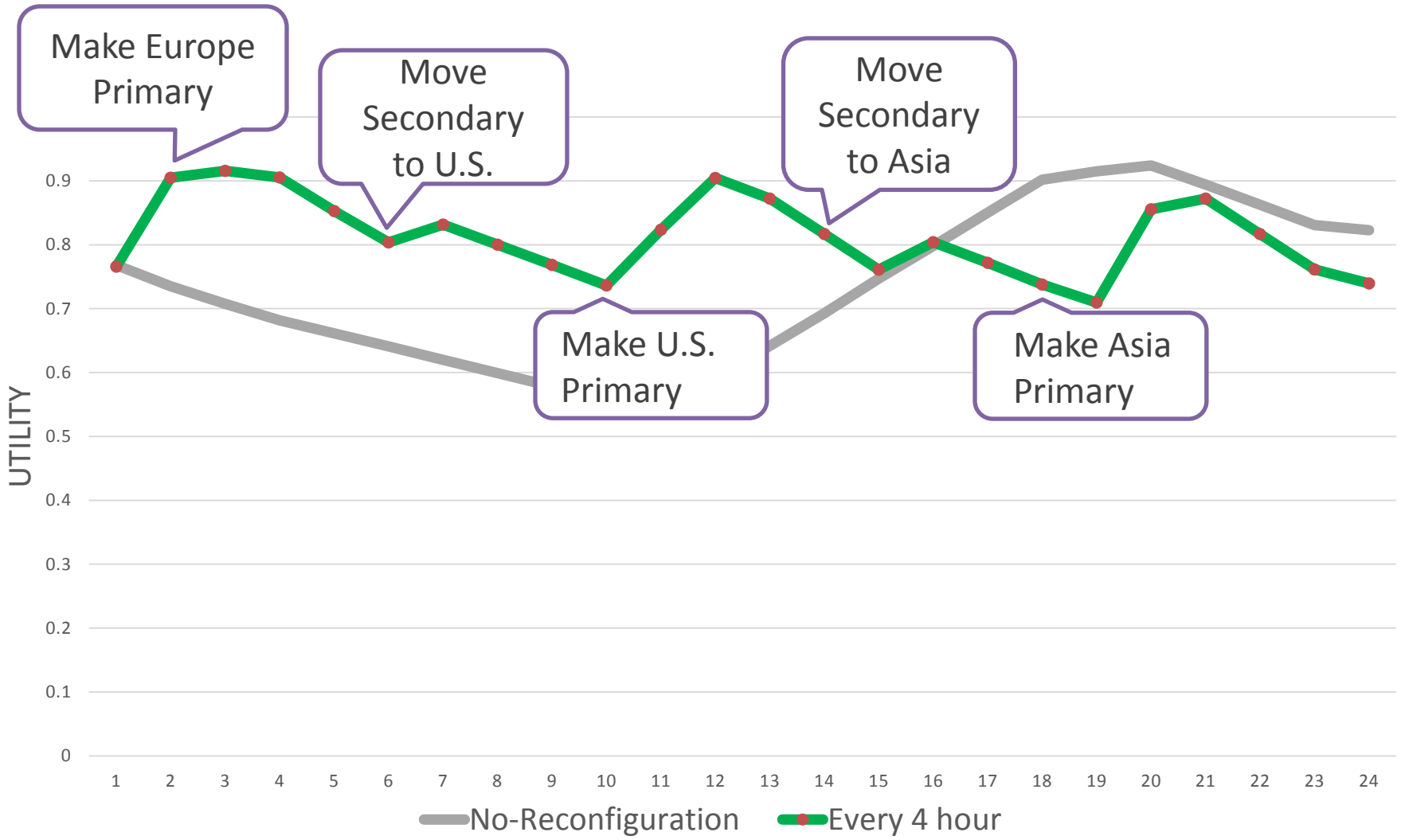
Consistency	Latency	Utility
Strong	100 ms	1.0
RMW	100 ms	0.7
Eventual	250 ms	0.5

Constraints
Replication factor ≤ 2
Single primary

Evaluation

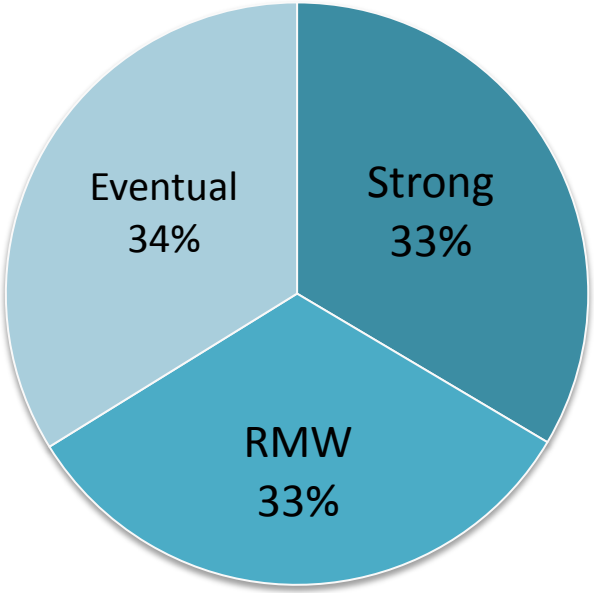

UTILITY
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
No-Reconfiguration

Evaluation

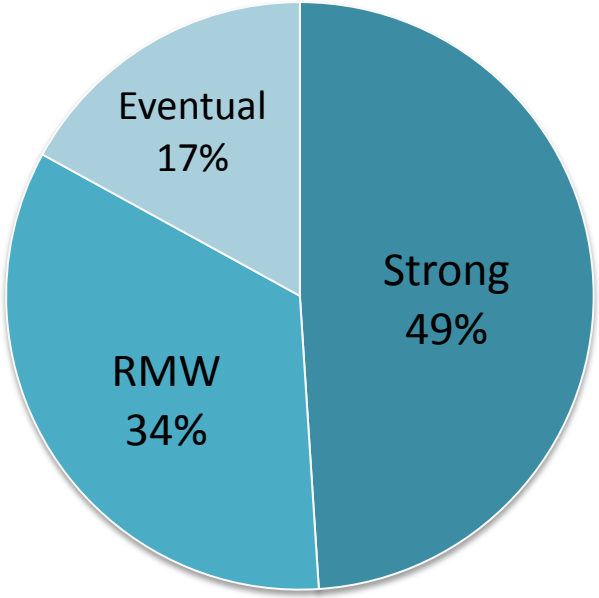

Make Europe Primary
Move Secondary to U.S.
Move Secondary to Asia
Make U.S. Primary
Make Asia Primary
UTILITY
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
No-Reconfiguration
Every 4 hour

Improved Consistency

No-Reonfiguration



Every 4 hour



Conclusions

- Storage systems should adapt to changing client demands
- Utility/cost is a useful metric for selecting improved configurations
- Automatic reconfiguration can occur in parallel with running applications
- Substantial consistency gains are possible