Geographically distributed Swift clusters

Alistair Coles
Swift core developer

alistairncoles@gmail.com
irc: acoles

SwiftStack

Overview

- What is Swift?
- Geographically distributed clusters
  - What?
  - Why?
  - How?
- Erasure coded geographically distributed cluster
  - Swift now supports these !
  - …enabled by fragment duplication and composite rings
- Summary

SwiftStack

What is Swift?

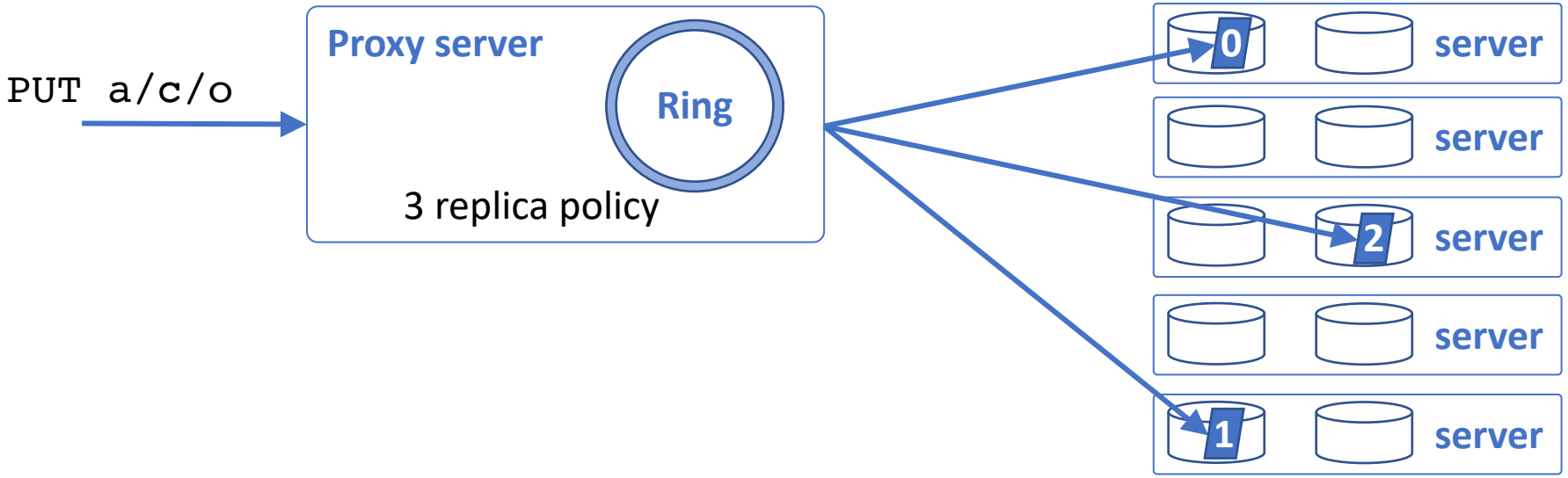
- openstack. object storage service
- REST API – Create, Read, Update, Delete
- Simple naming hierarchy
  - objects belong to containers
  - containers belong to accounts

SWIFT

an OpenStack Community Project

SwiftStack

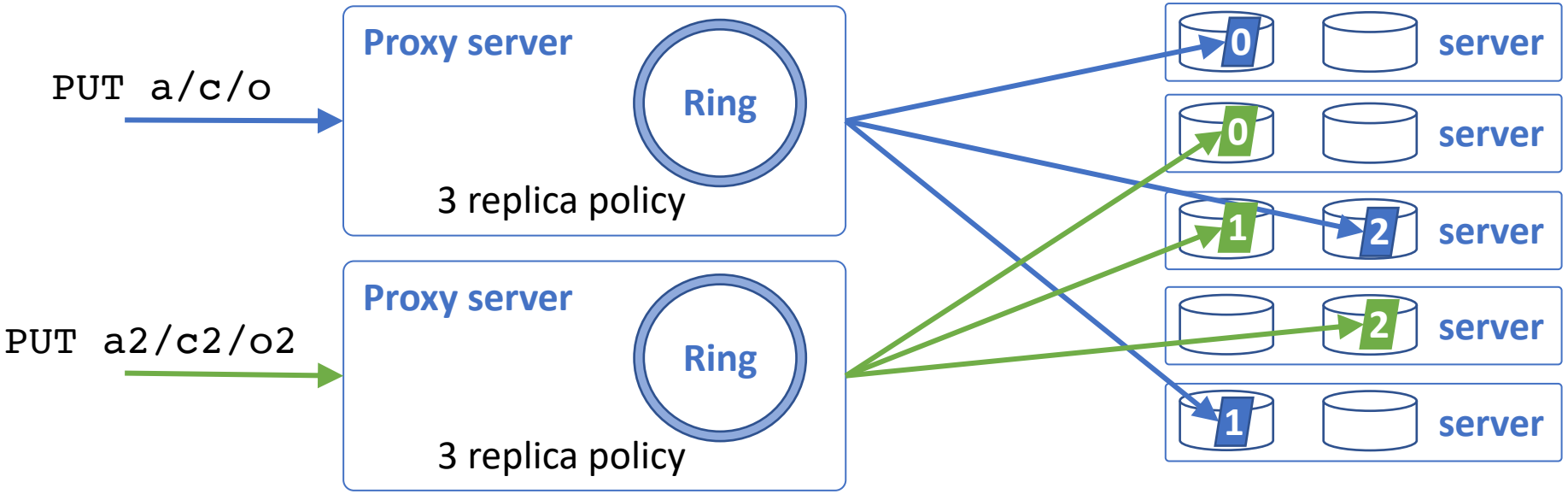
Swift is durable



- Multiple replicas of every object (or erasure coding)
- The Ring always tries to disperse replicas across different devices and servers

SwiftStack

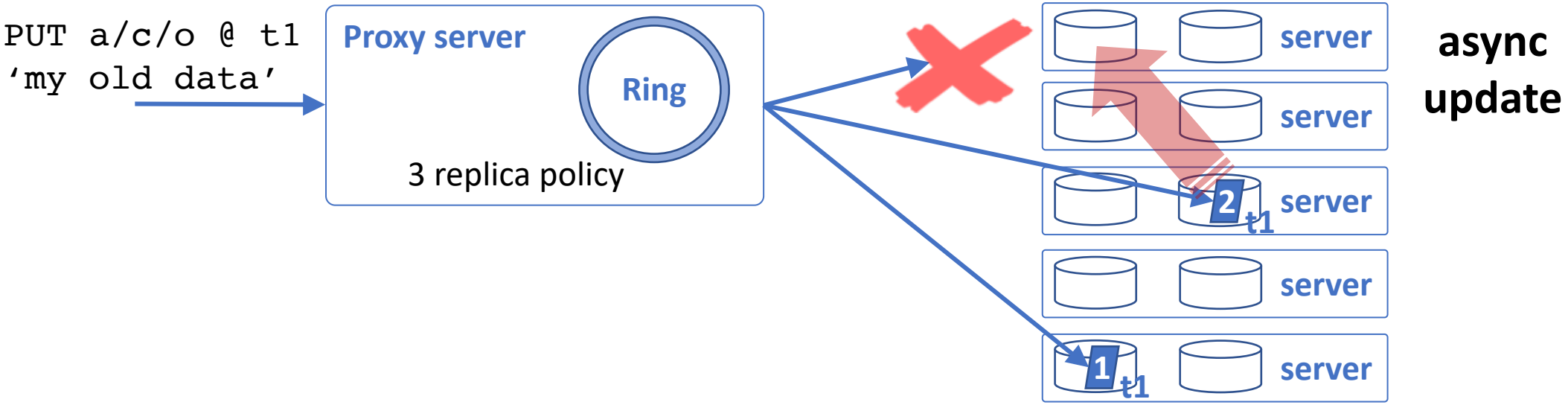
Swift is scalable



- The Ring always tries to balance load across all devices
- No centralized services

SwiftStack

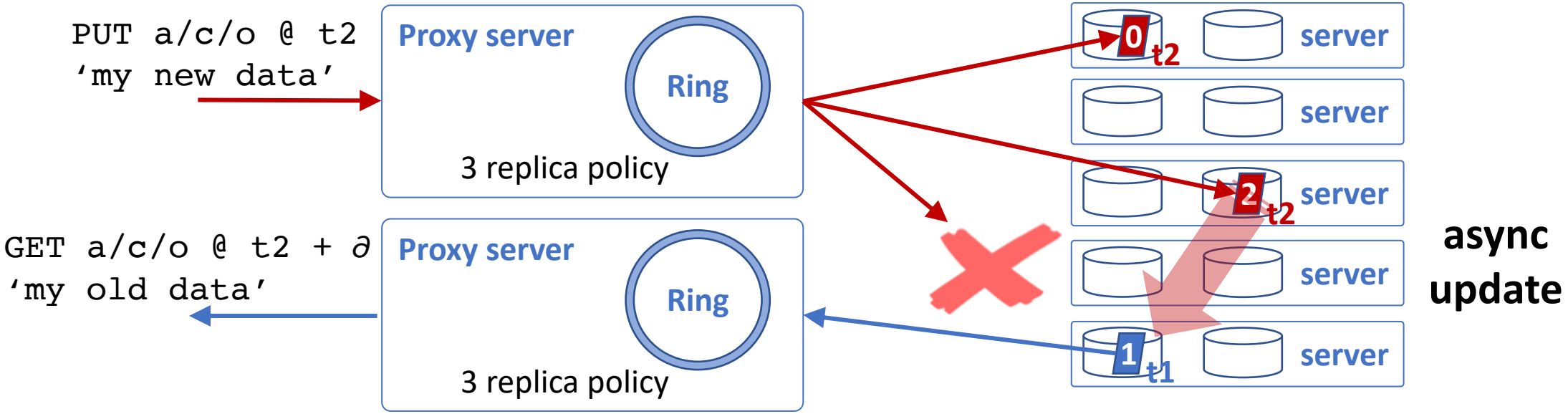
Swift is highly available



- Write succeeds on quorum of replicas
- Missing replicas are updated asynchronously

SwiftStack

Swift is eventually consistent



- Possibility to read stale data
- Async process makes data eventually consistent

SwiftStack

Overview

- ~~What is Swift?~~
- Geographically distributed clusters
  - What?
  - Why?
  - How?
- Erasure coded geographically distributed cluster
  - Swift now supports these!
  - … but there's some new stuff to know about
- Deep dive: erasure coding, fragment duplication, composite rings
- Summary

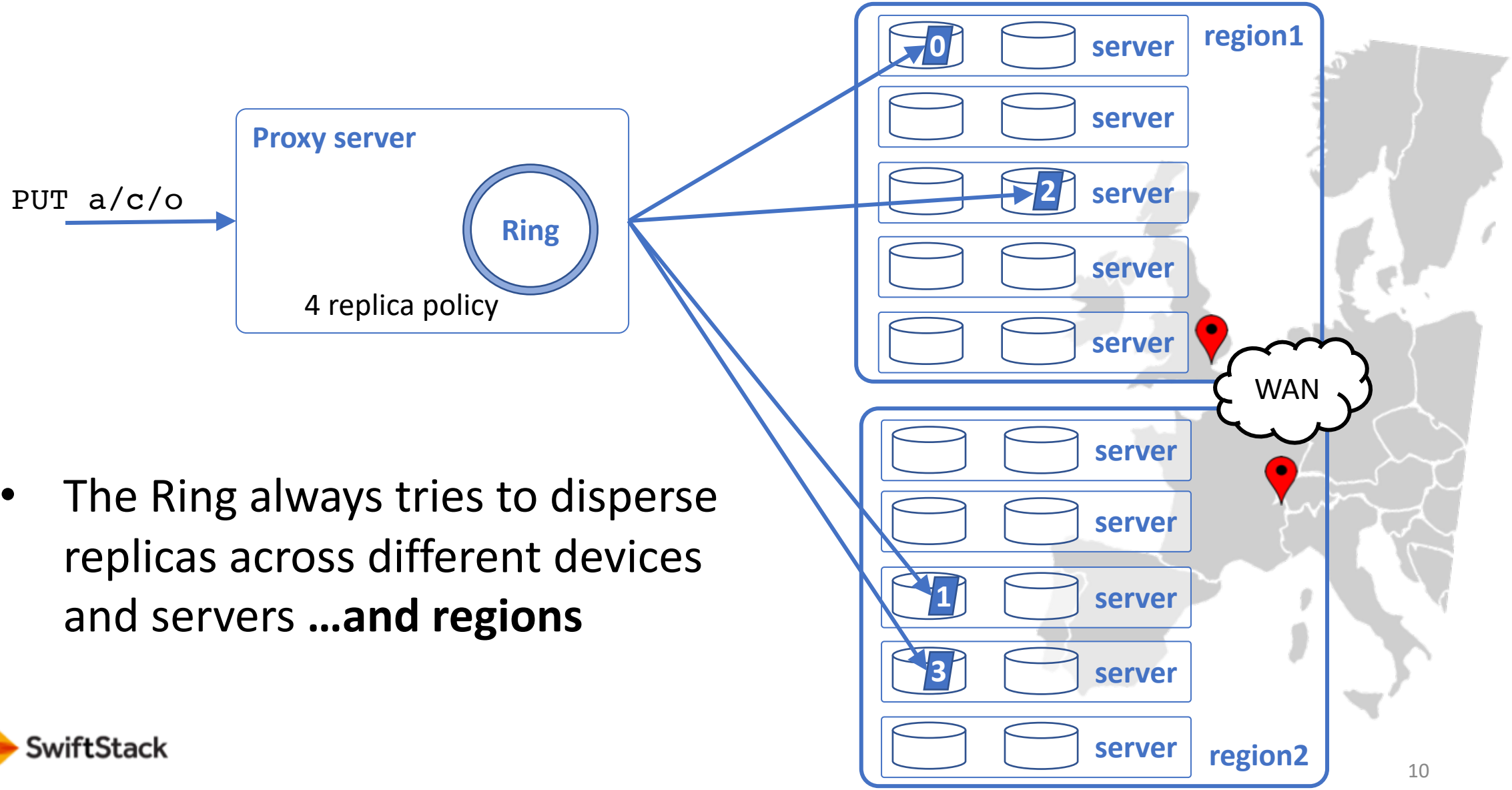
SwiftStack

Geographically distributed clusters

- What?
  - Multiple physical locations
    - Typically connected by high latency/low bandwidth WAN
  - Copies of data in each location
  - Single namespace
- Why?
  - Disaster recovery
  - Data locality
- Also known as “Global clusters”, “Multi-region Swift”

SwiftStack

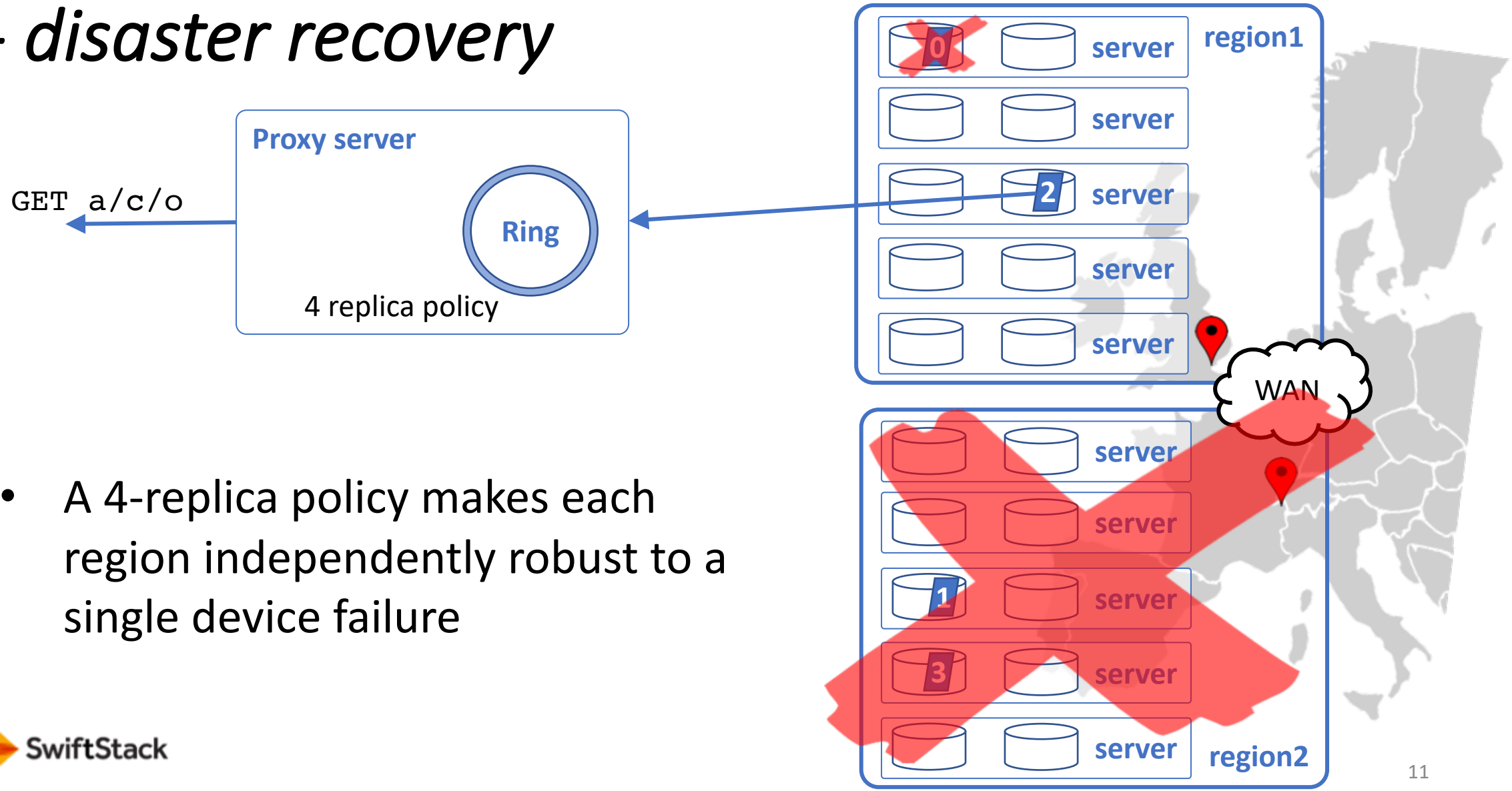
Geographically distributed Swift clusters



- The Ring always tries to disperse replicas across different devices and servers **…and regions**

SwiftStack

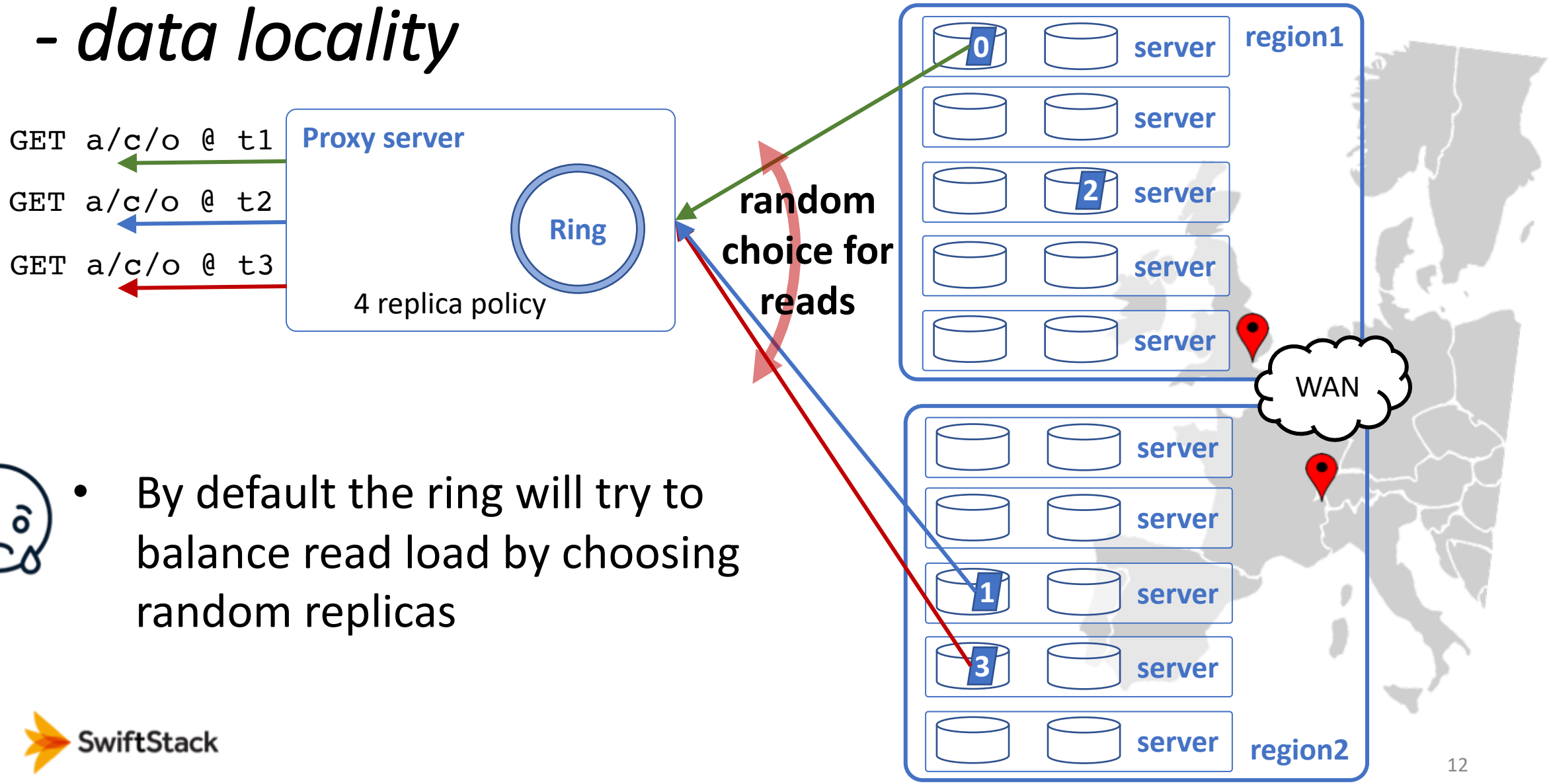
Geographically distributed clusters
- *disaster recovery*



- A 4-replica policy makes each region independently robust to a single device failure

SwiftStack

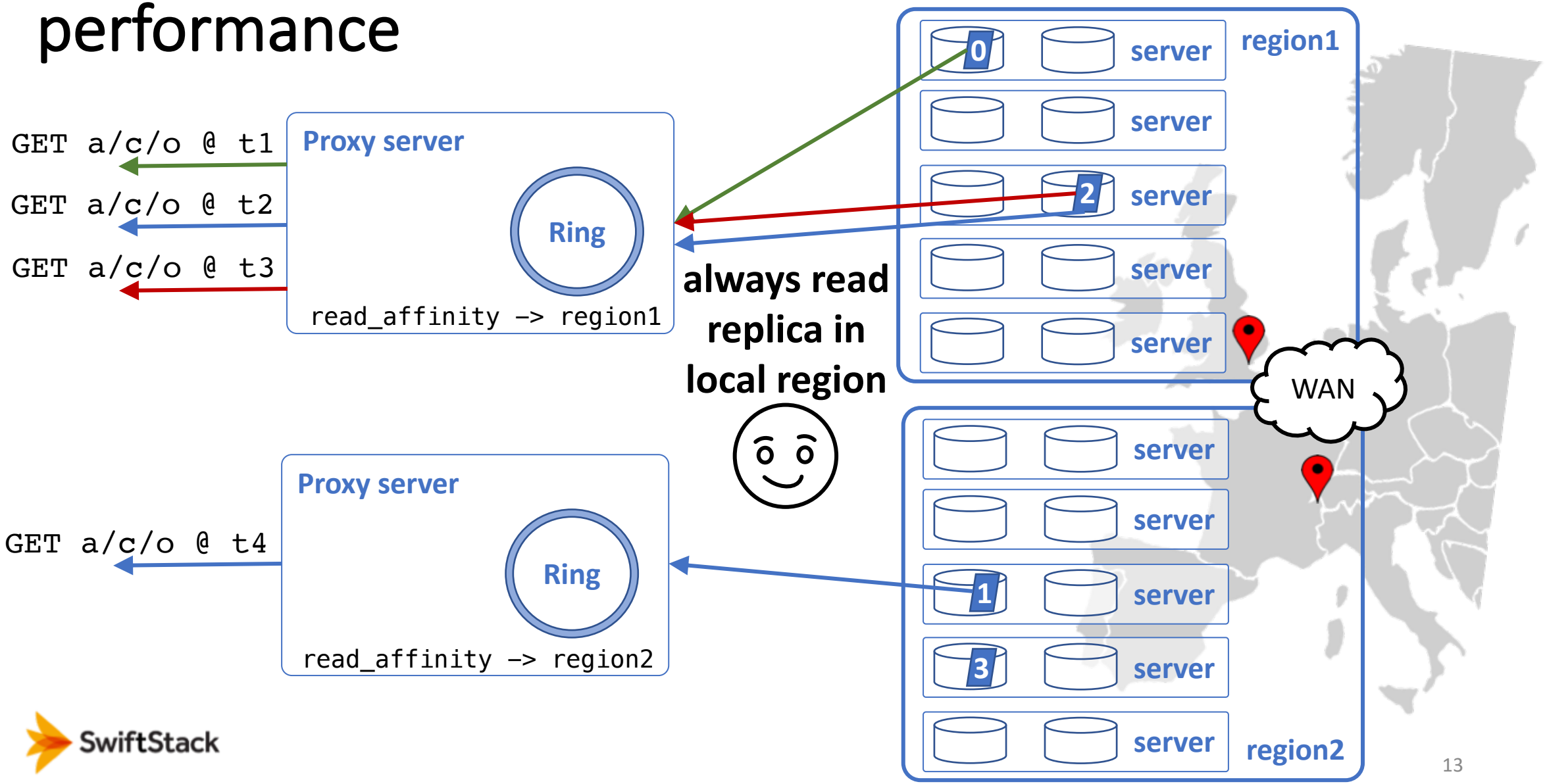
Geographically distributed Swift clusters - *data locality*



- By default the ring will try to balance read load by choosing random replicas

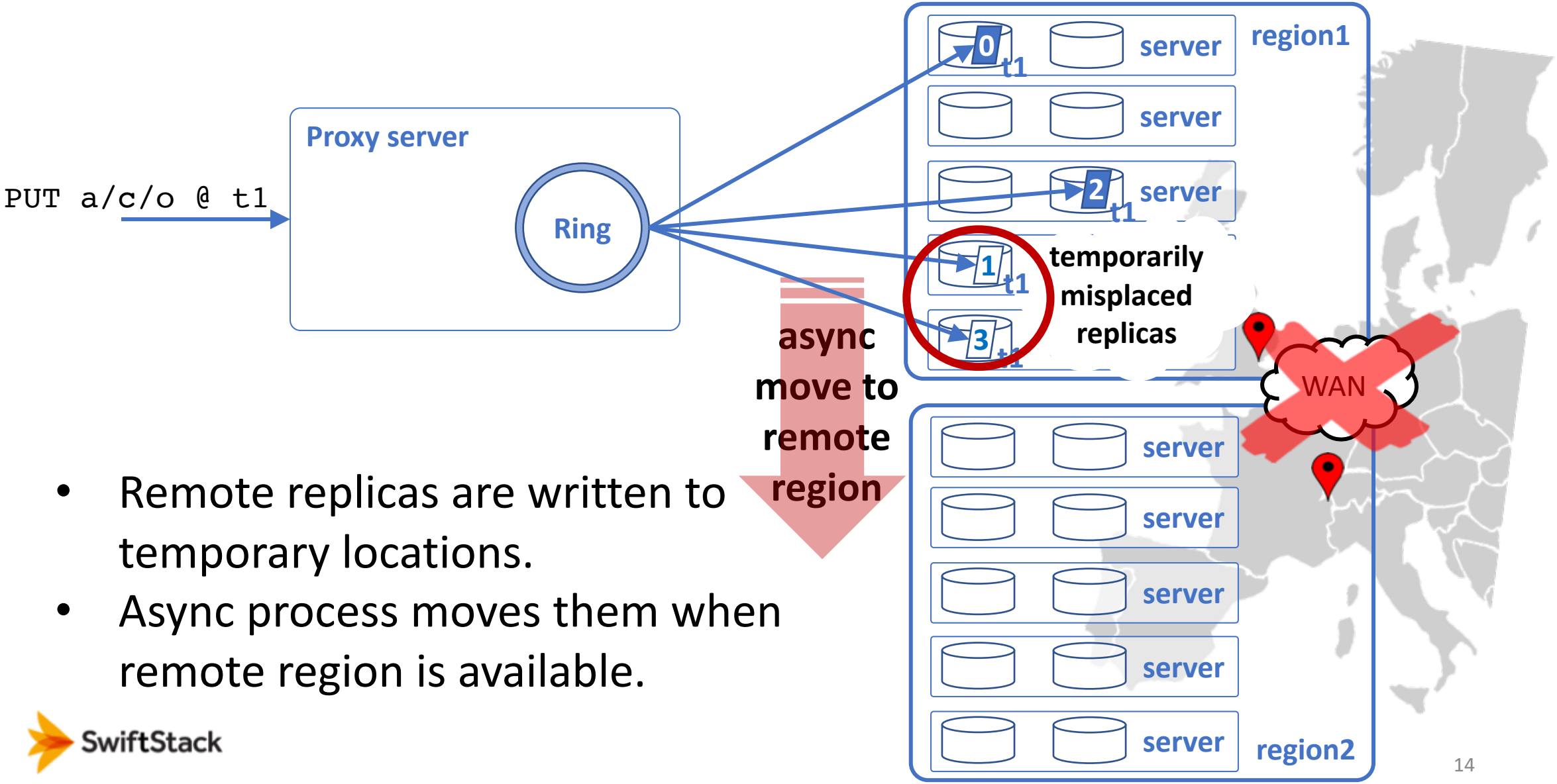
SwiftStack

Read affinity – trade off load balancing for read performance



SwiftStack

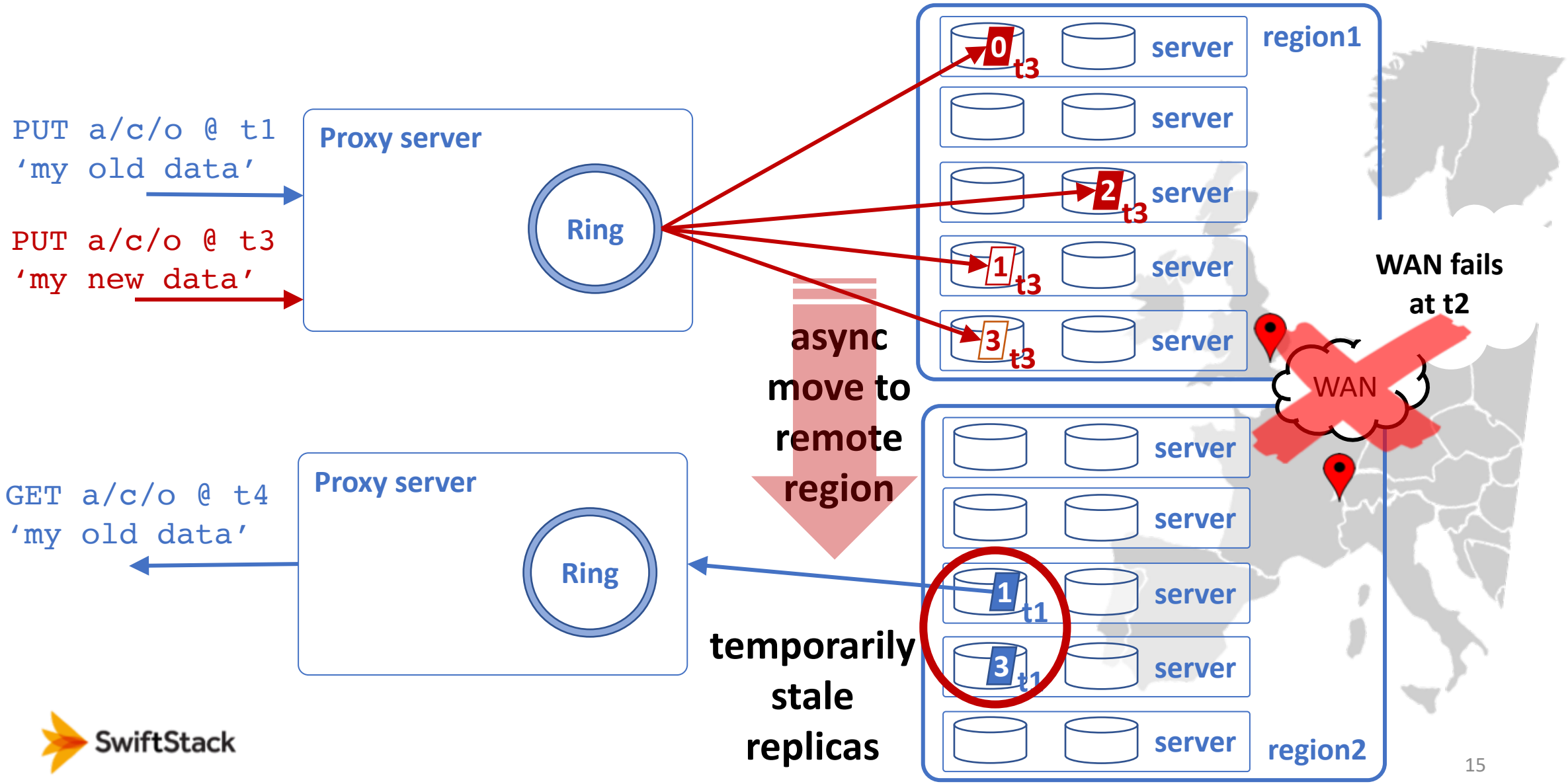
What if remote writes fail?



- Remote replicas are written to temporary locations.
- Async process moves them when remote region is available.

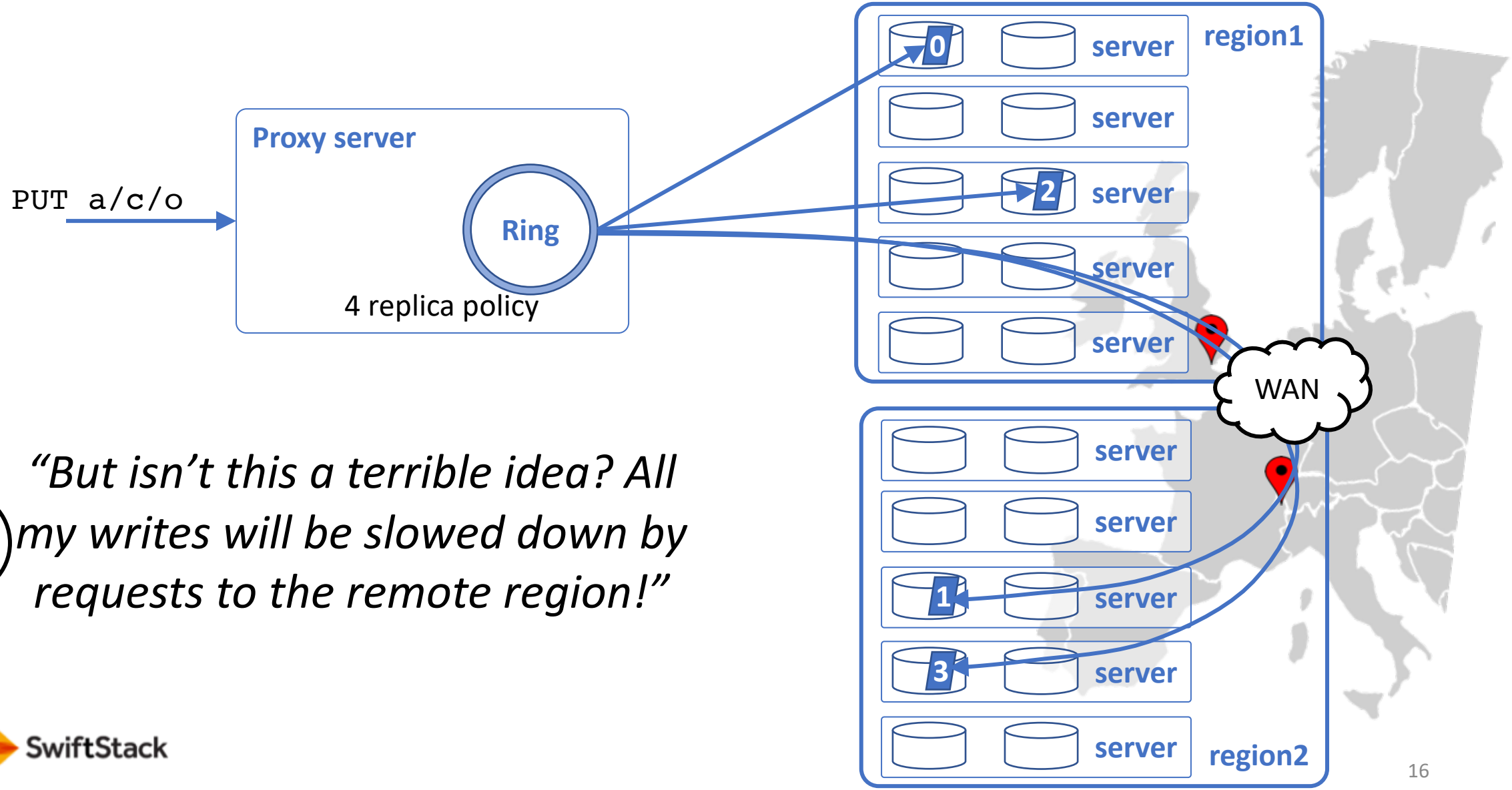
SwiftStack

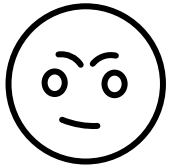
Overwrite - replicas are *eventually* consistent



SwiftStack

What if remote writes are slow?



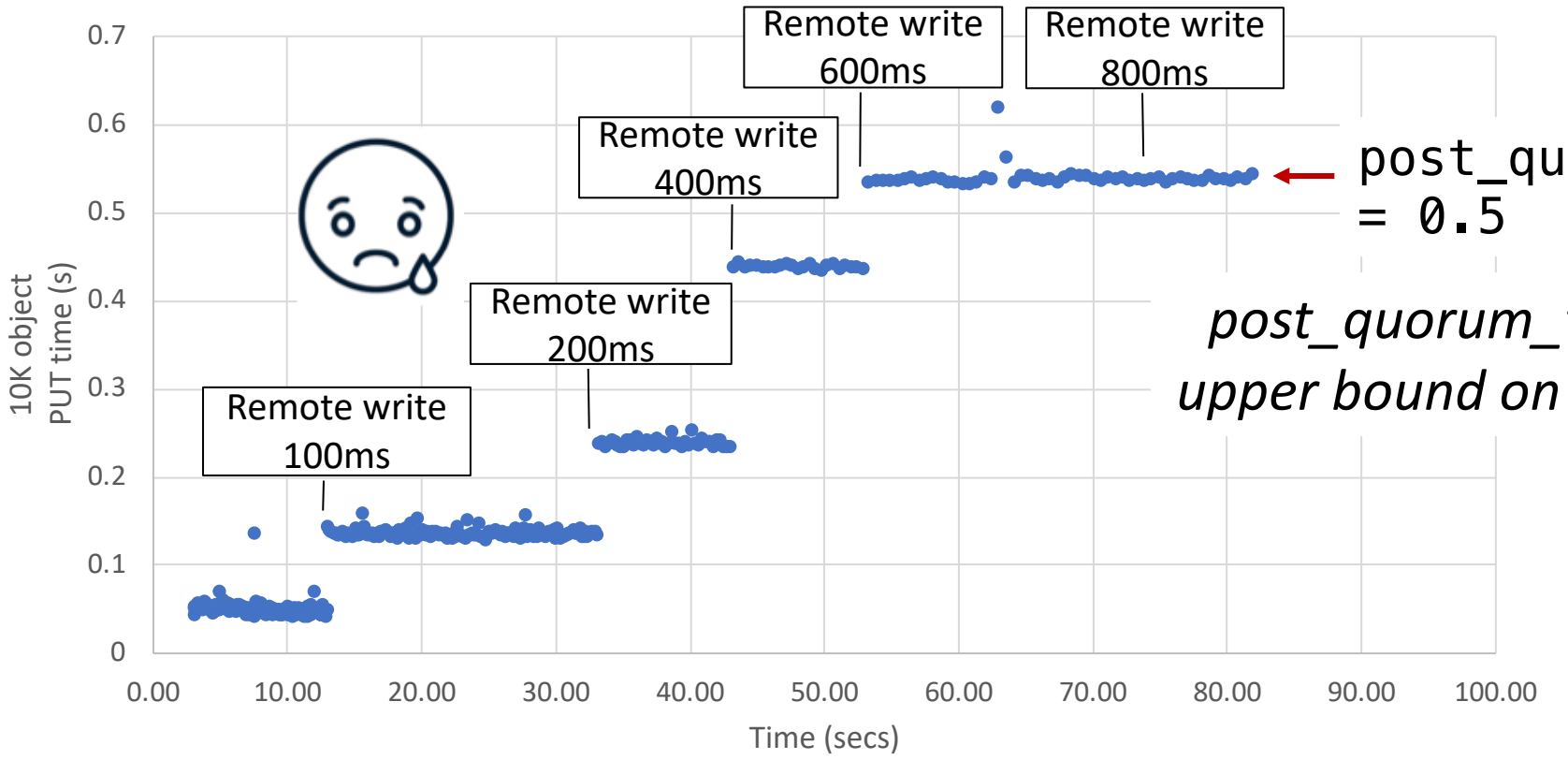


"But isn't this a terrible idea? All my writes will be slowed down by requests to the remote region!"

SwiftStack

Effect of remote region write time on PUTs

(remote region servers artificially slowed)

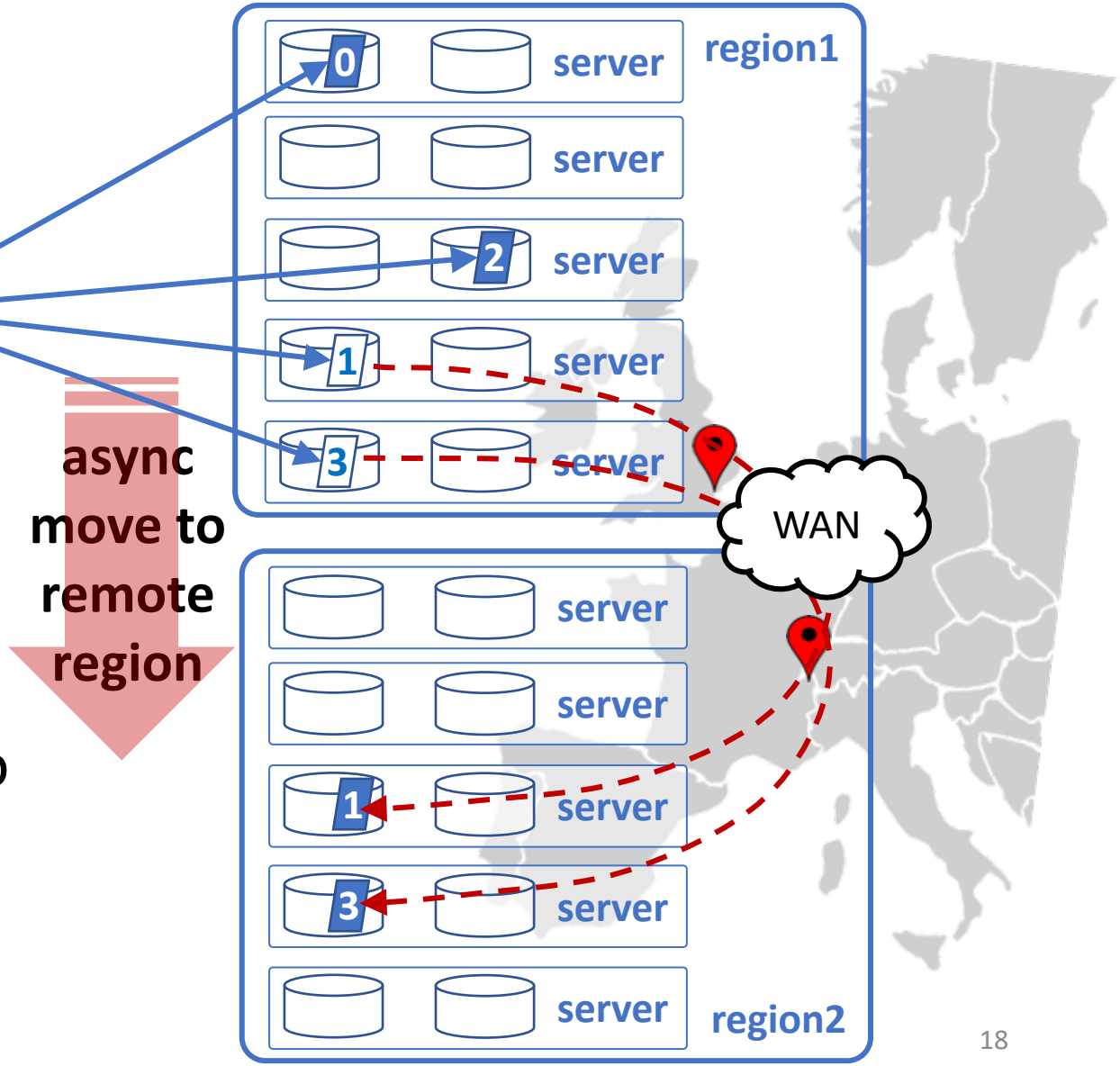


`post_quorum_timeout = 0.5`

post_quorum_timeout puts an upper bound on the extra latency

SwiftStack

Write affinity – temporarily trade off *dispersion* for write performance

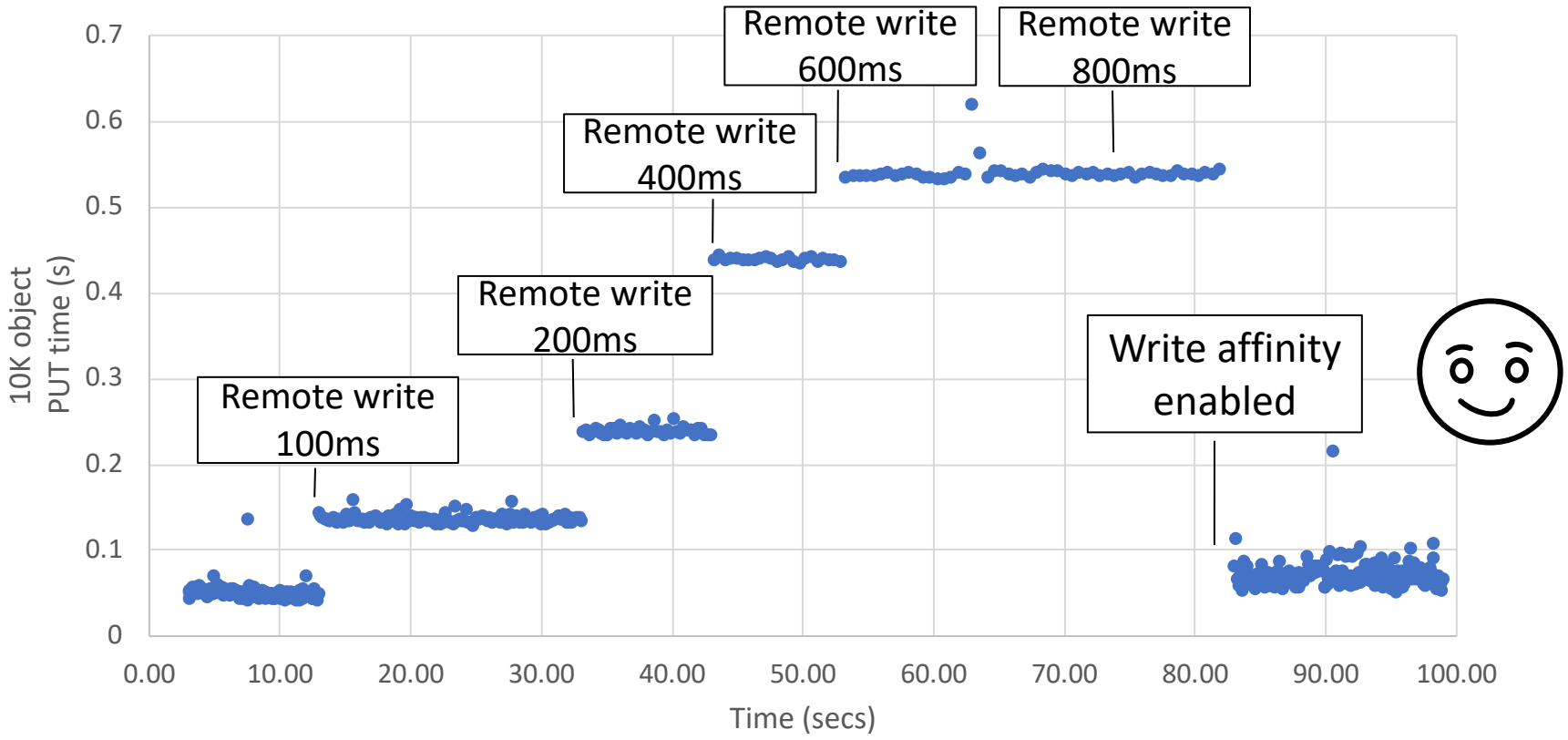


- Remote replicas are initially written to the local region.
- Async process moves them to remote region.

SwiftStack

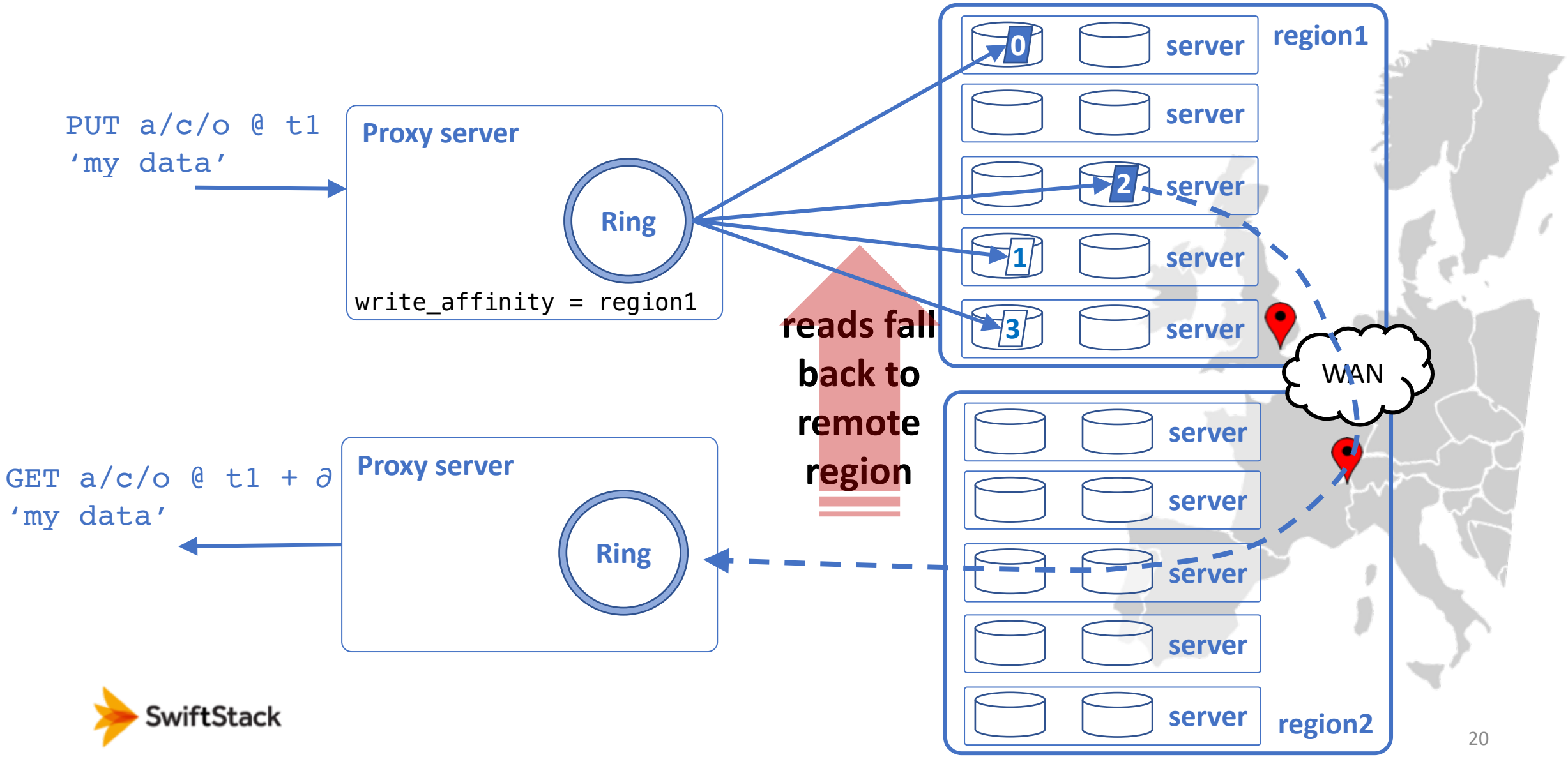
Write affinity performance improvement

(remote region servers artificially slowed)



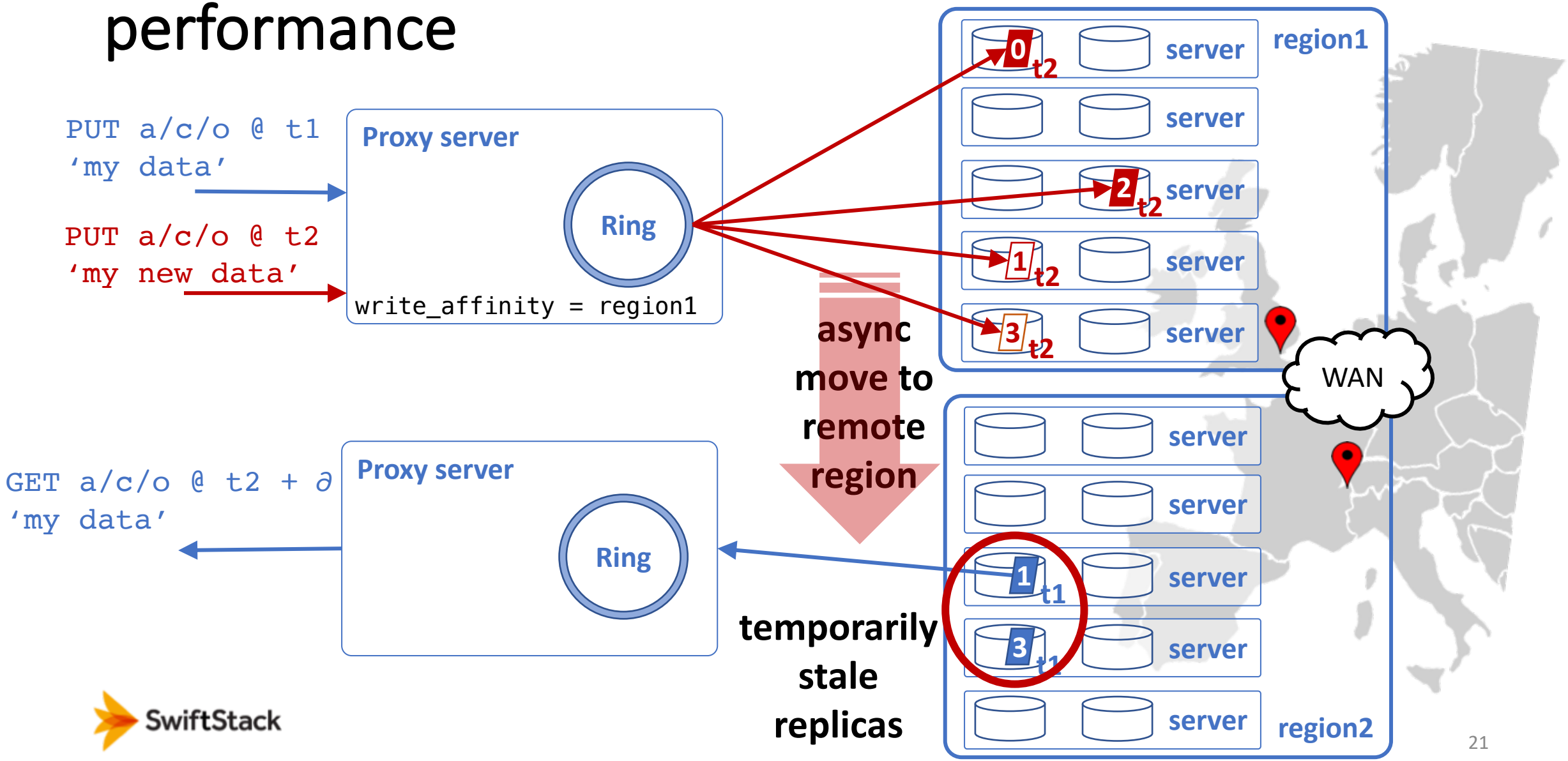
SwiftStack

Write affinity – data is always available



SwiftStack

Write affinity – trade off *consistency* for write performance



SwiftStack

Workload considerations

- Use write affinity with care!
  - “No free lunch” - replicas have to be copied across WAN eventually
- Suitable workloads:
  - Moderate or bursty write rates
  - Non-immediate remote reads
  - E.g. replicating archive data across sites
- Unsuitable workloads:
  - Continuous high write rate
    - misplaced replicas will back up in local region
  - Immediate remote reads
    - reads will fetch data over WAN before async move has happened

SwiftStack

Overview

- ~~What is Swift?~~
- ~~Geographically distributed clusters~~
  - ~~What?~~
  - ~~Why?~~
  - ~~How?~~
- Erasure coded geographically distributed cluster
  - Swift now supports these!
  - … but there's some new stuff to know about
- Deep dive: erasure coding, fragment duplication, composite rings
- Summary

SwiftStack

Erasure coding – same durability, less storage

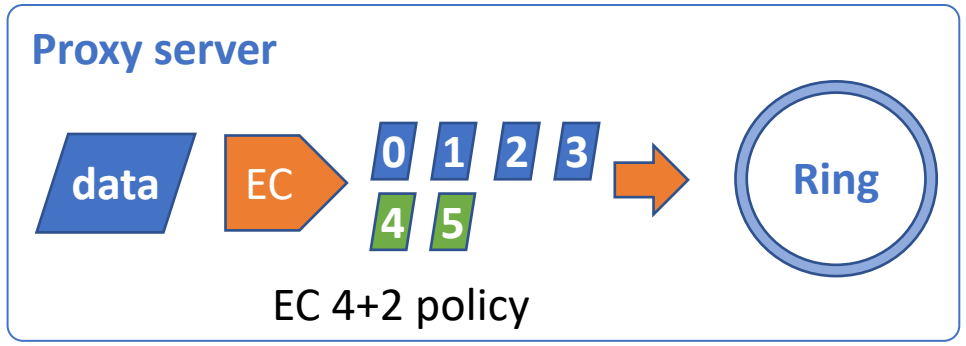
Example: 4 data fragments + 2 parity fragments

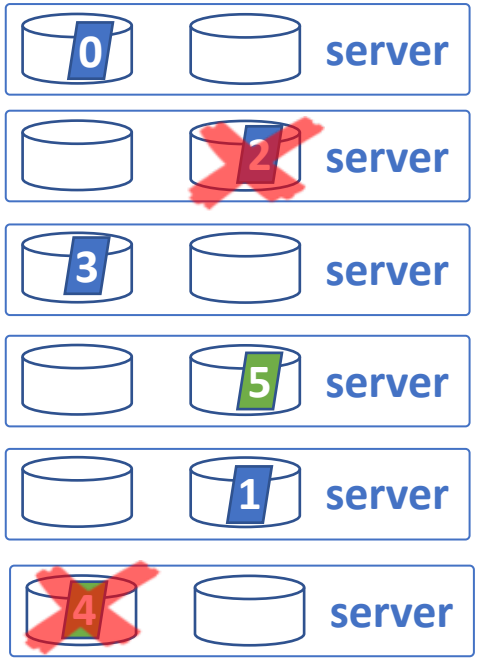
Any subset of 4 unique fragments is sufficient to reconstruct data:

*Requires only **1.5 x size of data** to store all fragments*

SwiftStack

Erasure coding in Swift





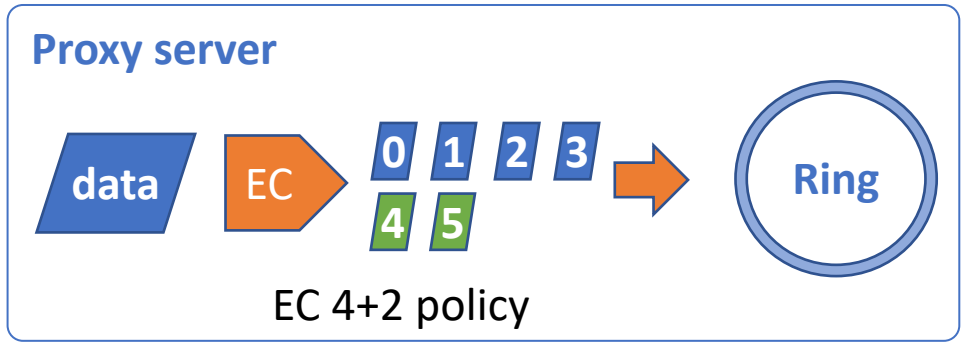
https://github.com/openstack/pyeclib
Python interface to liberasurecode

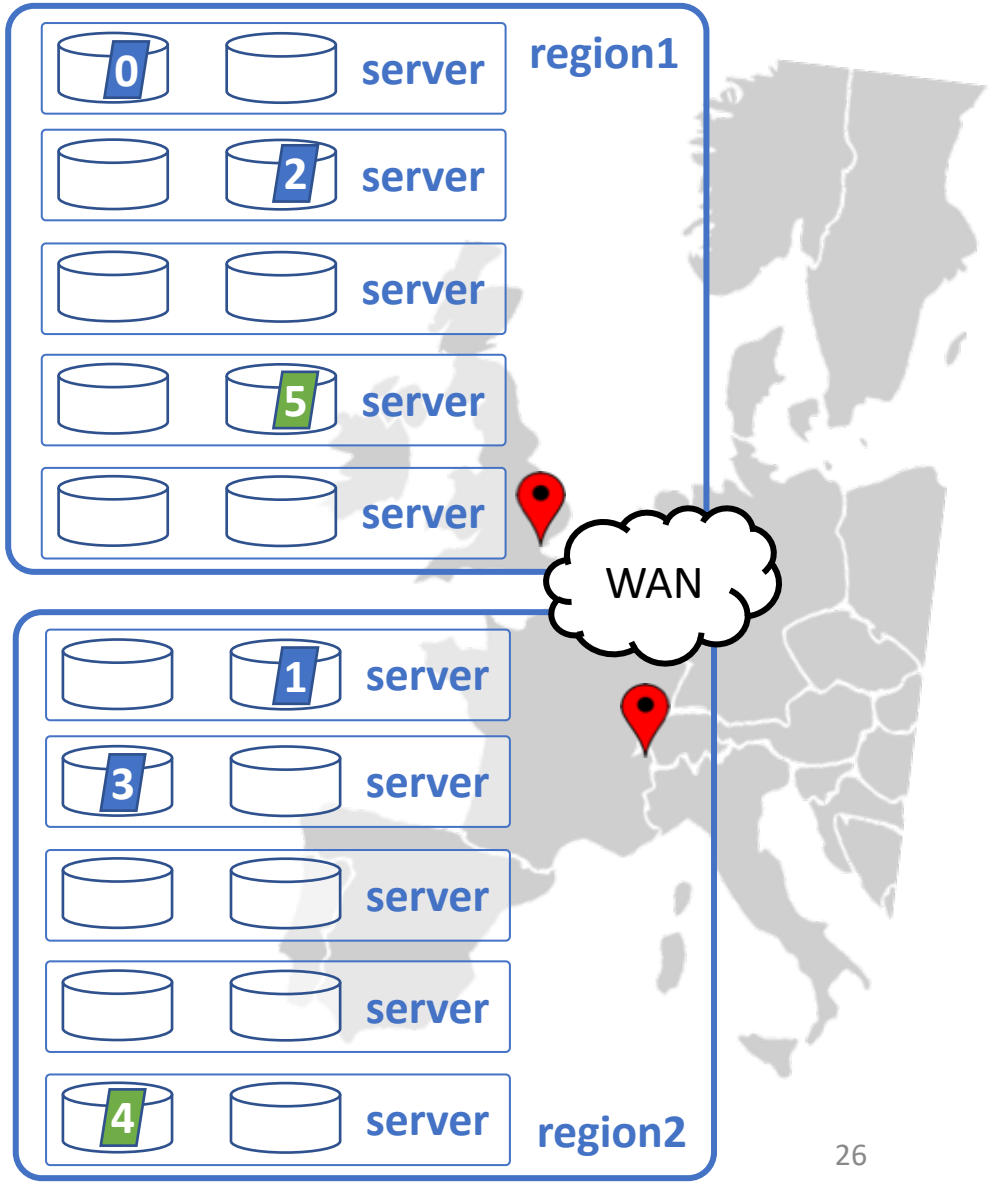
https://github.com/openstack/liberasurecode
C library with pluggable Erasure Code backends

4 + 2 erasure coding requires approx. ***50% storage vs*** *3 replicas with* ***similar durability***

SwiftStack

Erasure coding across regions

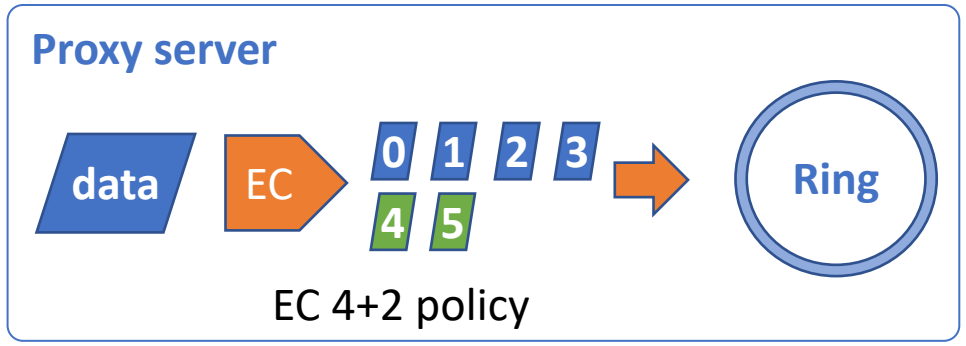




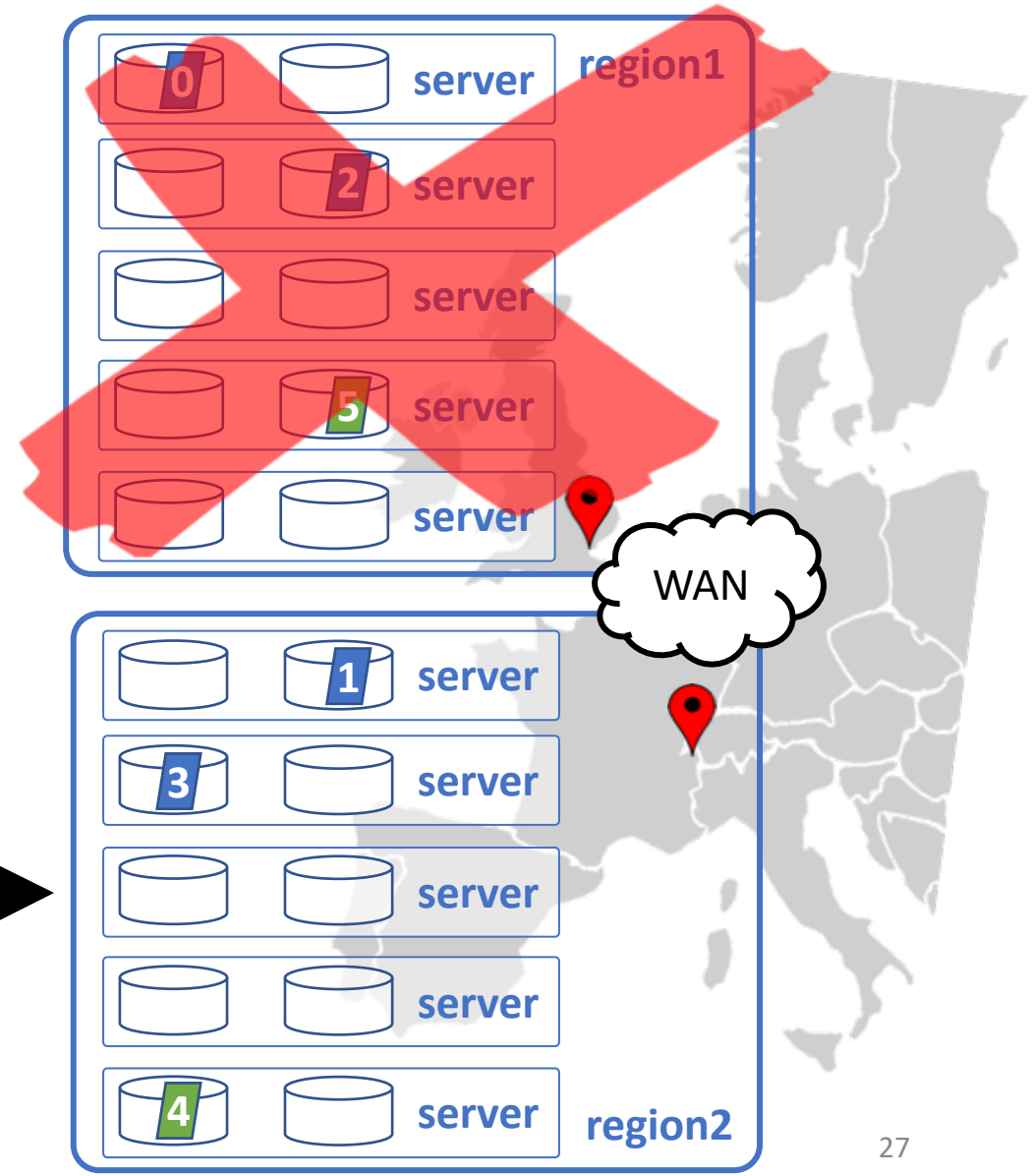
- The Ring always tries to disperse **fragments** across different devices and servers **…and regions**

SwiftStack

Erasure coding across regions

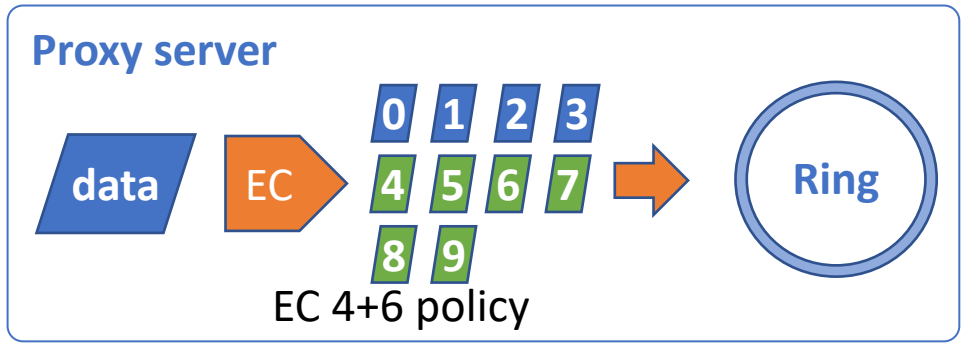


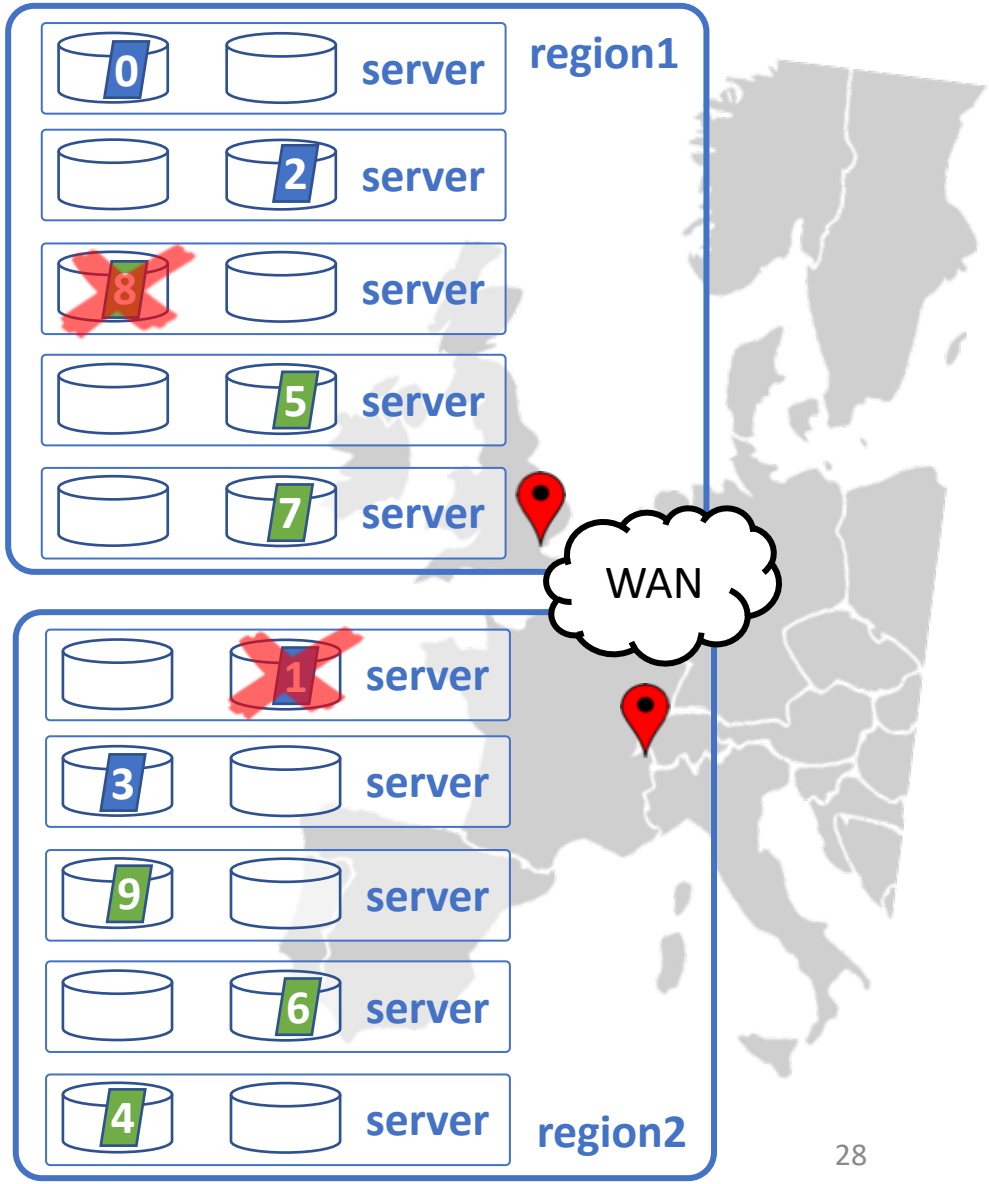
With EC 4+2 we don't have enough fragments in one region to reconstruct the data!



SwiftStack

Erasure coding across regions requires more fragments

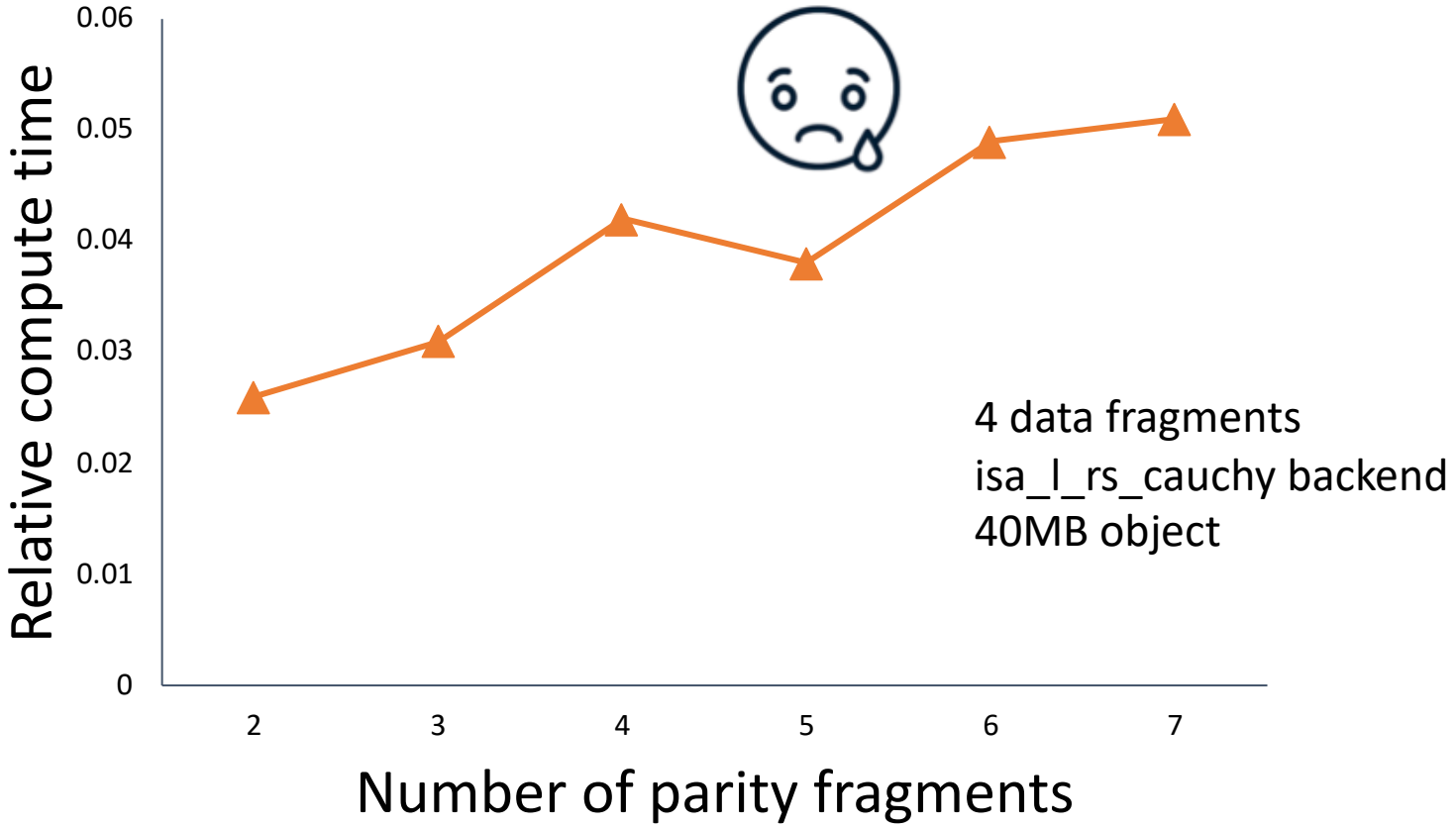




How about EC 4+6?
*requires **2.5 x size of data***
vs
*Replication requires **4 x size of data** for similar durability*

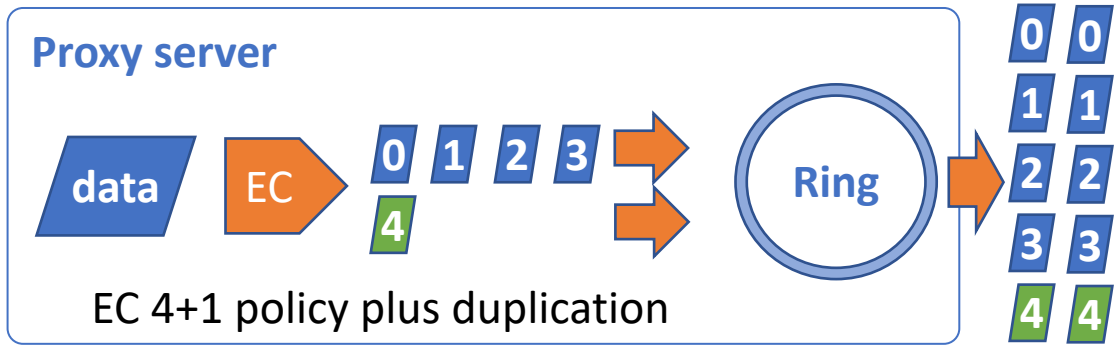
SwiftStack

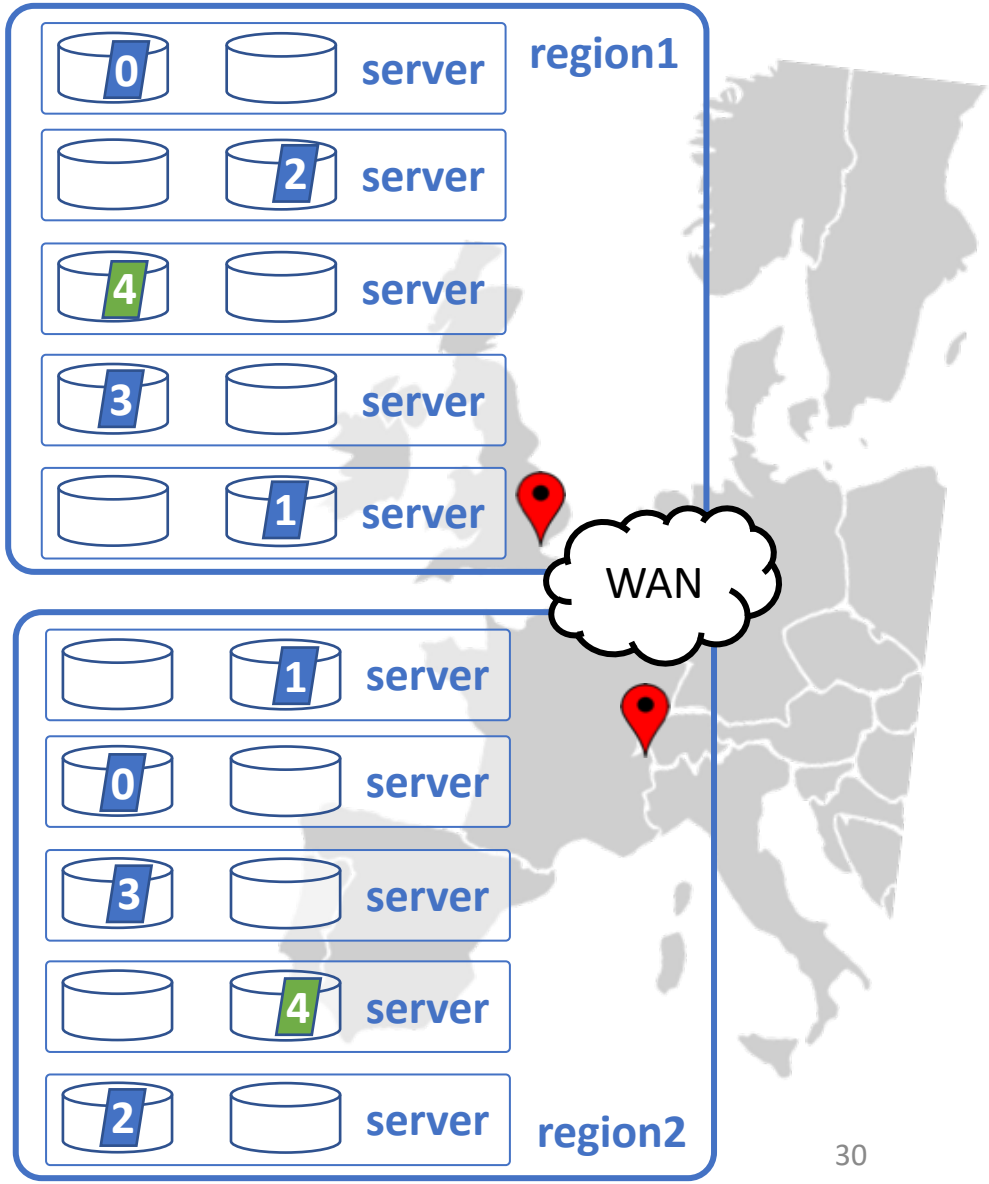
Erasure coding time increases with number of parity fragments



SwiftStack

EC duplication: more fragments, less compute





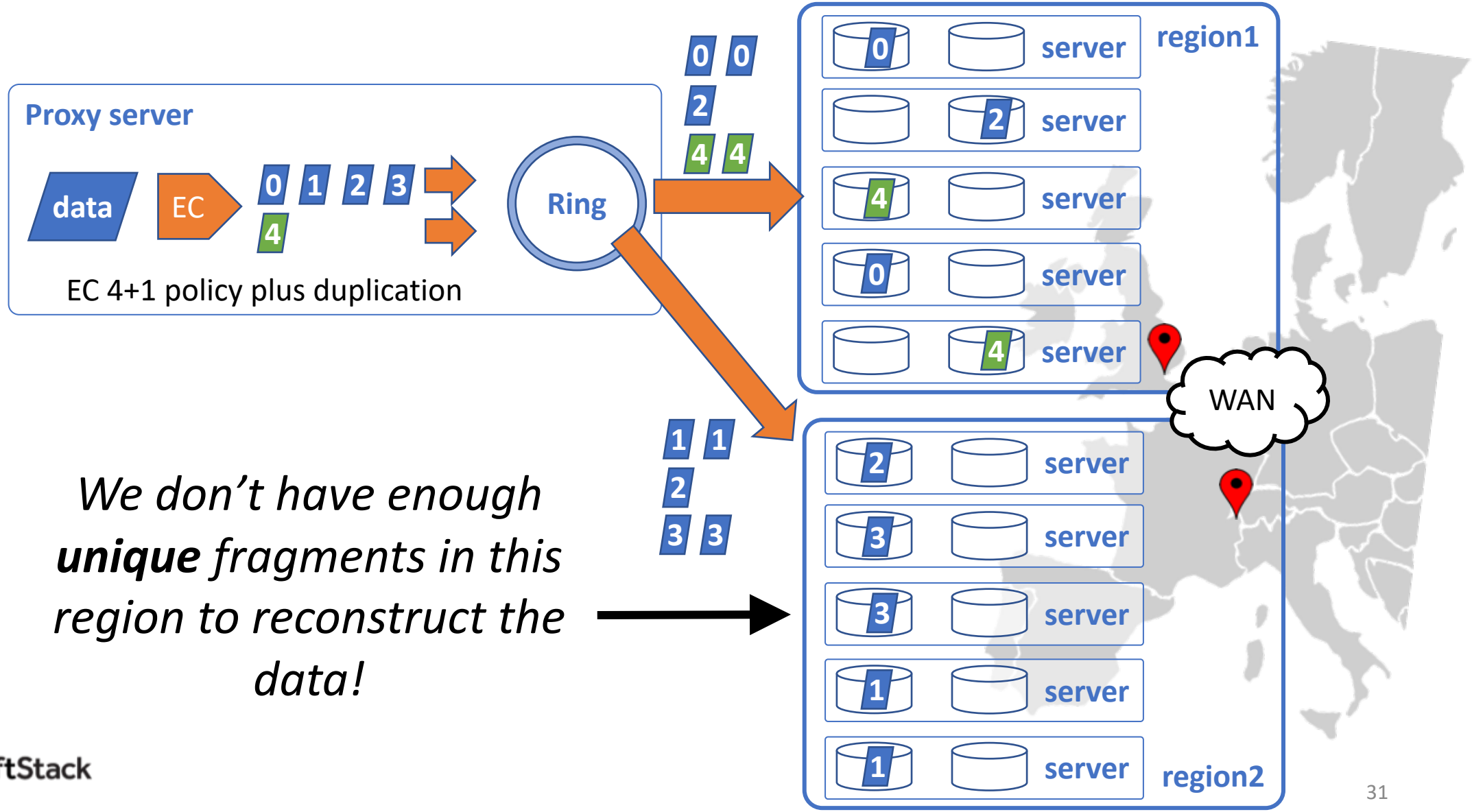
*Each region has EC 4+1 fragments requires **2.5 x size of data***

vs

*Replication requires **4 x size of data** for similar durability*

SwiftStack

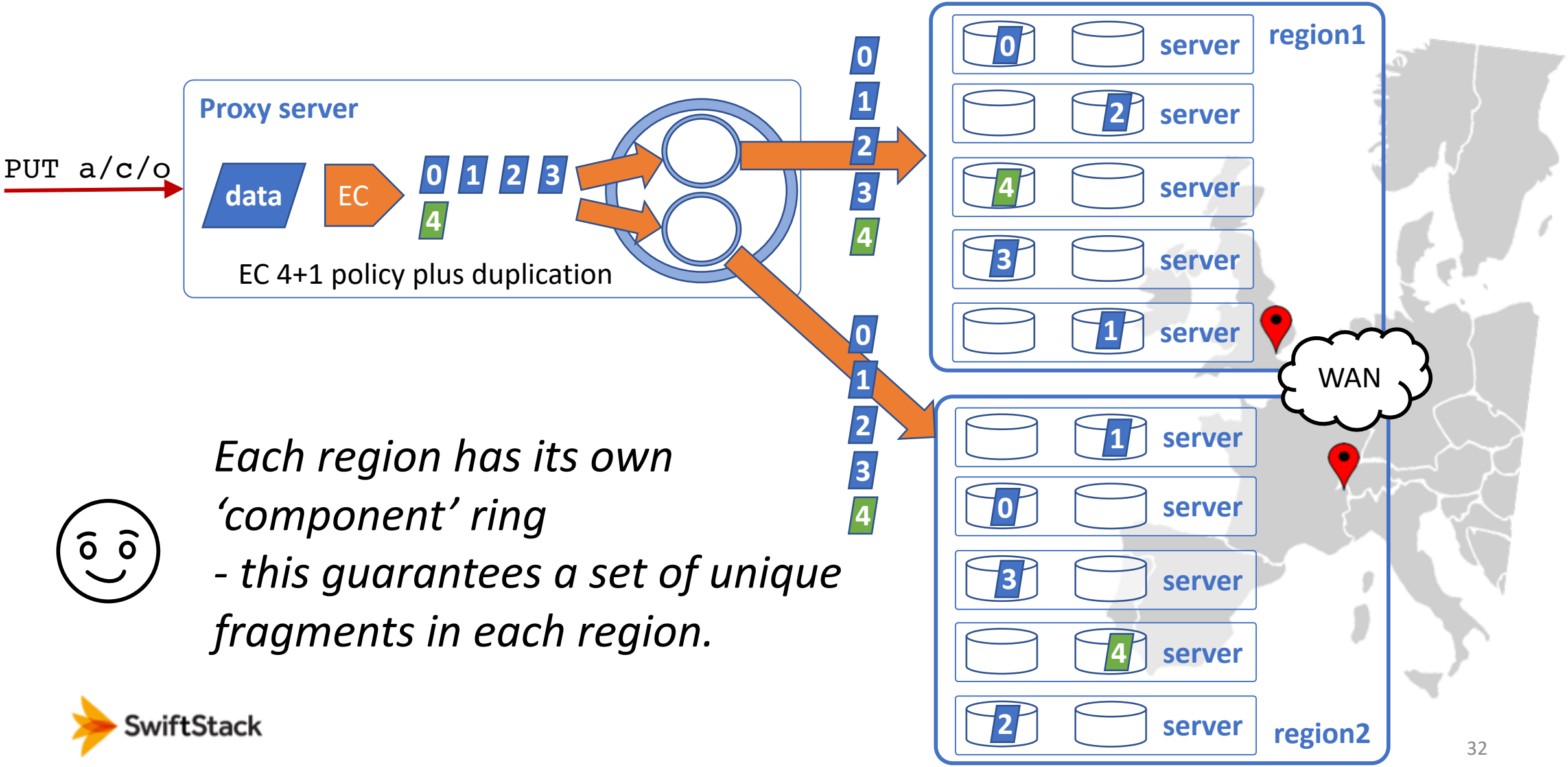
But duplicates must be correctly dispersed…



*We don't have enough **unique** fragments in this region to reconstruct the data!*

SwiftStack

Solution: EC duplication + composite rings



Each region has its own 'component' ring
- this guarantees a set of unique fragments in each region.

SwiftStack

Summary

- Swift enables you to build geographically distributed clusters
- Good for disaster recovery and data locality
- Tuning via options in the Swift proxy server
  - `write_affinity, read_affinity`
  - `Understand your workloads!`
- Erasure coded storage policies can also be geographically distributed[1]
  - Uses new features: EC fragment duplication and composite rings

[1] new in Swift 2.15.0

SwiftStack

Swift welcomes new users and contributors

- You can find us in freenode #openstack-swift
- Project: https://launchpad.net/swift
- Docs: https://docs.openstack.org/swift
- Code: https://github.com/openstack/swift

SWIFT

an OpenStack Community Project

SwiftStack