GlobalFS: A Strongly Consistent Multi-Site Filesystem

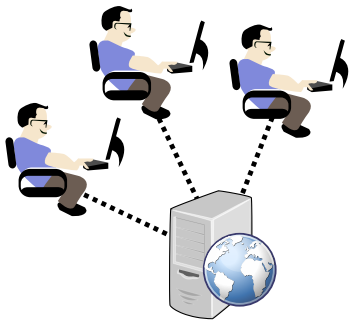
Leandro Pacheco Raluca Halalai Valerio Schiavoni
Fernando Pedone Etienne Rivière Pascal Felber

Università della Svizzera italiana

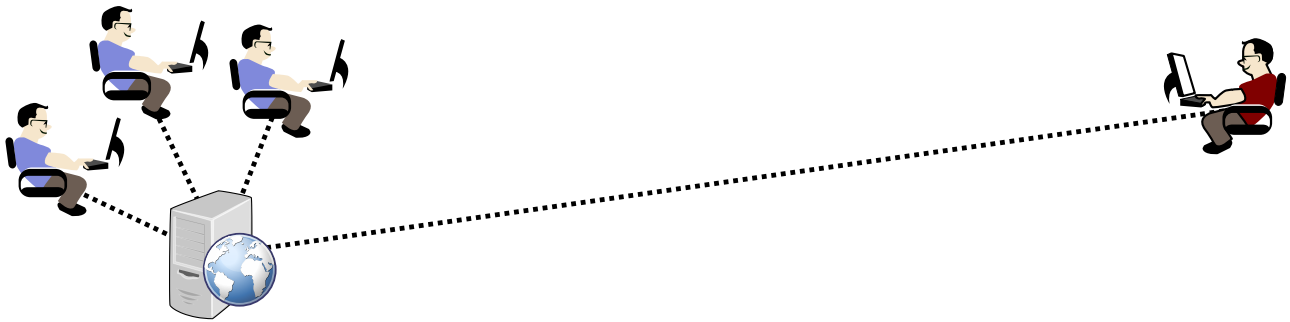
unine
UNIVERSITÉ DE NEUCHÂTEL

RainbowFS Workshop
May 3rd, 2017

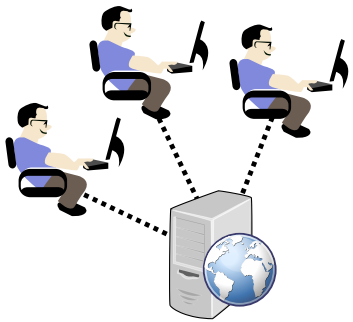
Distributed applications

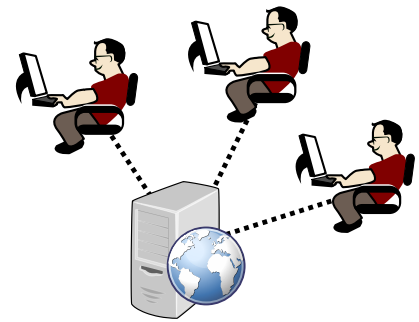


Distributed applications



Distributed applications





Distributed applications

Distributed applications

Distributed applications

Distributed Storage

SQL Databases

NoSQL Databases

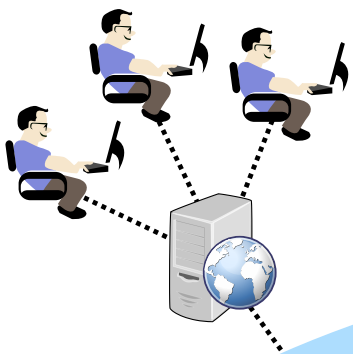
Key-value storage

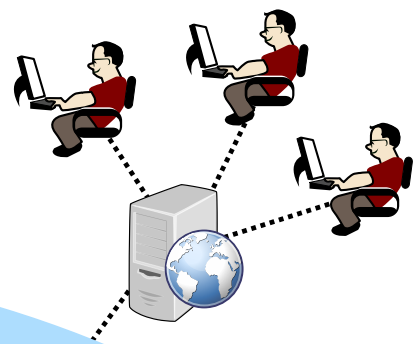
Coordination Systems

Caches

File Systems

Distributed applications





Distributed Storage

SQL Databases

NoSQL Databases

Key-value storage

Coordination Systems

Caches

File Systems → **Easy interoperability for existing aplications**

Global infrastructure

Amazon's AWS global infrastructure

CAP theorem

Weak Consistency	Strong Consistency
Lower latency	Clear semantics and guarantees
Higher availability	Easier to reason about
Possibly incorrect/unexpected results	Block instead of providing incorrect results

What is GlobalFS?

Geographically distributed filesystem

Familiar interface (POSIX)

Strong consistency

Fault-tolerance through replication

Flexible performance through locality

Overall design

Separate data and metadata

Partial replication

Metadata protocol exploiting *atomic multicast*

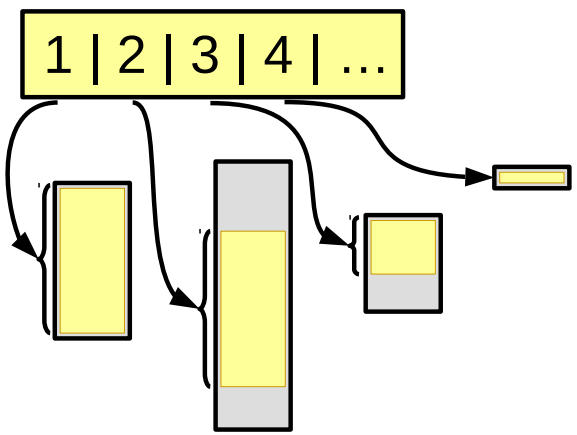
Causal reads

Separate data and metadata

Immutable data
Variable sized blobs

Metadata
Controls file contents, properties and filesystem structure

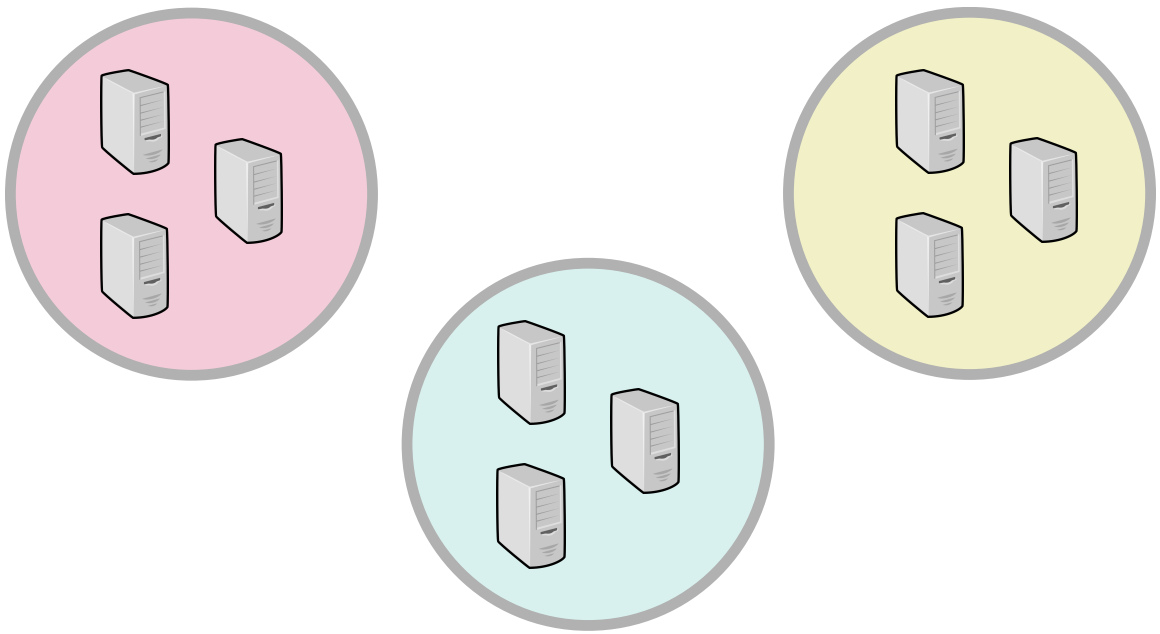
Metadata refers to data blobs



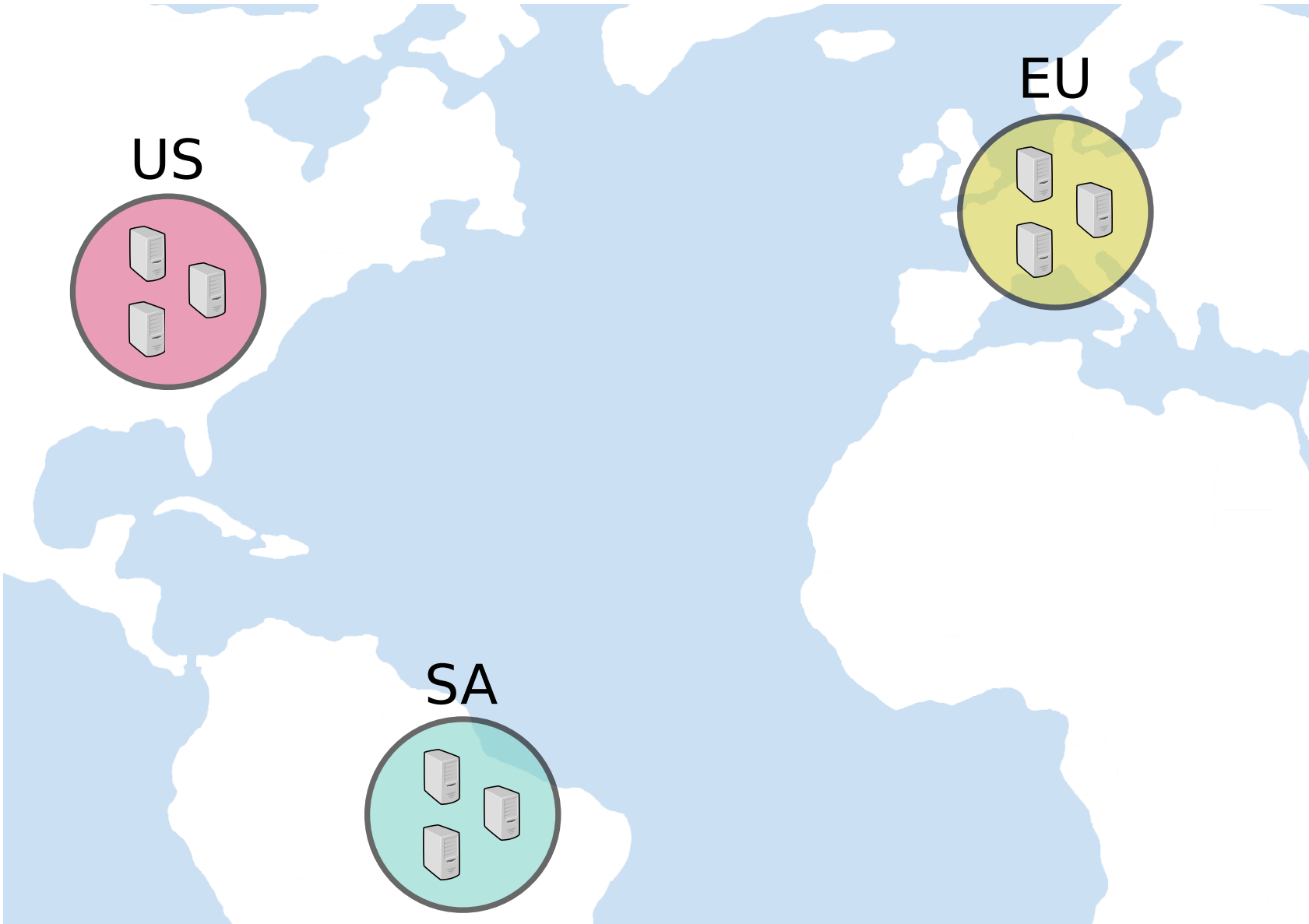
Partial replication

Immutable data is simple to replicate consistently

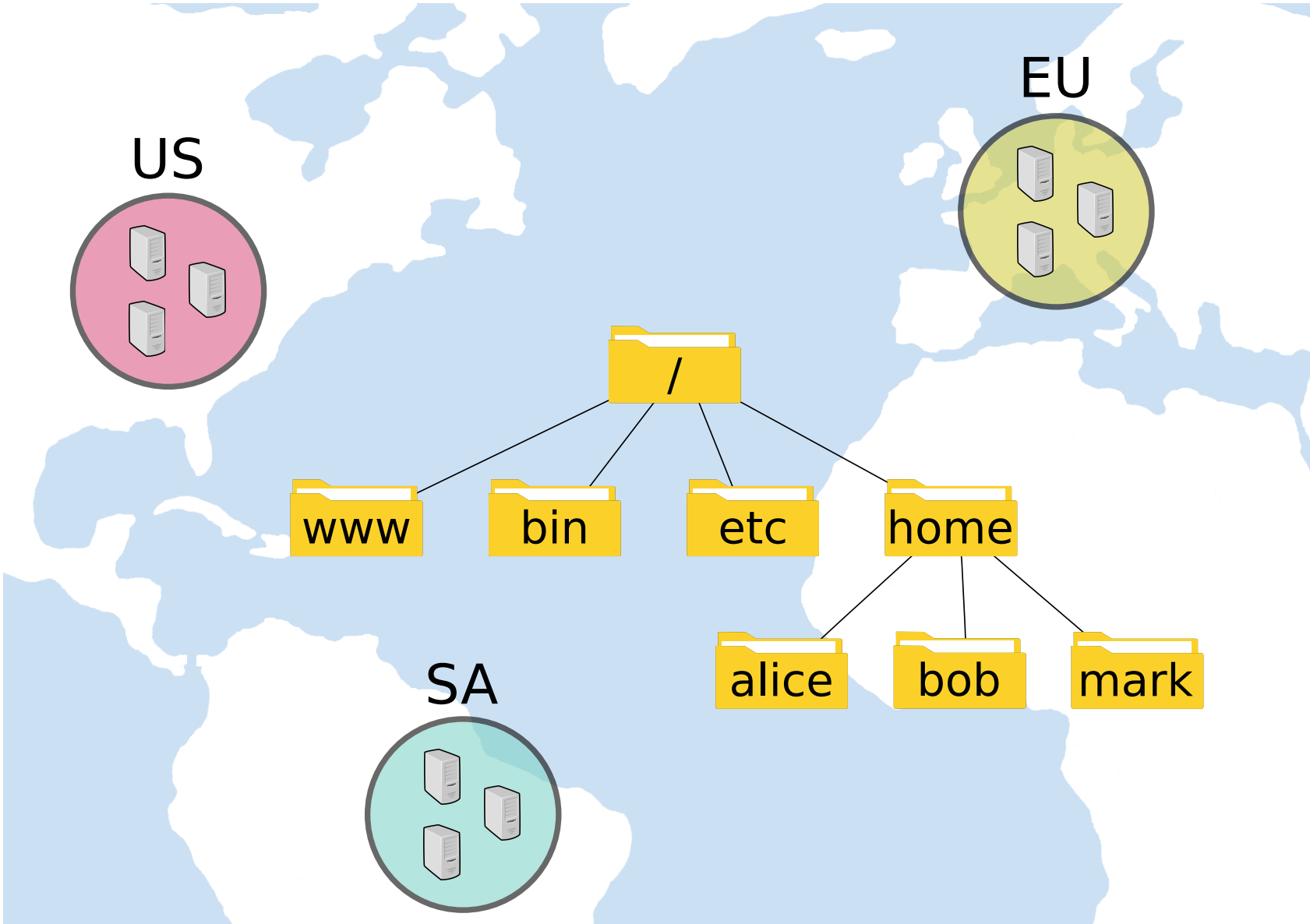
Metadata is partitioned between replica groups (i.e., partitions)



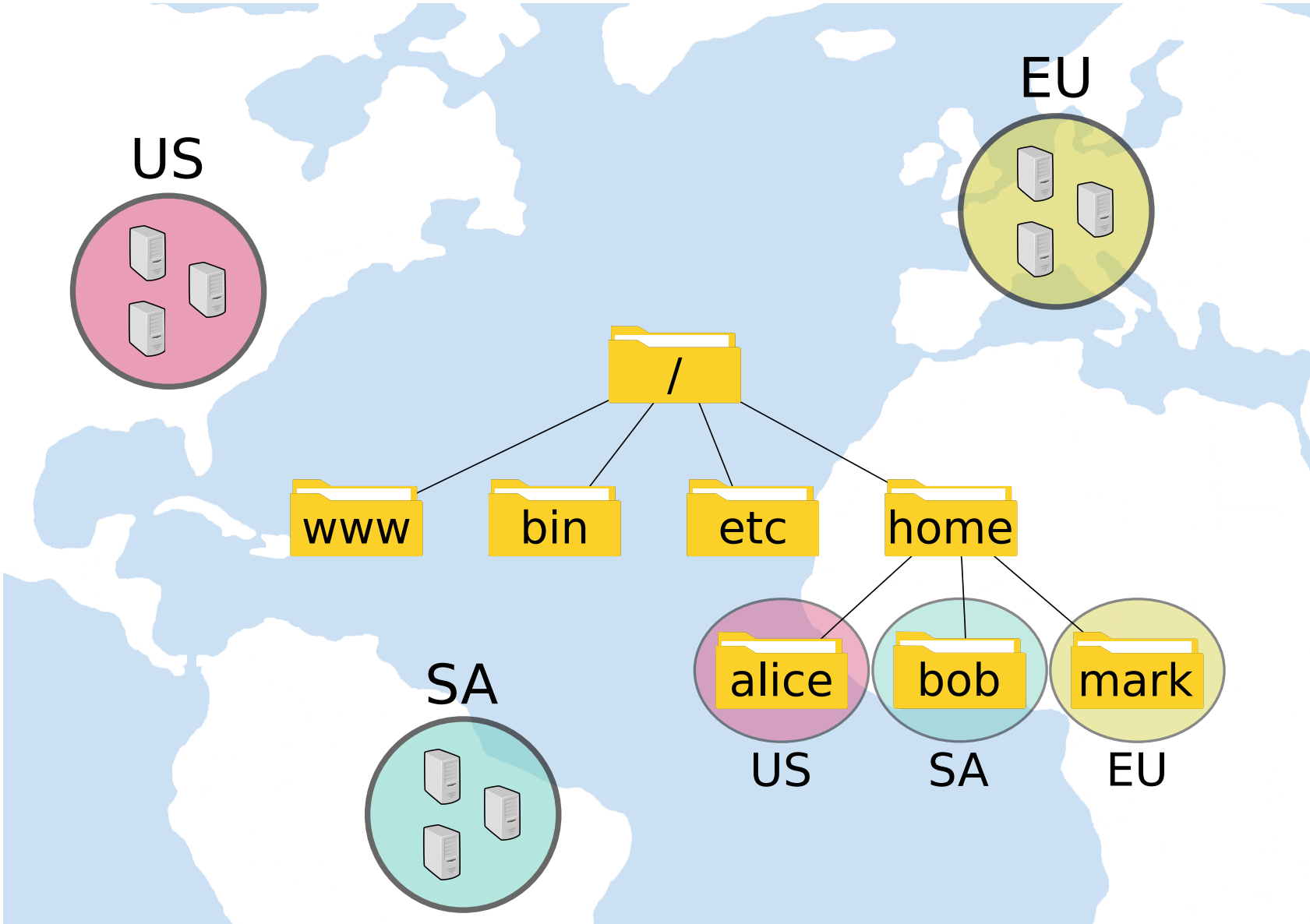
Partial replication



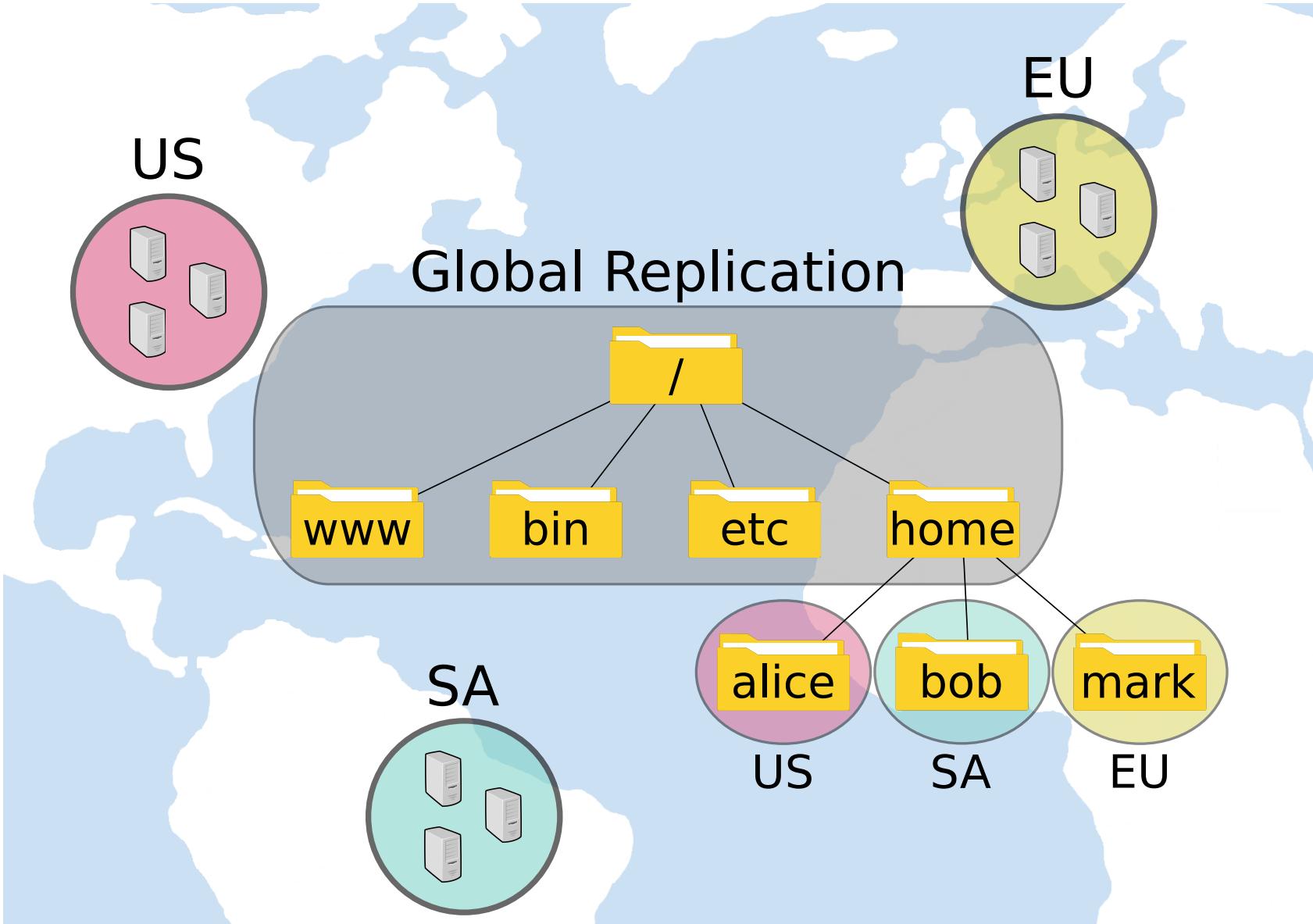
Partial replication



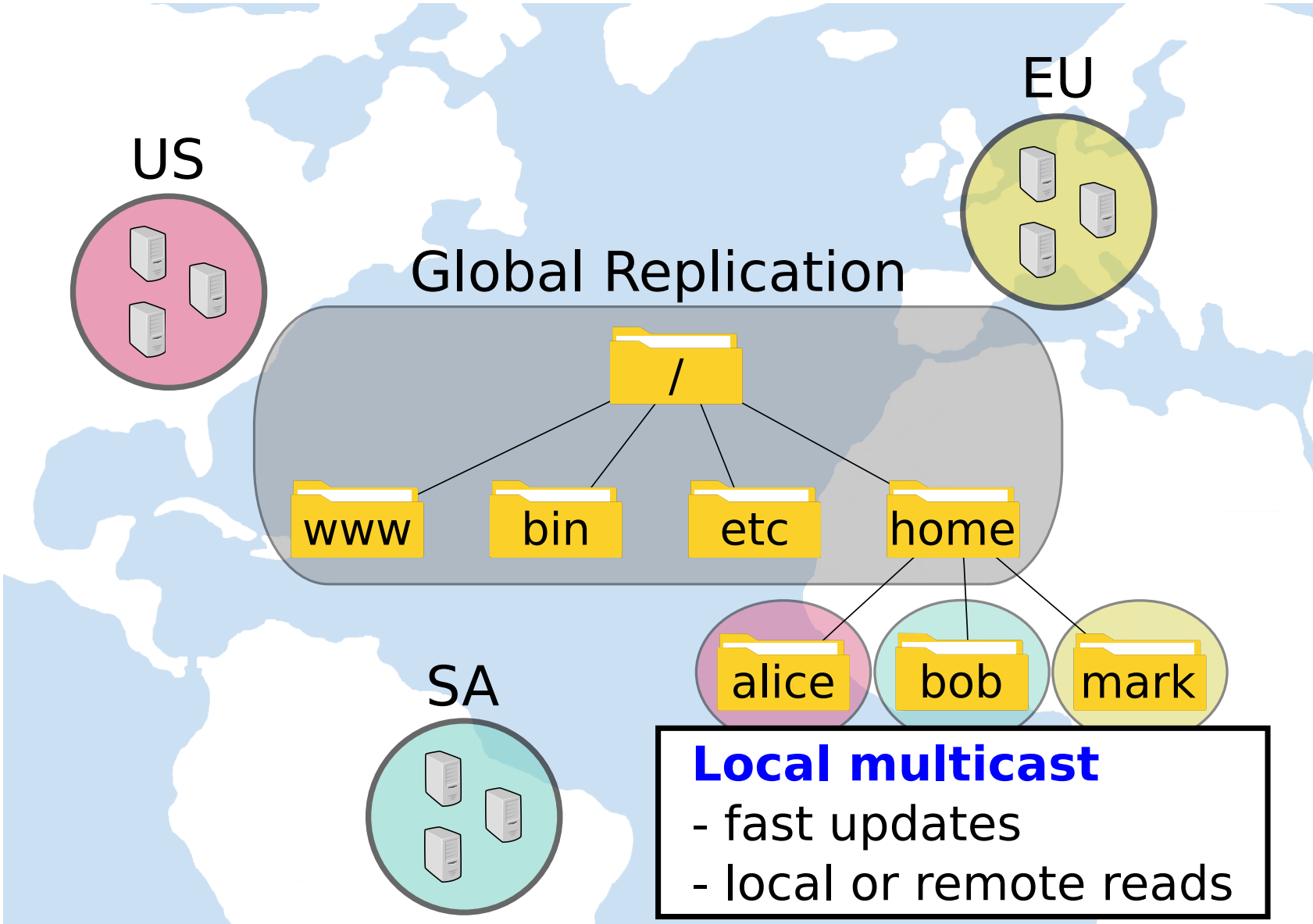
Partial replication



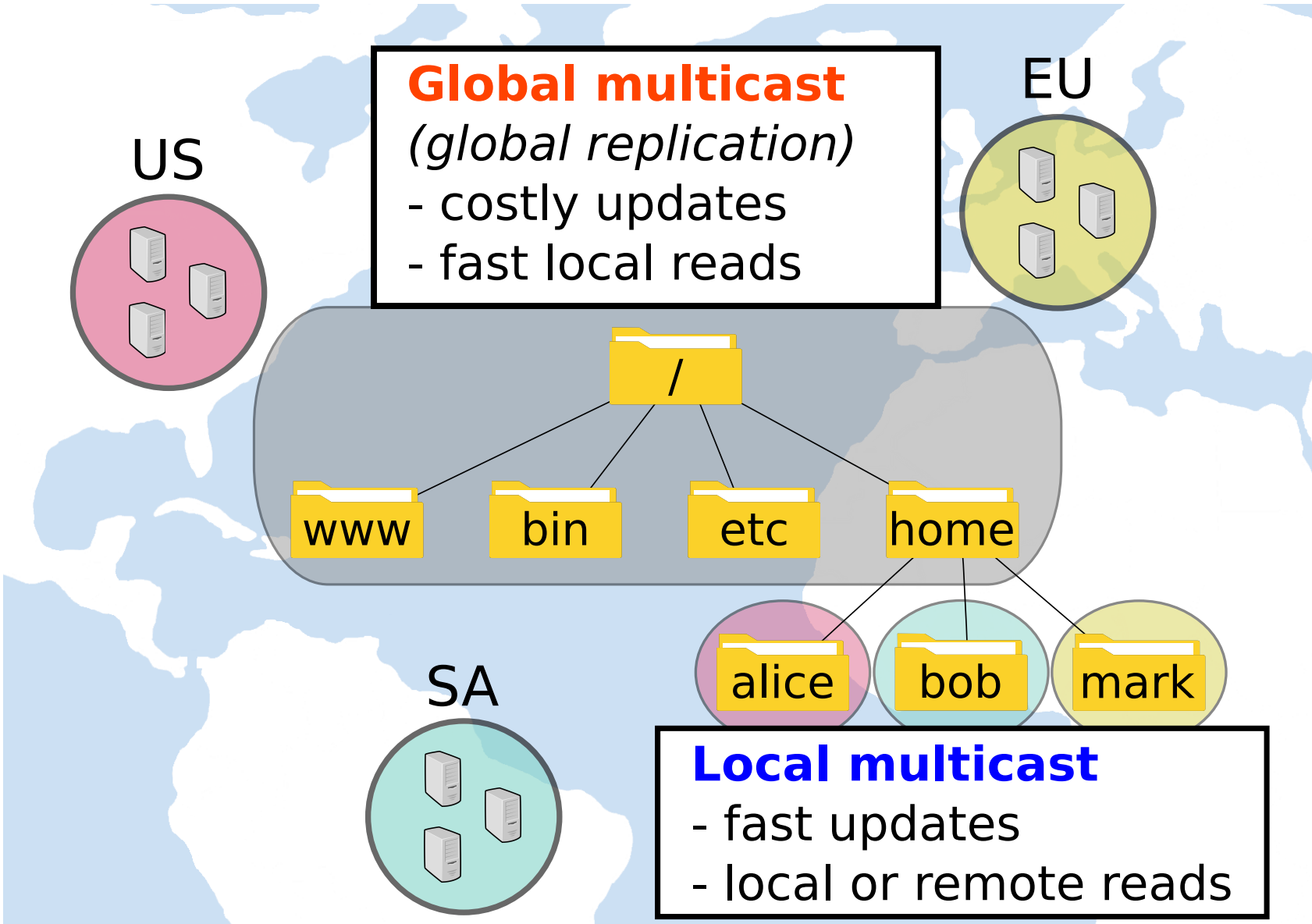
Partial replication



Partial replication



Partial replication



Partial ordering

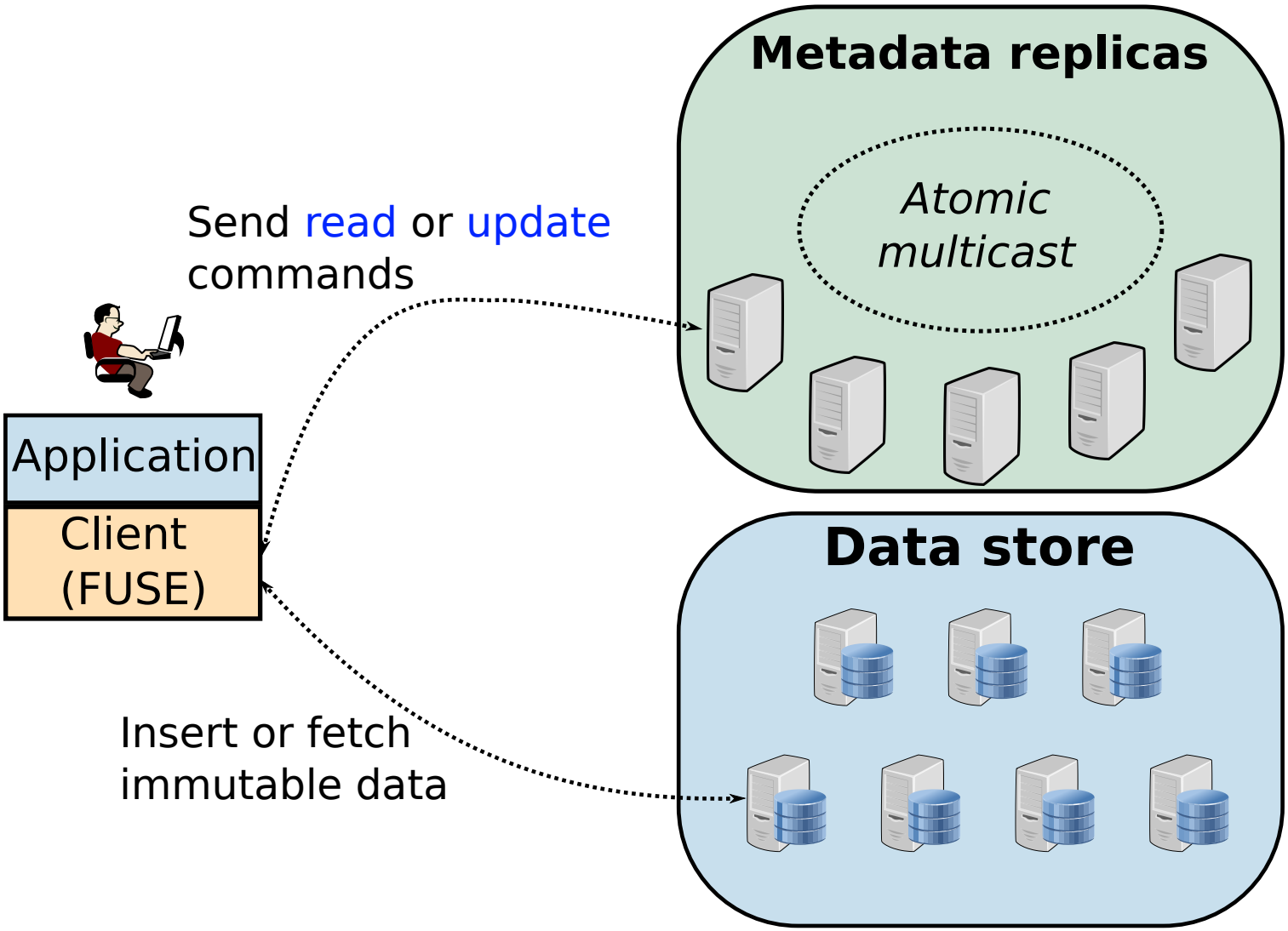
GlobalFS exploits atomic multicast

Atomic delivery to groups of processes

Partial ordering: messages for different groups don't have to be ordered betweem themselves

Partial ordering is critical for scalability

Architecture



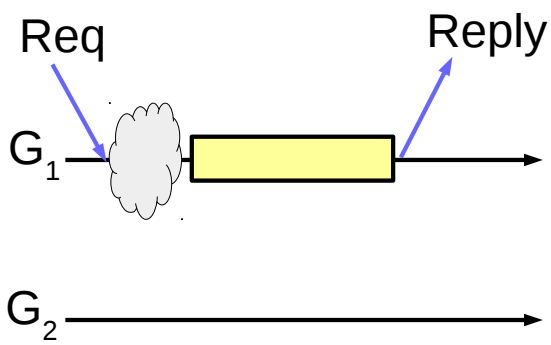
Consistent update operations

Step 1 Write data blobs to data store
Step 2 Issue a metadata update

Consistent update operations

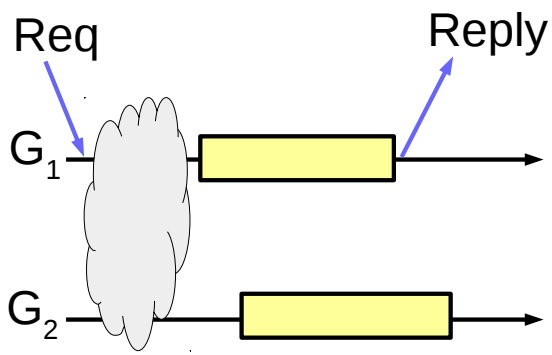
Step 1 Write data blobs to data store
Step 2 Issue a metadata update

Single-partition



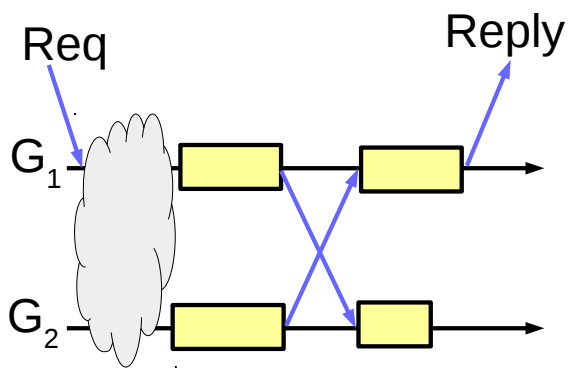
write to file in G_1

Uncoordinated multi-partition



write to file in $\{G_1, G_2\}$

Coordinated multi-partition



move file from G_1 to G_2

Causal read operations

Causally related updates are seen in the same order

e.g., operations done by the same client

Causal read operations

Causally related updates are seen in the same order

e.g., operations done by the same client

Client A

Creates an image `cat.jpg`

Modifies a page `pets.html` to include the image `cat.jpg`

Causal read operations

Causally related updates are seen in the same order

e.g., operations done by the same client

Client A

Creates an image `cat.jpg`

Modifies a page `pets.html` to include the image `cat.jpg`

Client B

Opens the `pets.html` page and finds a broken image reference

Where is the cat?

Causal read operations

Step 1 Contact a metadata replica for a list of blob ids
Step 2 Get the data from the data store

Approach inspired by vector clocks

Vector is composed of one counter per replica group

Evaluation

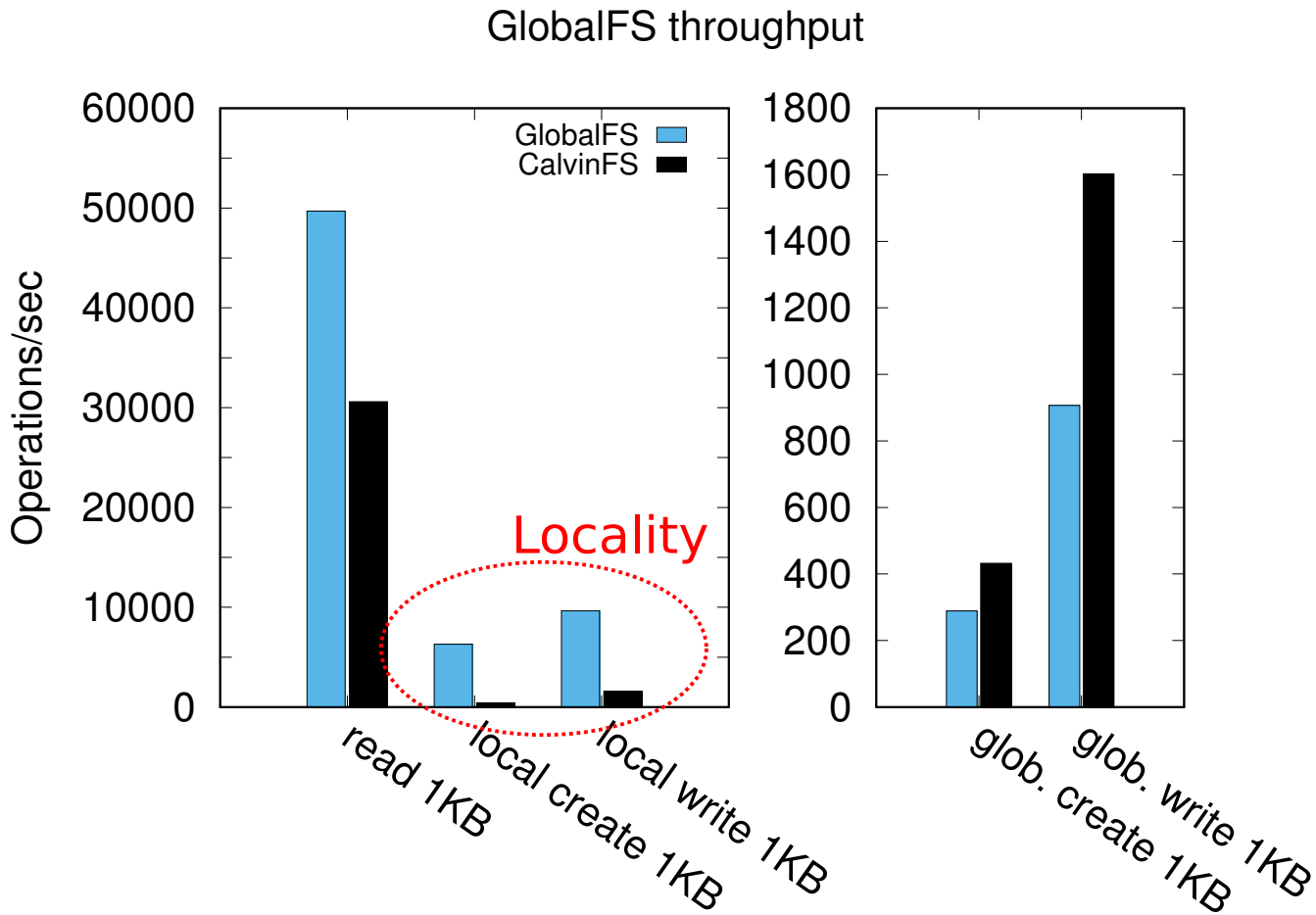
Complete prototype in Java
https://github.com/pacheco/GlobalFS

Filesystem in Userspace (FUSE)

URingPaxos for atomic multicast

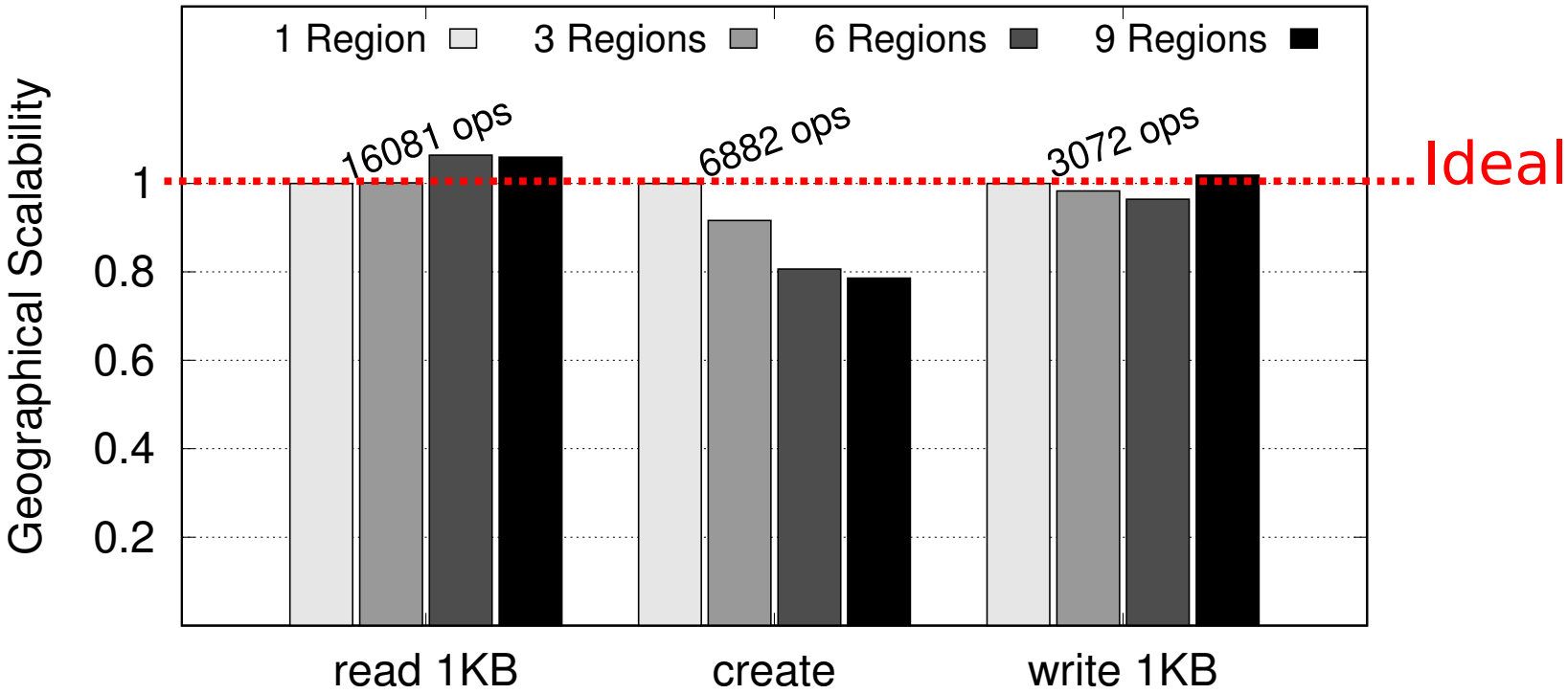
Global deployment using Amazon EC2

Maximum throughput by operation



3 region deployment US west, US east and Europe

Geographical scalability



Normalized throughput per region as more regions are added

9 regions uses all EC2 regions available at the time

GlobalFS: Summary

Strong consistency at global scale

Simple and familiar API (POSIX)

Flexible performance through partial replication and locality

Cheap causal read operations

GlobalFS: Summary

Strong consistency at global scale

Simple and familiar API (POSIX)

Flexible performance through partial replication and locality

Cheap causal read operations

Thank you!

Leandro Pacheco
pachecol@usi.ch