



redislabs
HOME OF REDIS



redhat.
MSP

MSP RHUG: Running Geo-distributed Applications with Redis Enterprise on Red Hat OpenShift

Brad Barnes, Solutions Architect

Agenda

- 1 A brief introduction to Redis and Redis Enterprise
- 2 ~~Deep~~ Shallow ~~dive~~ *dip* into geo-distributed applications
- 3 Redis Enterprise active-active on OpenShift
- 4 Demo GeoDogs 🙌

A Brief Introduction to Redis and Redis Enterprise



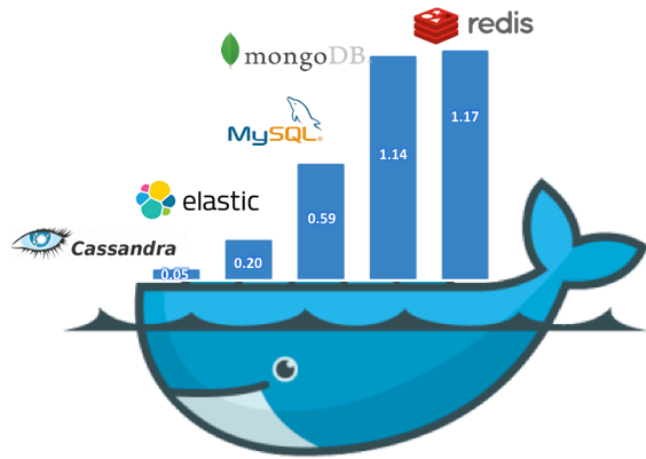
redislabs
HOME OF REDIS

Our Roots Are in Open Source

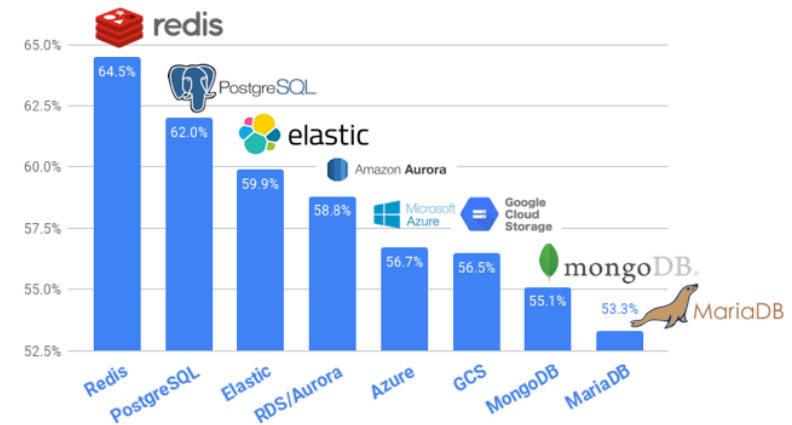


An **In-memory open source database**, supporting a variety high performance operational, analytics or hybrid use case.

Growing in Popularity and Leadership

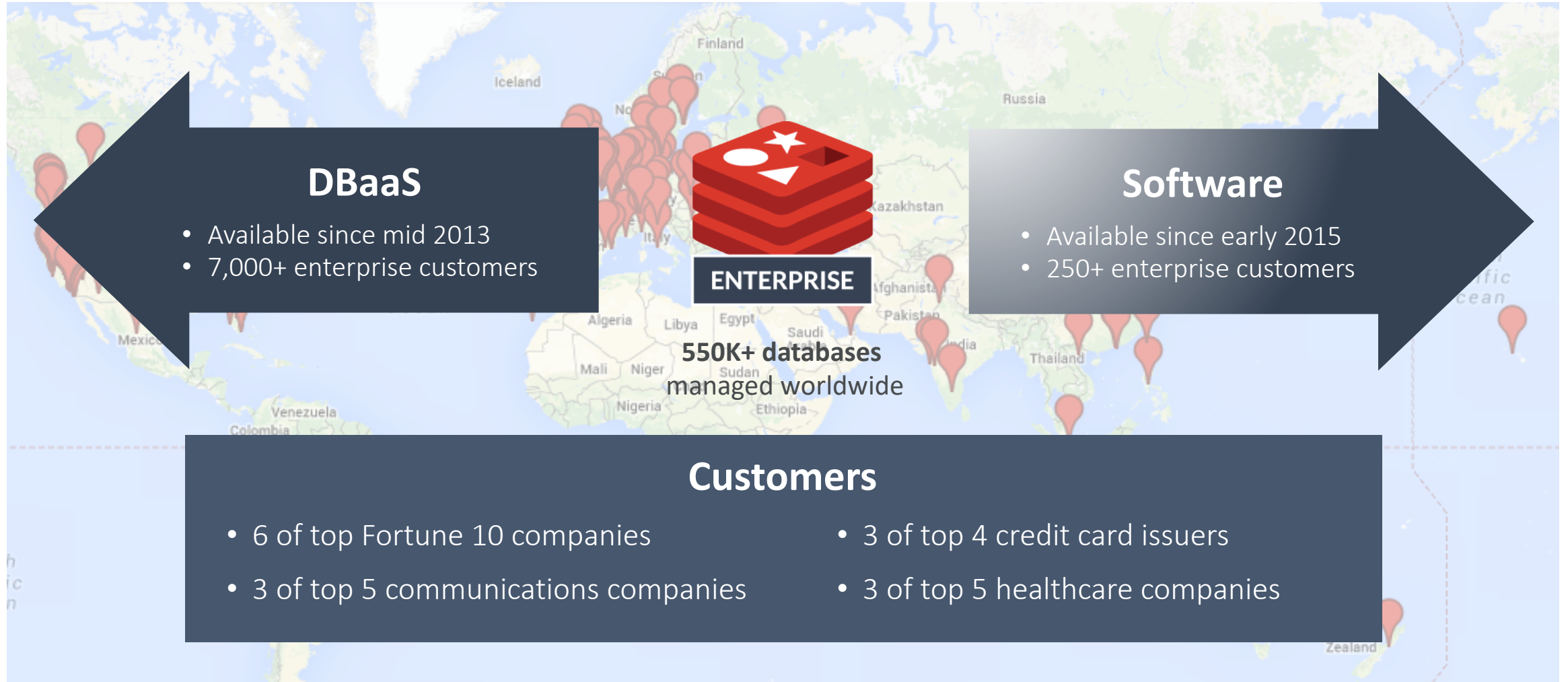


Most Popular Database Container
first to reach 1B+ containers
launched



Most Loved Databases 2017 & 2018
Stack Overflow survey among
>100,000 developers

Redis Enterprise

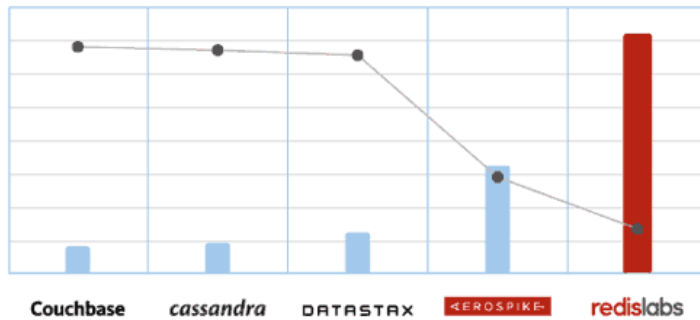


Redis Top Differentiators

1

Performance

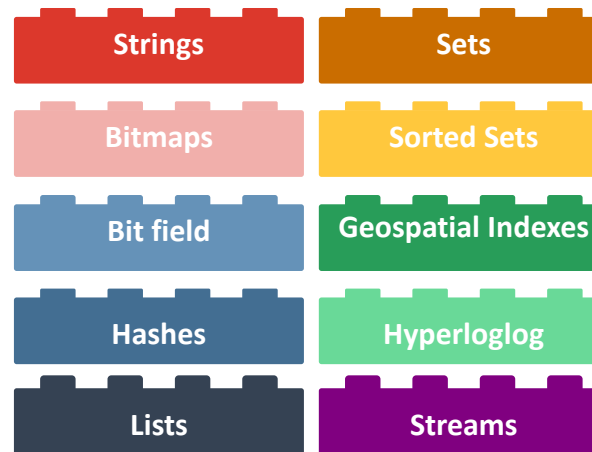
NoSQL Benchmark



2

Simplicity

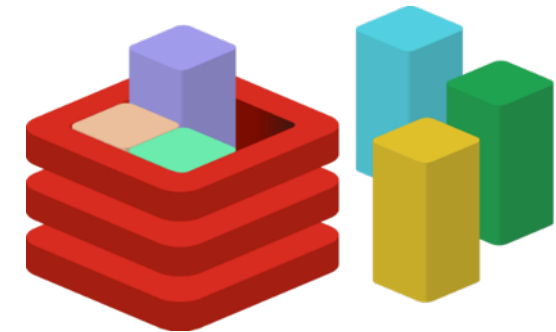
Redis Data Structures



3

Extensibility

Redis Modules



Redis Enterprise : A Unique Primary Database

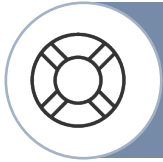
FAST

RELIABLE

FLEXIBLE



HIGHEST PERFORMANCE,
LINEAR SCALING



HIGH AVAILABILITY WITH INSTANT
FAILOVER



DURABILITY AT MEMORY SPEEDS



ACTIVE-ACTIVE GEO DISTRIBUTION
(CRDT-BASED)



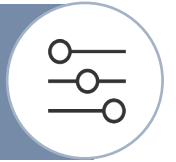
BUILT-IN HIGH PERFORMANCE
SEARCH



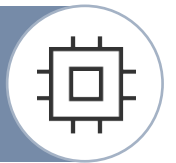
MULTI-MODEL
(ML/AI, Time series, Graph, Document, Search)



FLEXIBLE DEPLOYMENT OPTIONS
(CLOUD, ON-PREM, HYBRID)



INTELLIGENT TIERED DATA ACCESS
(RAM & FLASH MEMORY)



Uniquely Suited to Modern Use Cases

A full range of capabilities that simplify and accelerate next generation applications



Real Time Analytics



User Session Store



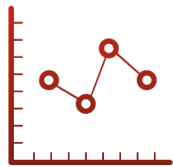
Real Time Data Ingest



High Speed Transactions



Job & Queue Management



Time Series Data



Complex Statistical Analysis



Notifications



Distributed Lock



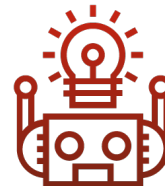
Content Caching



Geospatial Data



Streaming Data



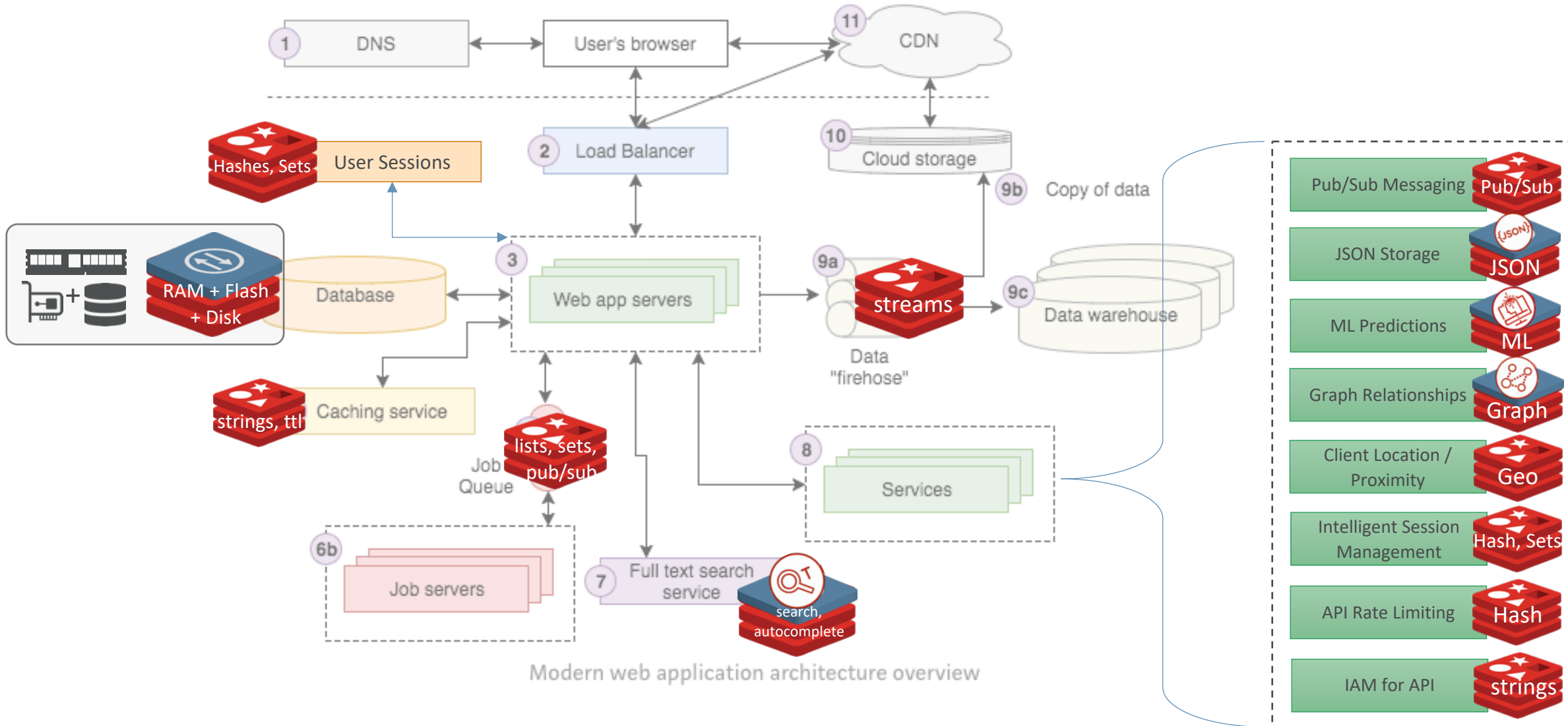
Machine Learning



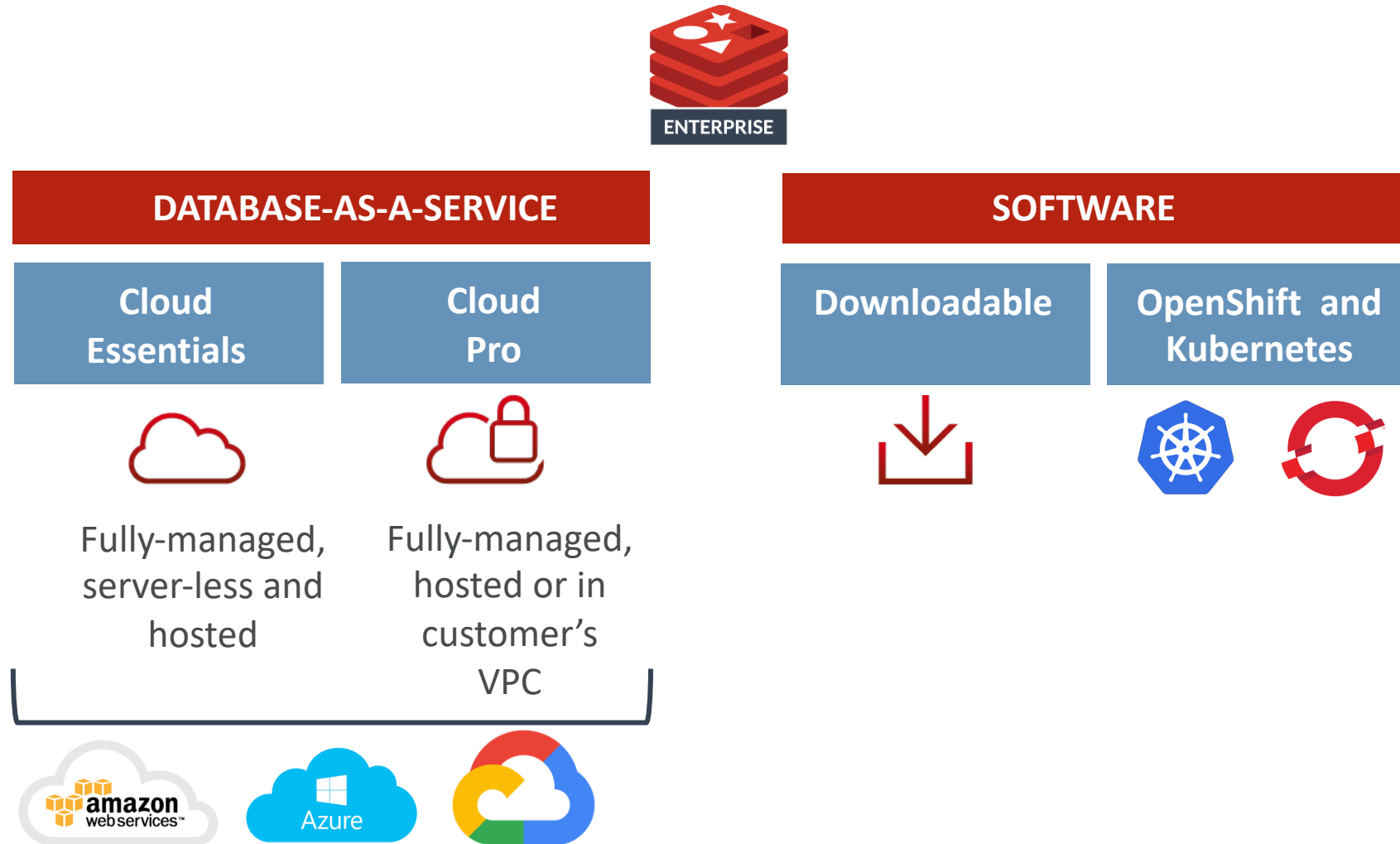
Search

Web App Architecture 101 medium.com

Animated slide.

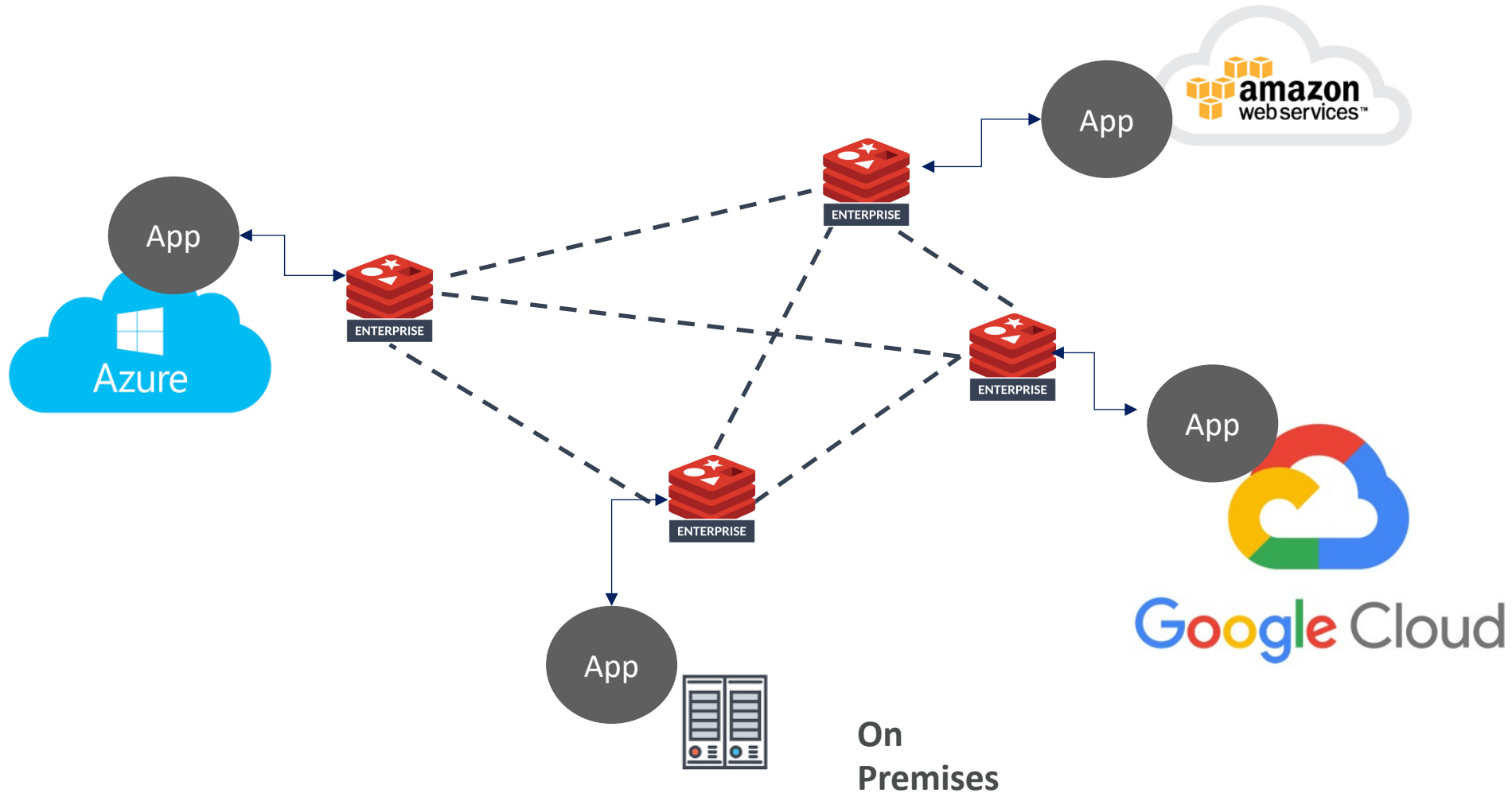


Multiple Delivery Models



Multi-Cloud and Hybrid Cloud Support

Active-Active or Active-Passive



SOFTWARE ARCHITECTURE

The topics to watch in software architecture

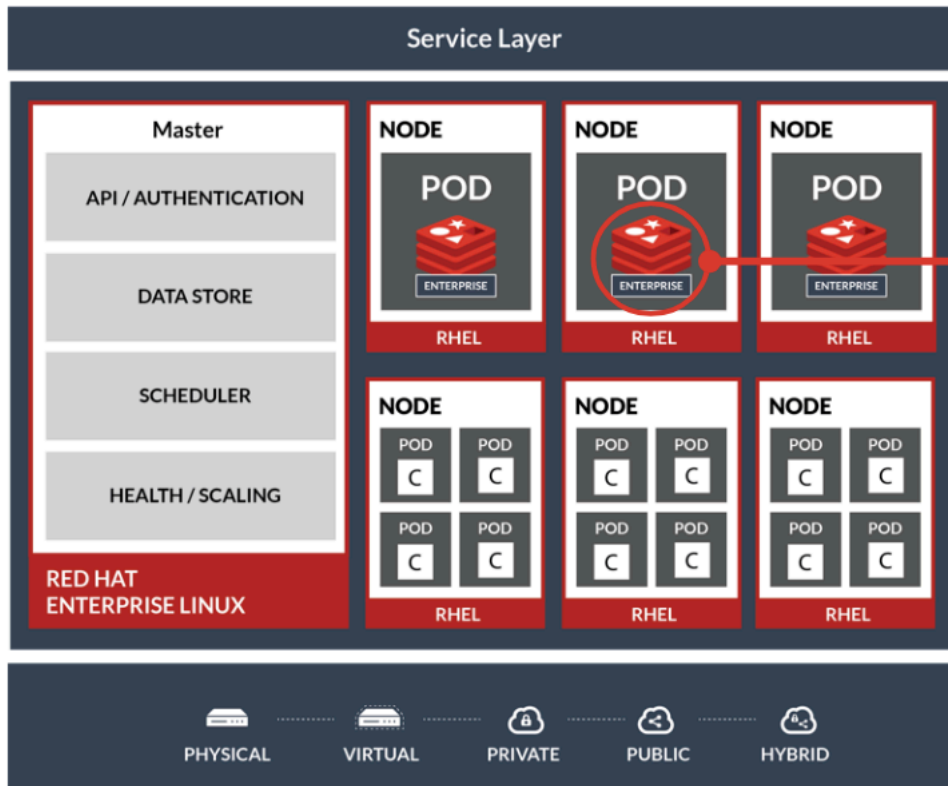
Microservices, serverless, AI, ML, and Kubernetes are among the most notable topics in our analysis of proposals from the O'Reilly Software Architecture Conference.

By Roger Magoulas and Chris Guzikowski. May 16, 2019


Redis Enterprise and OpenShift Integration Architecture



OpenShift + Redis Enterprise



- Node auto-healing
- Node scaling

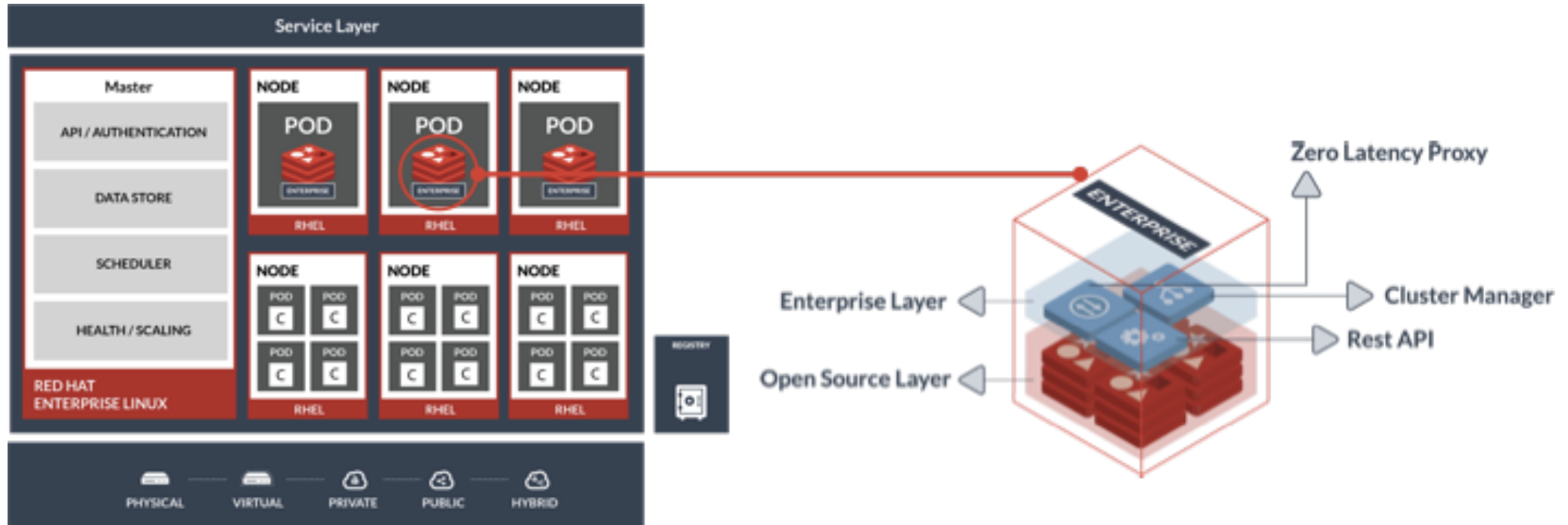


- Failover & shard level scaling
- Configuration & monitoring



- Service discovery
- Upgrade

Redis Enterprise on Red Hat OpenShift



Provision, deploy and operate Redis Enterprise database seamlessly on the OpenShift platform

Predictable performance

Seamless scale

Built-in orchestration

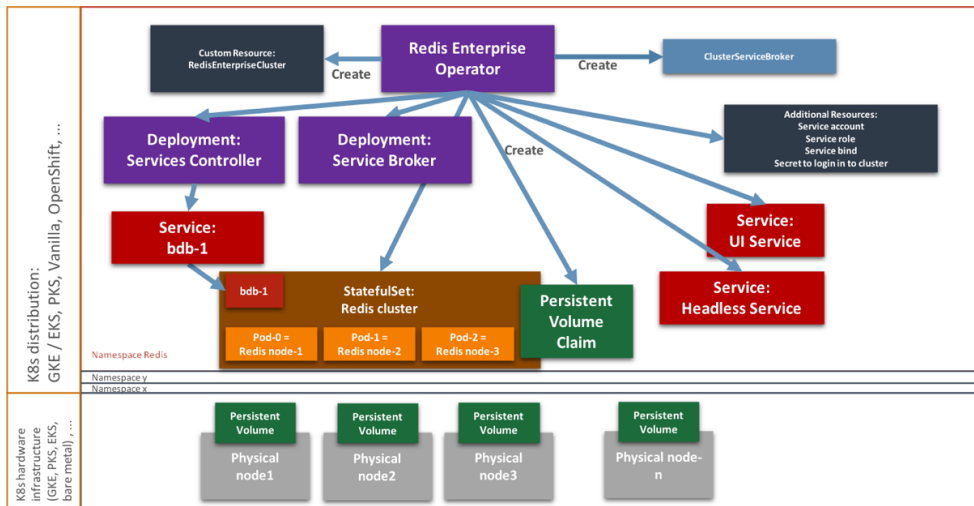
Resilience to failures

Cloud portability

Integrations with OpenShift

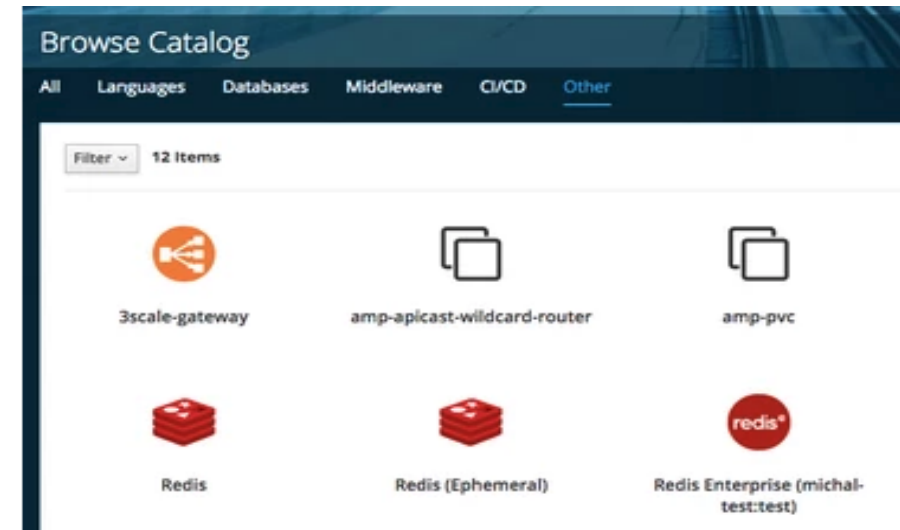
Redis Enterprise Operator

- Deploy and maintain a Redis Enterprise Cluster in k8 and OpenShift.



Redis Enterprise Service Broker

- Makes Redis service plans available in the OpenShift Service Catalog.



Running Geo-Distributed Applications with Redis Enterprise on OpenShift



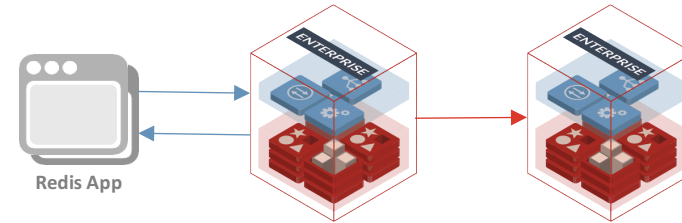
redislabs
HOME OF REDIS

Replication Techniques with Redis Enterprise

1. Active – Passive

Passive server is a cold standby

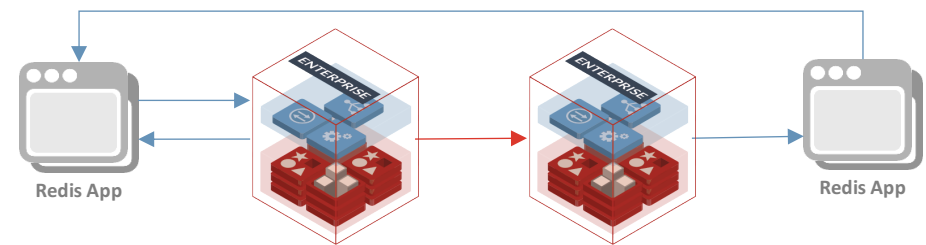
Uses: High Availability, Disaster Recovery, Data Durability



2. Active – Read-replica

Read-replica is available in the read-only mode

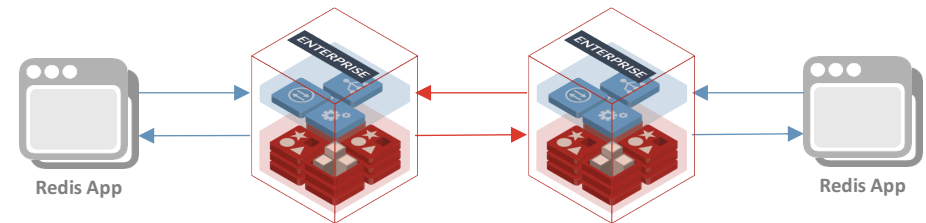
Uses: Distributed caching, offline data analytics, content distribution



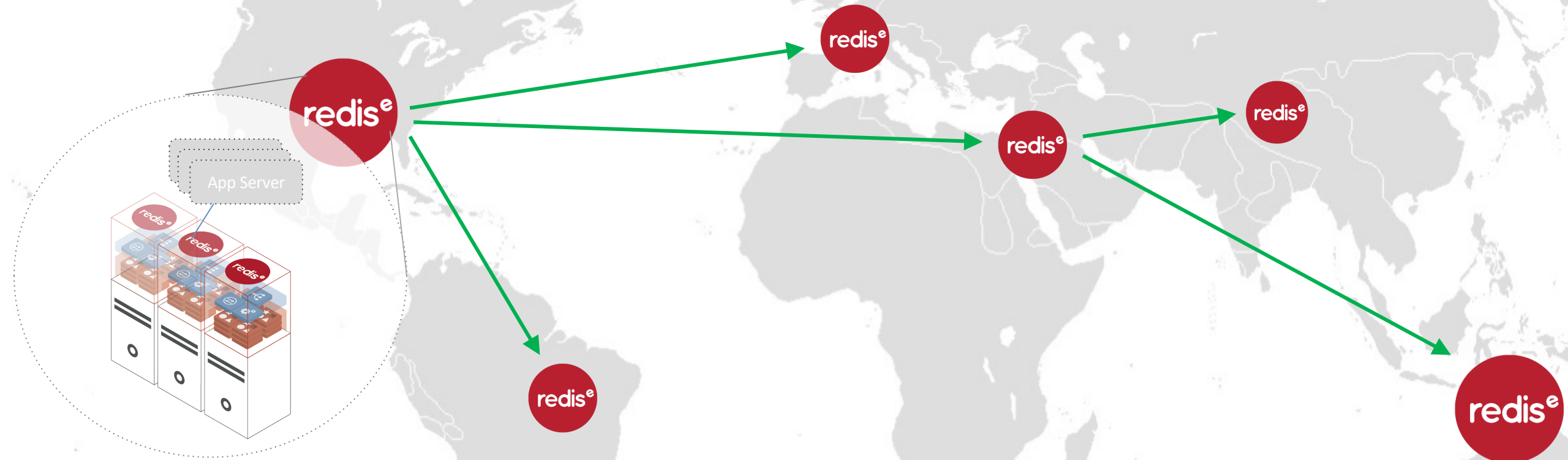
3. Active – Active

All database instances are available for read and write operations

Uses: Local latencies for geo-distributed apps, load distribution, data consolidation



Replica Of: Geo Distribution for Fast Local Data Access



Geo Distribution for Local Data Access (CDN Like)

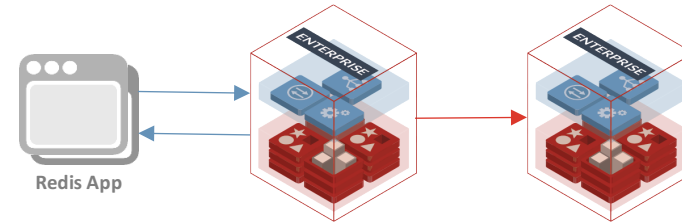
- Read local copy with low latency, instead of crossing borders
- Push updates to all regions with fast, memory based replication

Replication Techniques with Redis Enterprise

1. Active – Passive

Passive server is a cold standby

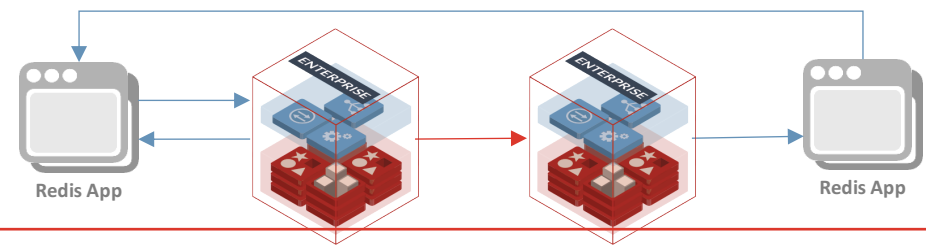
Uses: High Availability, Disaster Recovery, Data Durability



2. Active – Read-replica

Read-replica is available in the read-only mode

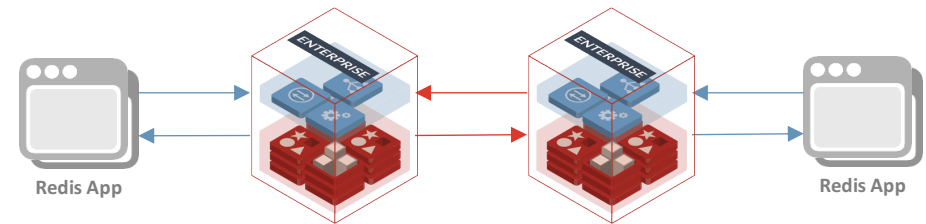
Uses: Distributed caching



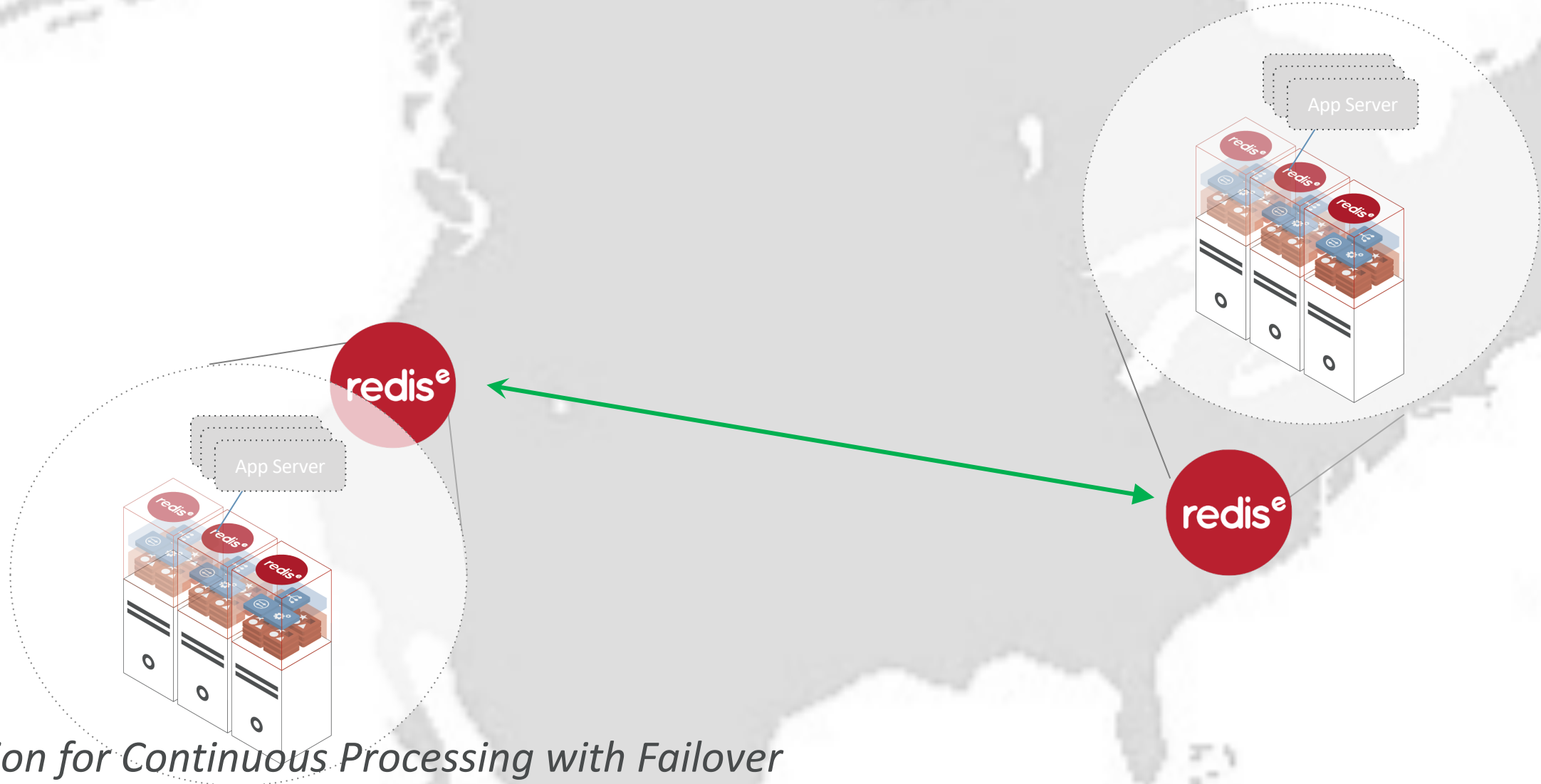
3. Active – Active

All database instances are available for read and write operations

Uses: Local latencies for geo-distributed apps, load distribution, data consolidation



Redis CRDTs: Active-Active Geo Distribution for Geo-Failover/Distribution



Geo Distribution for Continuous Processing with Failover

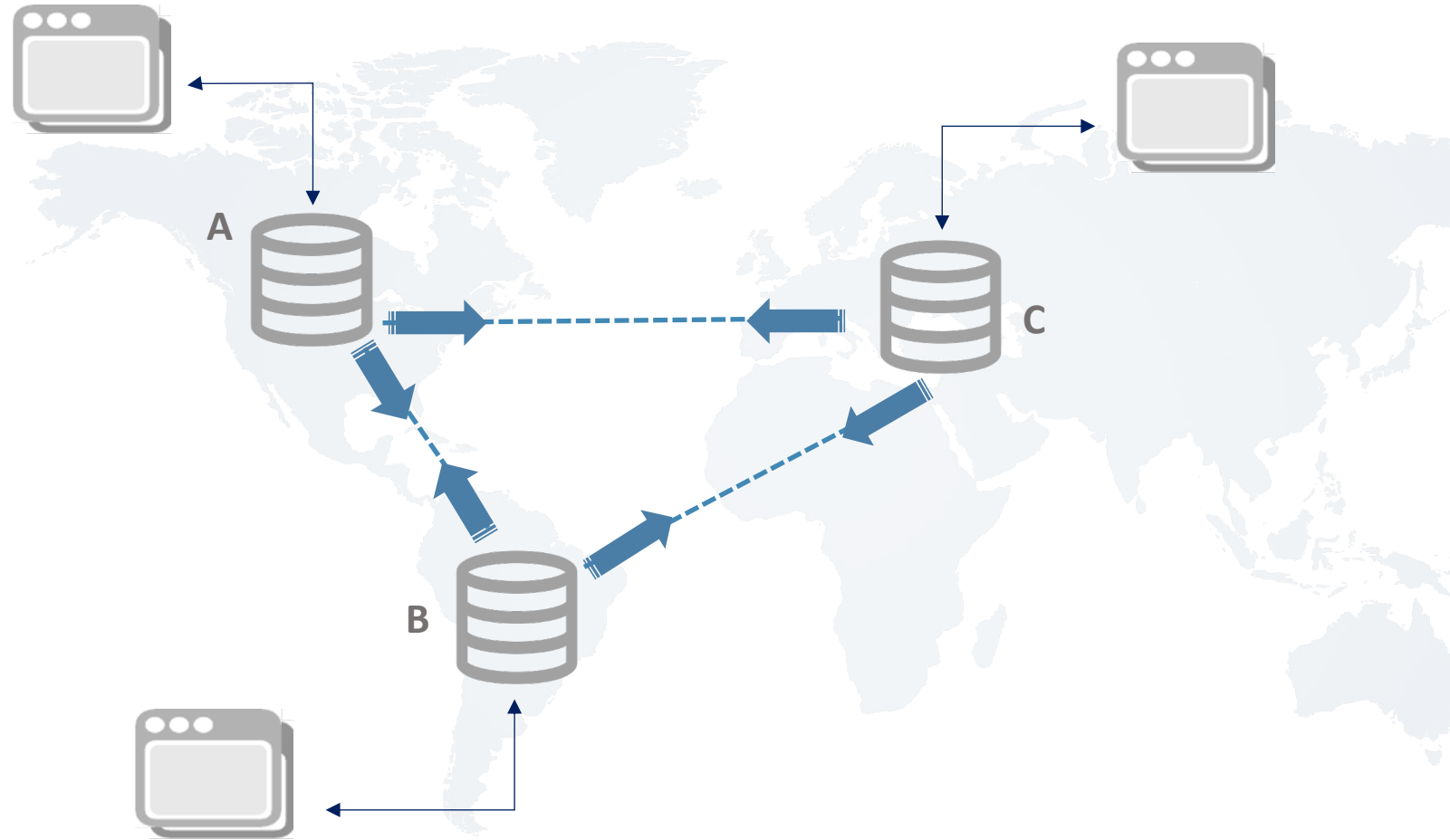
- Redis CRDTs for reliable handling of race conditions during geo-failover

Active – Active Geo Distributed Use Cases

- Migrating user sessions across data centers in real-time
- Handling Cluster failures or network outages
- High volume load distribution across multiple Geographically distributed databases
- Data consolidation in a microservices environment
- High frequency writes/reads in multiple regions
- Delivering local latencies for geographically distributed apps

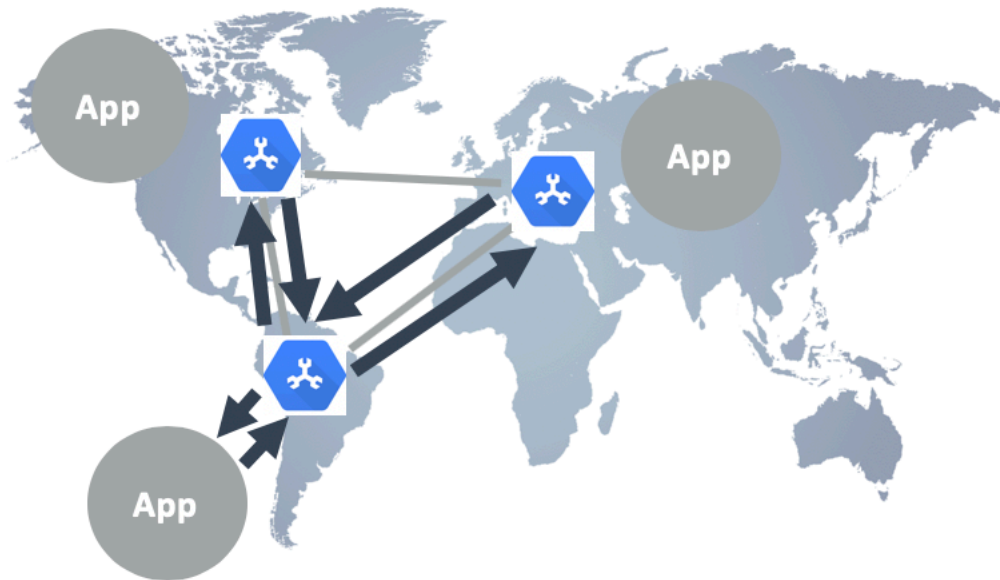
Two Problems with Geo-distributed Active-Active Databases

1. Latency
2. Conflict resolution



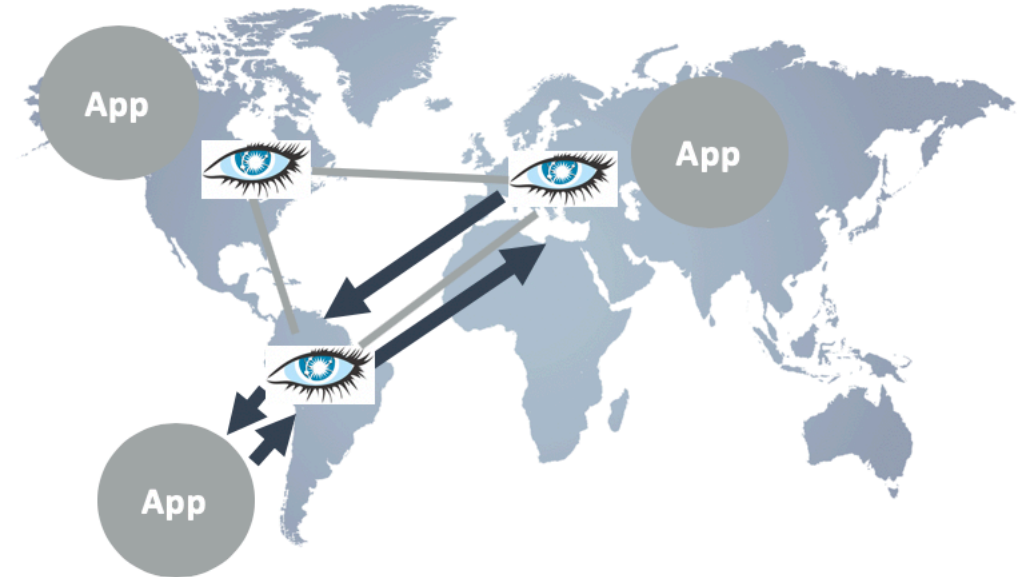
Problem 1: Active-Active: Existing Approaches are Slow

Spanner



Strong Consistency → 200msec

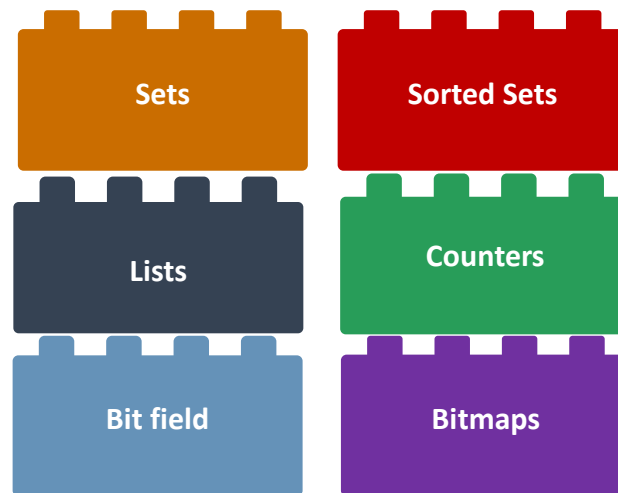
Cassandra/DynamoDB



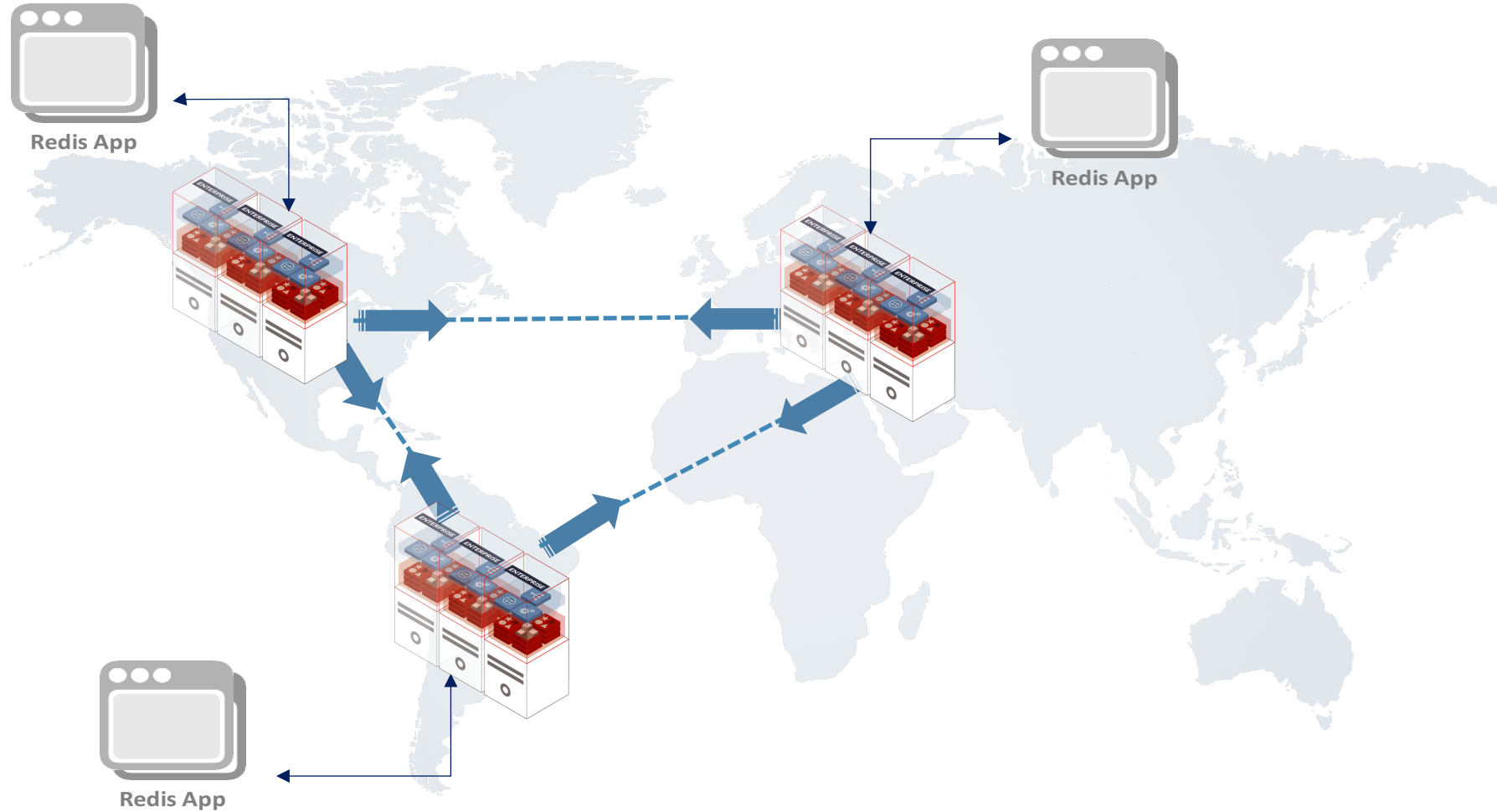
Eventual Consistency → 100msec

Problem 2: Conflict Resolution is Hard

- Application level solution → too complex to write
- LWW (Last Write Wins) → doesn't work for many data requirements








CRDTs – Solving The Active-Active Latency Problem

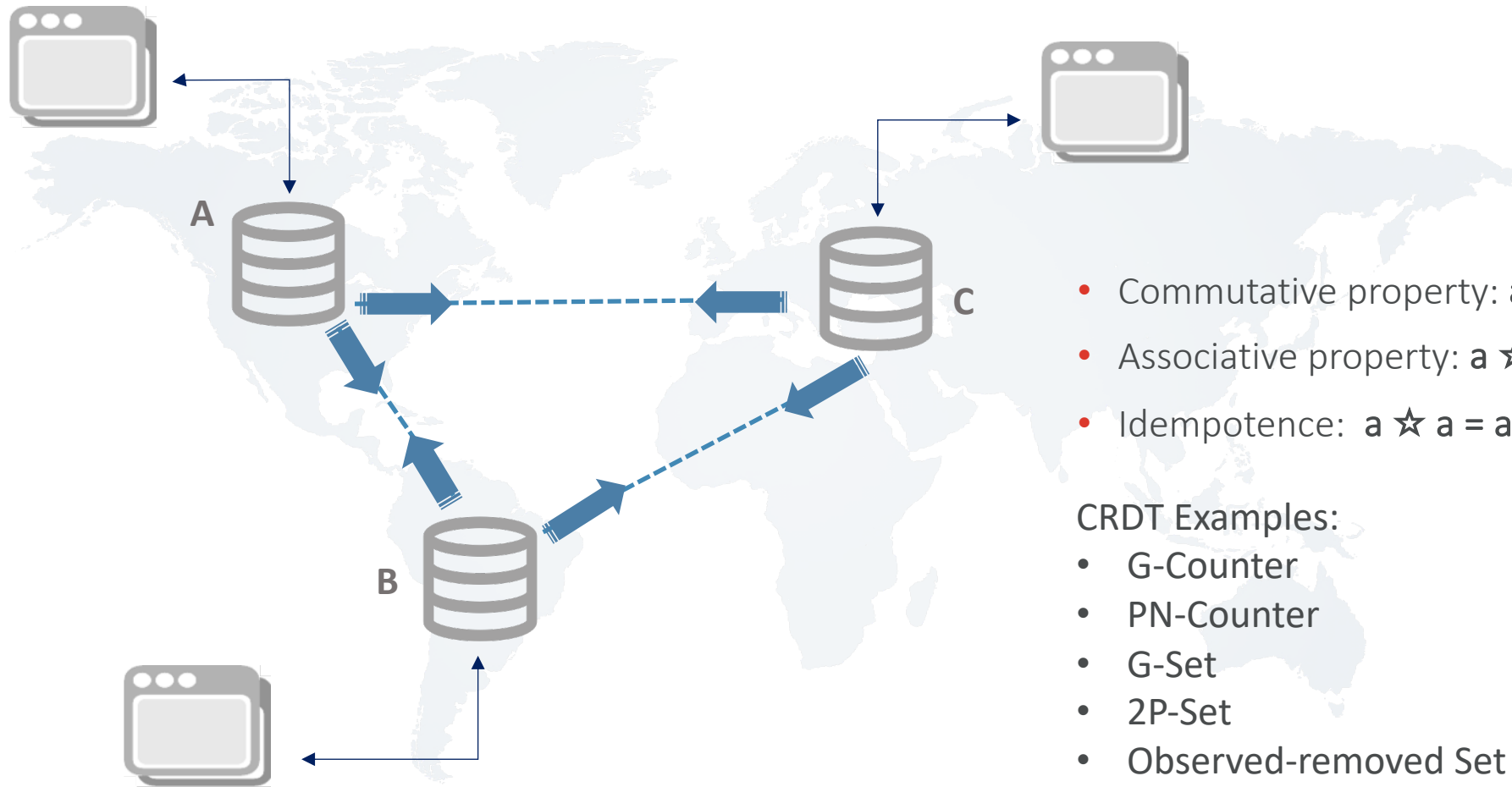


Strong Eventual Consistency → **1 msec**

Redis CRDTs – Solving The Conflict Resolution Problem

Data-Type	Conflict Resolution
 Strings	<ul style="list-style-type: none">• Counters: Conflict-free merge• Simple value: LWW (Last Write Wins)
 Hashes	<ul style="list-style-type: none">• New key/value: Conflict-free merge• Simple value: LWW (Last Write Wins)• Counters: Conflict-free merge
 Sets	<ul style="list-style-type: none">• Observed Removed Add Wins
 Sorted-Sets	<ul style="list-style-type: none">• Observed Removed Add Wins• Scores: Conflict-free merge
 Lists	<ul style="list-style-type: none">• Cumulative

How CRDTs work?



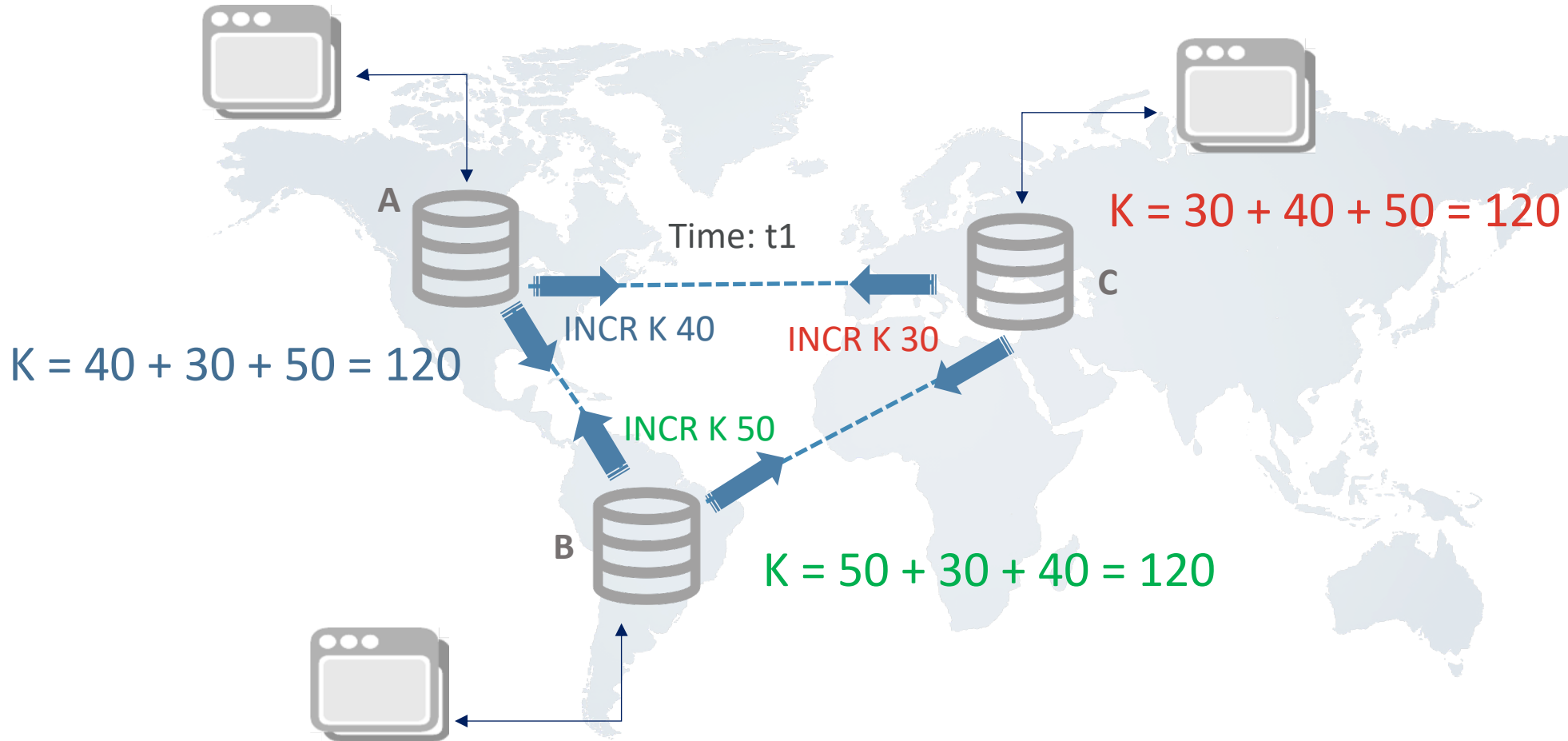
- Commutative property: $a \star b = b \star a$
- Associative property: $a \star (b \star c) = (a \star b) \star c$
- Idempotence: $a \star a = a$

CRDT Examples:

- G-Counter
- PN-Counter
- G-Set
- 2P-Set
- Observed-removed Set
- Register
- Sequence CRDTs

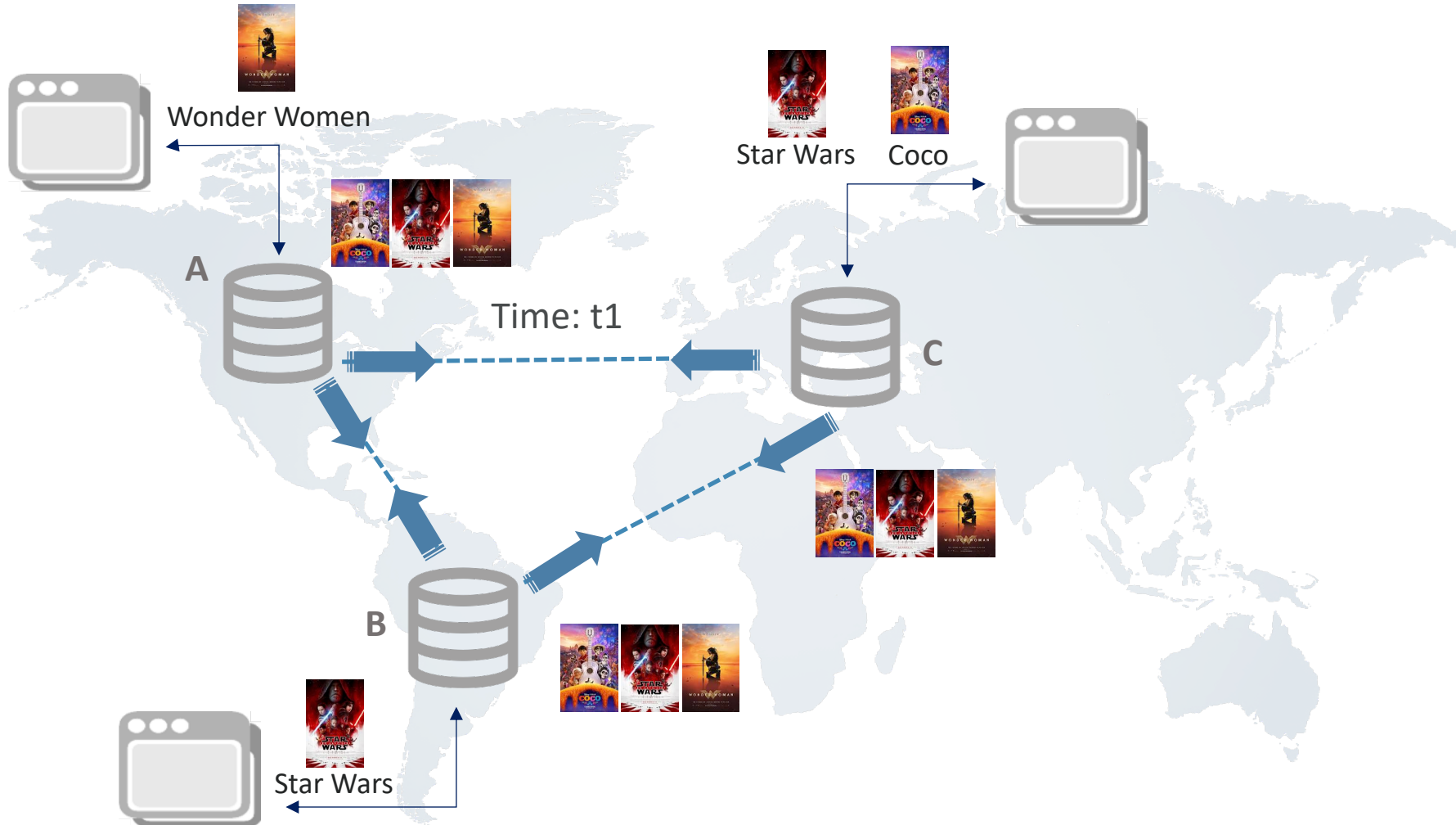
CRDT Example: Counter

Applies Commutative and Associative Properties



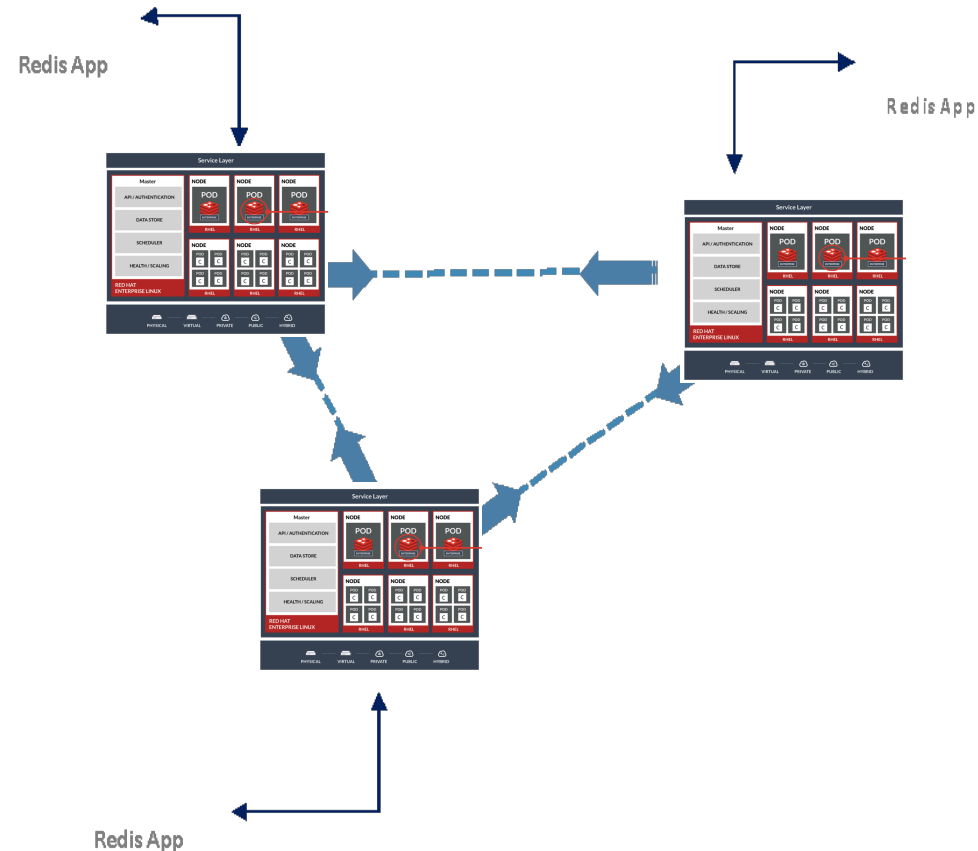
CRDT Example: Set

Applies idempotence



Redis CRDTs == CRDBs (Conflict free replicated databases)

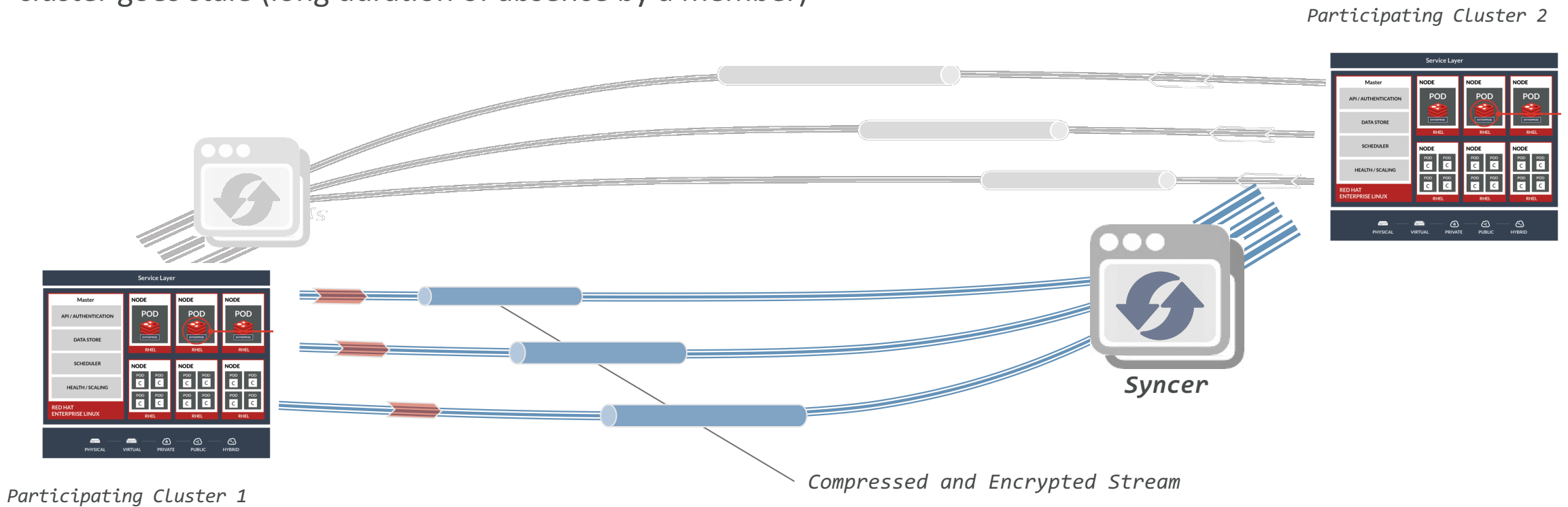
- CREATING a CRDB
 - Set global CRDB options
 - Initialize member CRDB on each participating cluster
 - In case of Error, Rollback
 - Establish bi-directional replication among all members



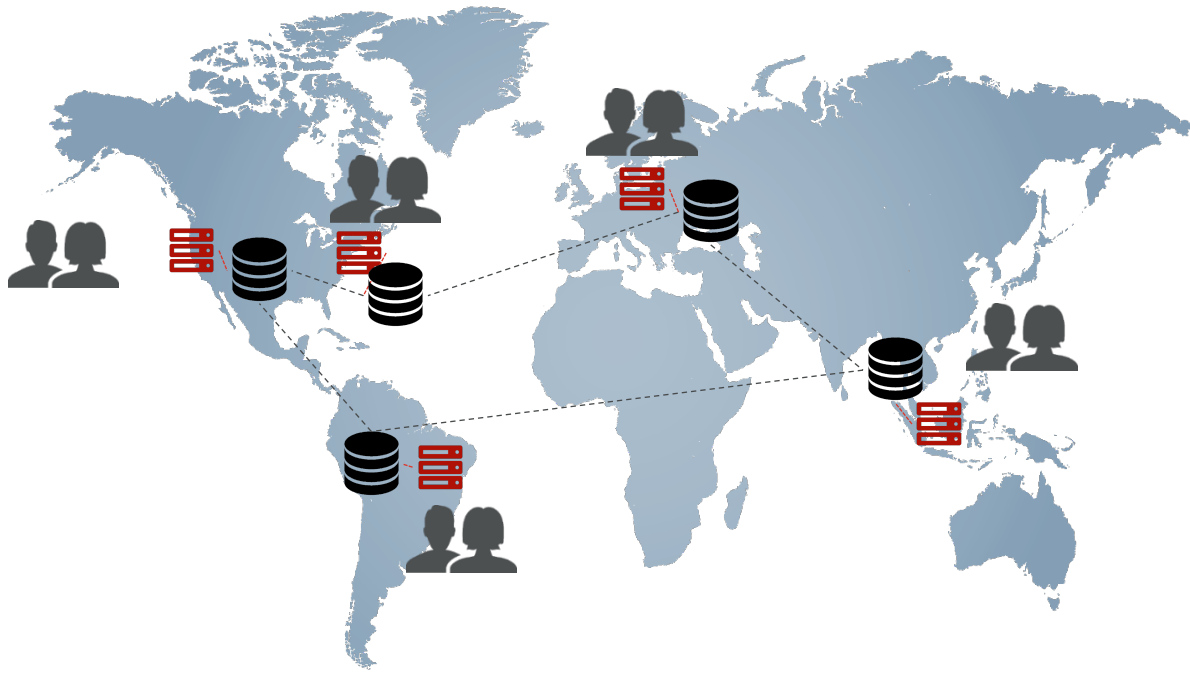
CRDB Architecture

Bi-directional Replication

- *Syncer* uses replicas to replicate operations in a streaming fashion
- Resume-able replication under failures but may resync if a participating cluster goes stale (long duration of absence by a member)



Redis Labs Active-Active with Eventual Consistency



- Read and write with low local latency: *sub-millisecond latencies*
- All databases eventually **converge automatically** to the same state (strong eventual consistency guaranteed)
- Automatic handling of conflicts with smart conflict resolution even for complex data types-CRDTs
- Develop as if it's a single app in a single geo, we take care of all the rest

DEMO!



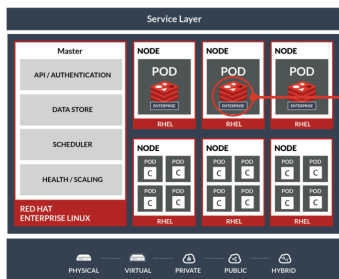
Redis Enterprise on Red Hat OpenShift Demo

App Server

In Node.Js
Simple Website
running in each
location

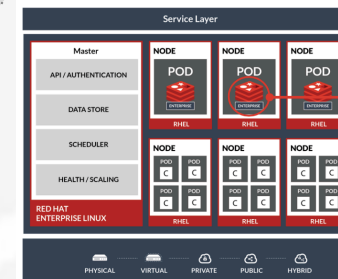
AWS East -- Ohio
Openshift Cluster: okde2.demo-
rlec.redislabs.com

AWS West – Oregon
Openshift Cluster: okdw1.demo-
rlec.redislabs.com



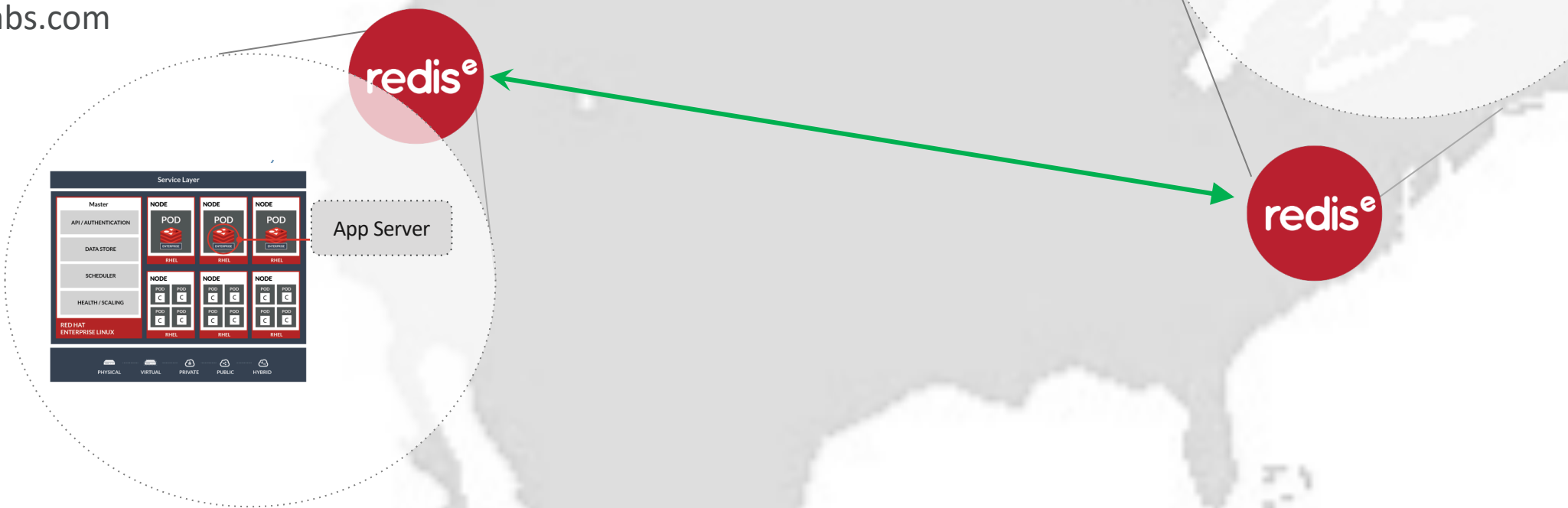
App Server

redis^e



App Server

redis^e



Categorize operations into CRDT-based and non CRDT-based

CRDT-based use cases	Non CRDT use cases
Counters	Certain Financial transactions
User activity tracker	Order processing
Session store	
Distributed caching	
Inventory management	
Disaster Recovery (Auto failover of DC)	



- Free Trial of Redis Enterprise
<https://redislabs.com/get-started/>
- Get started with Operators
- <https://github.com/operator-framework/getting-started>
- <https://www.operatorhub.io>
- Red Hat Container Catalog
<https://access.redhat.com/containers/#/vendor/redislabs>
- OperatorHub.io
<https://operatorhub.io/operator/redis-enterprise-operator.v0.0.1>

Questions



Learn more about Conflict-free Replicated Data Type (CRDT)

1. CRDT and Redis by Carlos Baquero
<https://youtu.be/ZoMlzBM0nf4>
2. Strong Eventual Consistency and CRDTs by Marc Shapiro
<https://youtu.be/ebWVLVhiaiY>