Redis and Redis^{e} Architecture

Cihan B. - VP of Product Management

Agenda

Redis and Redis^{e} (Redis Enterprise) Architecture

- Shards, Nodes, Clusters
- Replication in Redis^{e}
- Anatomy of Read/Write Operation

Scaling Applications with Redis^{e}

- Scaling Throughput – Redis^{e} Resharding and Rebalancing
- Scaling Connections - Redis^{e} Proxy
- Scaling Data Size – Redis^{e} Flash

High Availability with Redise

- Replication Architecture in Redis^{e}
  - Local Replication across the LAN
  - Cross-Geo Replication across the WAN

Architecture Overview

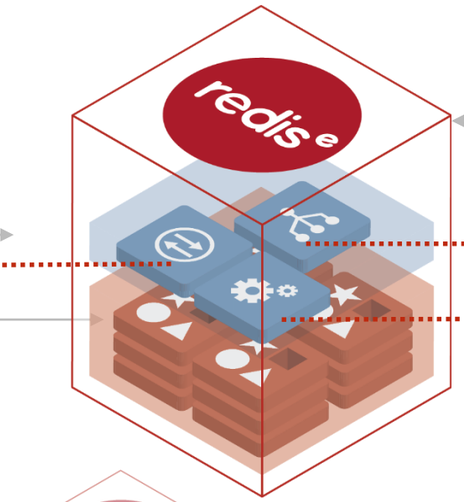
Redis Architecture

Redis^{e} - Open Source & Proprietary Technology

Enterprise Layer

Zero latency proxy

Open Source Layer



Redis^{e} Node

Cluster Manager

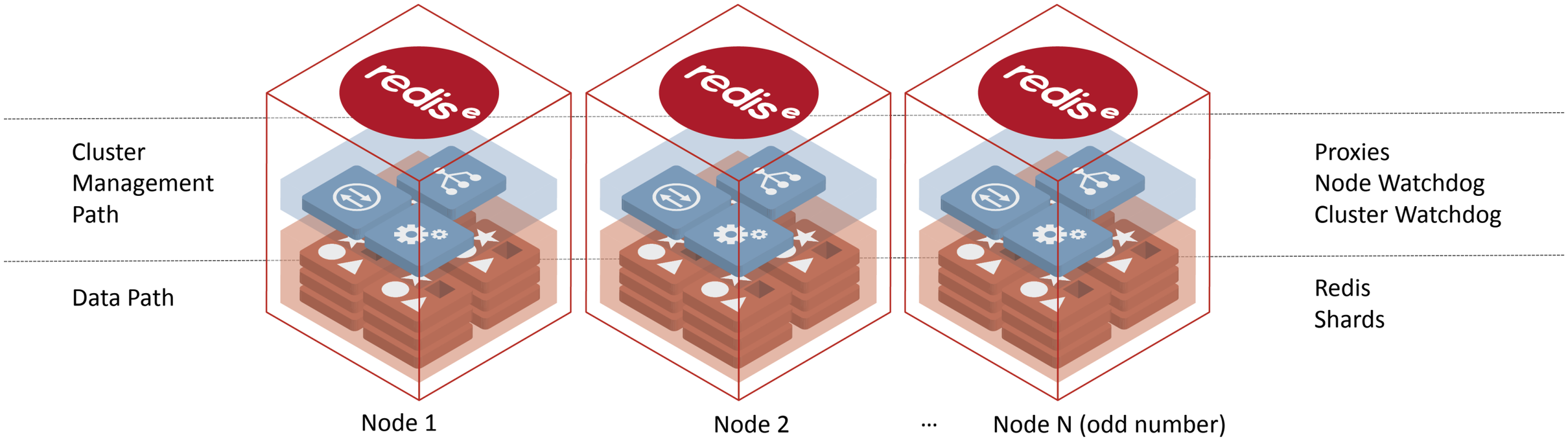
REST API

Redis^{e} Cluster

- Shared nothing cluster architecture
- Fully compatible with open source commands & data structures

Redis^{e} - Shared Nothing Symmetric Architecture

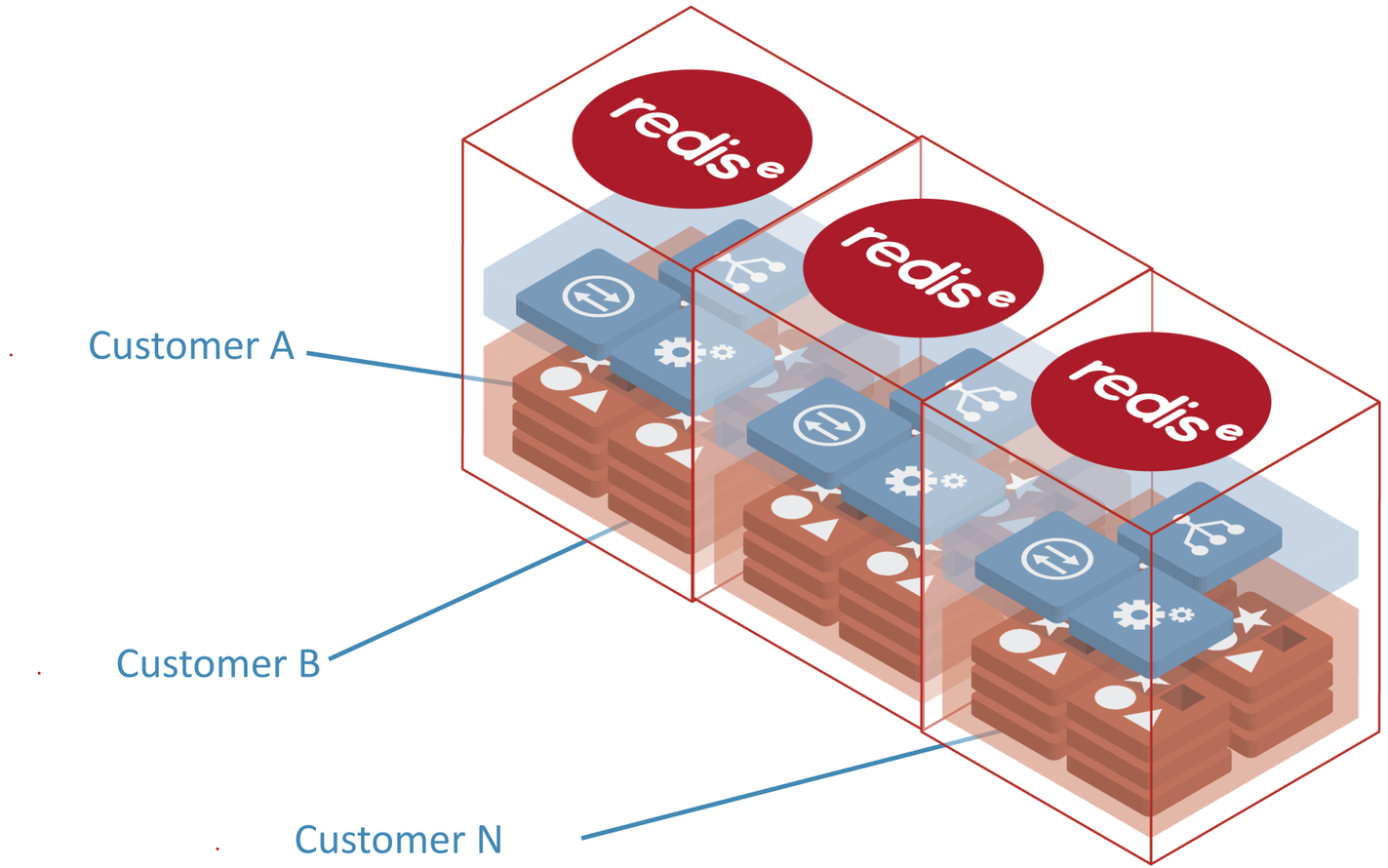
Distributed Proxies
Single or Multiple Endpoints



Unique multi-tenant “Docker” like architecture enables running hundreds of databases over a single, average cloud instance without performance degradation and with maximum security provisions

Redis^{e} - Multi-Tenancy

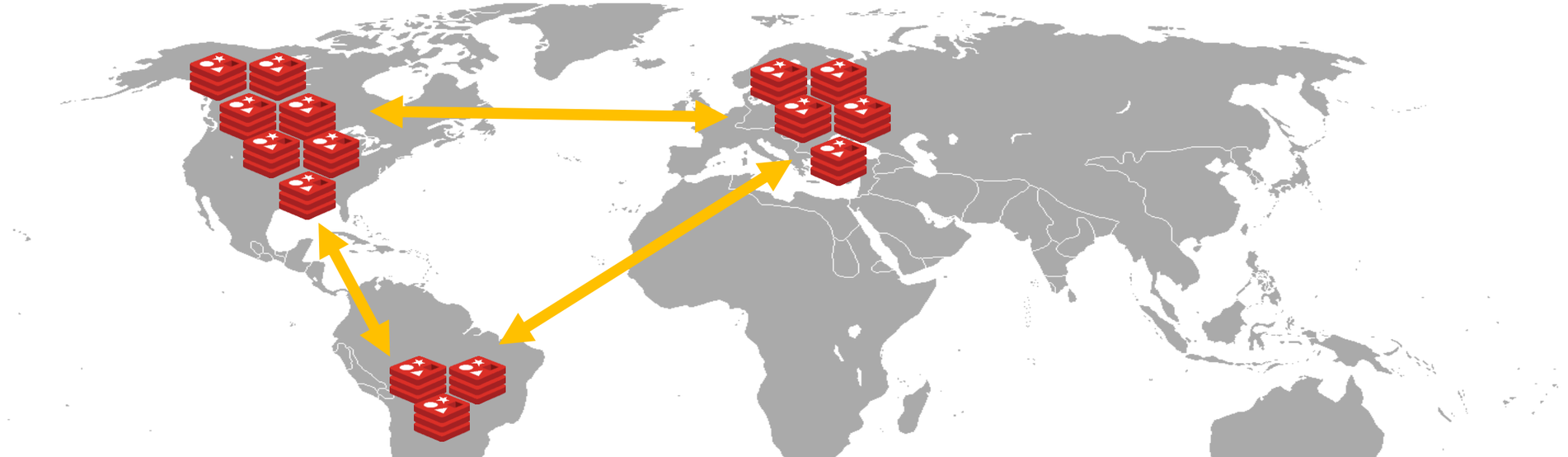
200+ customers/shards on a single 4vcore cloud instance



- Shard isolation/protection
- Noisy-neighbor cancellation
- Minimizing CPU consumption of inactive customers

Redis^{e} : State-of-the Art Database

- Multi-model, can support all popular database models and modern use cases
- ACID compliant with support for multi command/operation transactions
- Geographically distributed, “active-active”, multi-master architecture, with “strong eventual consistency”, based on CRDT (Conflict-free Replication Data Type) technology

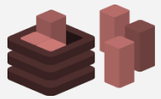


Redis[e] Benefits

Effortless Scaling

Simple, Seamless Clustering. Linear scaling

ACID Compliance in Cluster Architecture



Cluster Support for Redis Modules

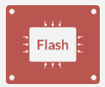
Always On Availability

Instant Failure Recovery, No Data loss

Stable and Predictable High Performance

Active-Active Geo Distribution

Substantially Lower Costs



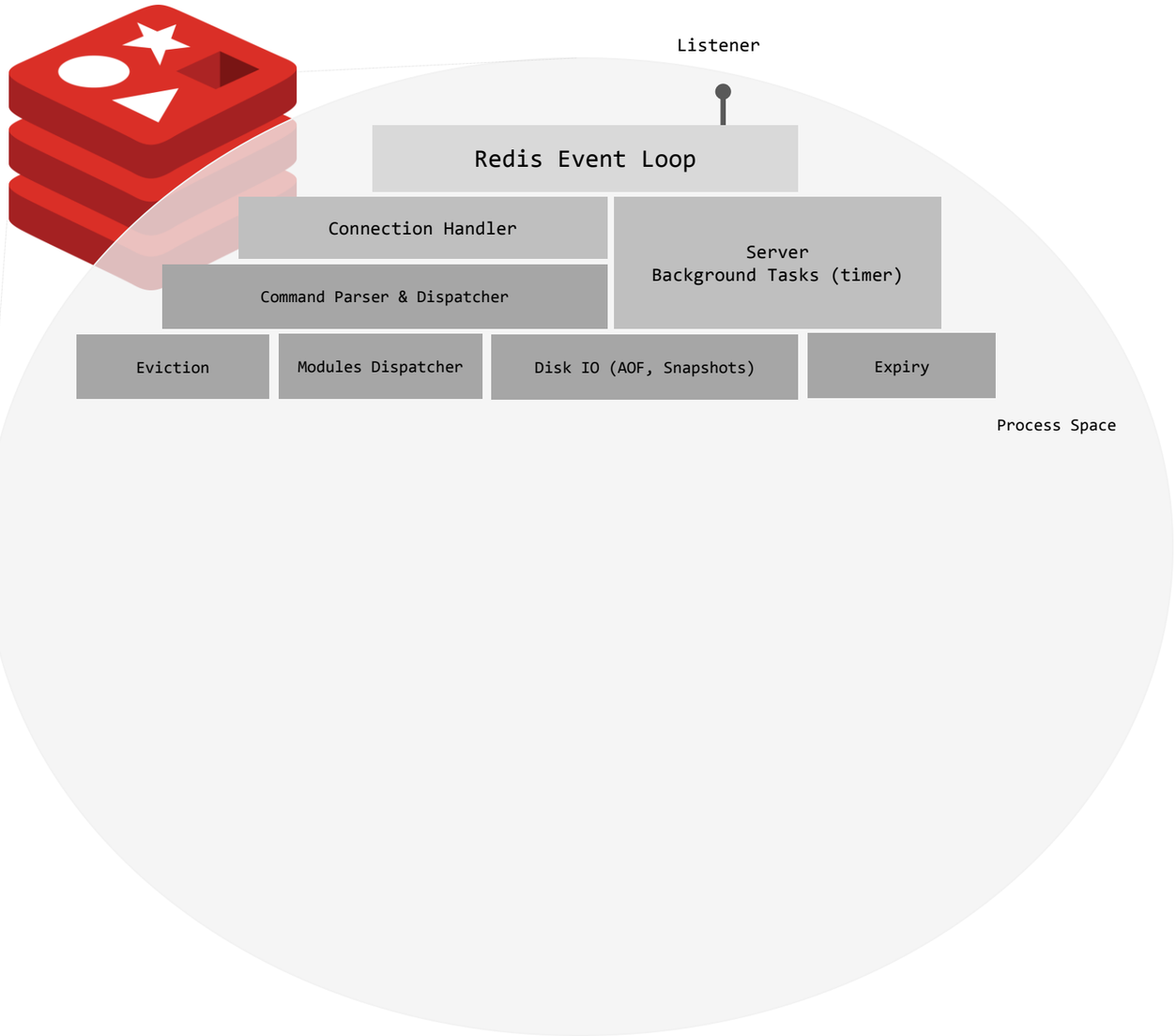
Run on Flash as a RAM extension

Fewer resources, lower overhead

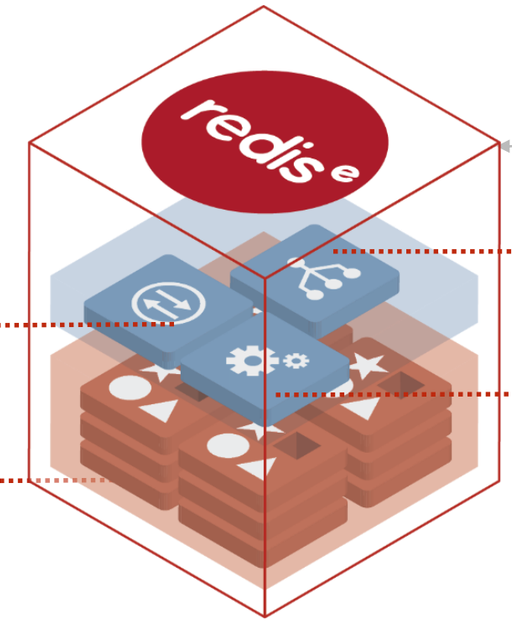
Top notch 24x7 expert support

Redis Architecture

- Single Threaded, In-memory Engine with Persistence
  - “Lock Free” architecture for fast execution



Redis^{e} Technology – Node Architecture



Redis^{e} Node

Cluster Manager

Govern Cluster, Orchestrate Failure Detection, Failover, Stats Collection & more

Secure UI & REST API

Allow programmable and visual administration over HTTPS

Proxy

Scale Connections
Improve Application Performance

Redise Shards

Based on Open Source Redis

Redis^{e} Technology – Cluster Architecture

Redis^{e} Cluster

- Shared nothing cluster architecture
  - Single node type for simple scalability
- Fully compatible with open source commands & data structures
  - Simply change your connection string to Redis^{e}

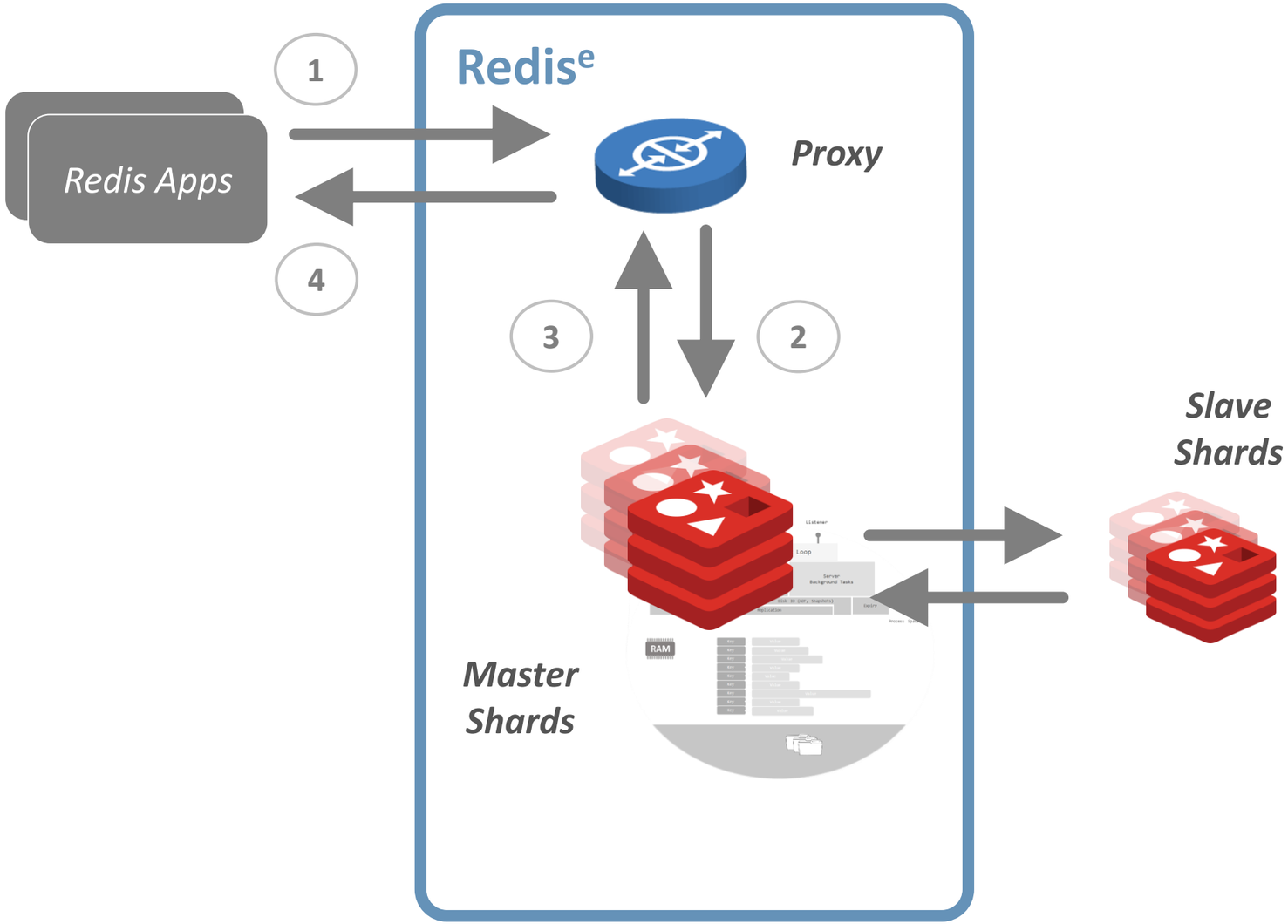
DEMO

Quick Tour of the Redis^{e} Pack Cluster

Anatomy of Read & Write

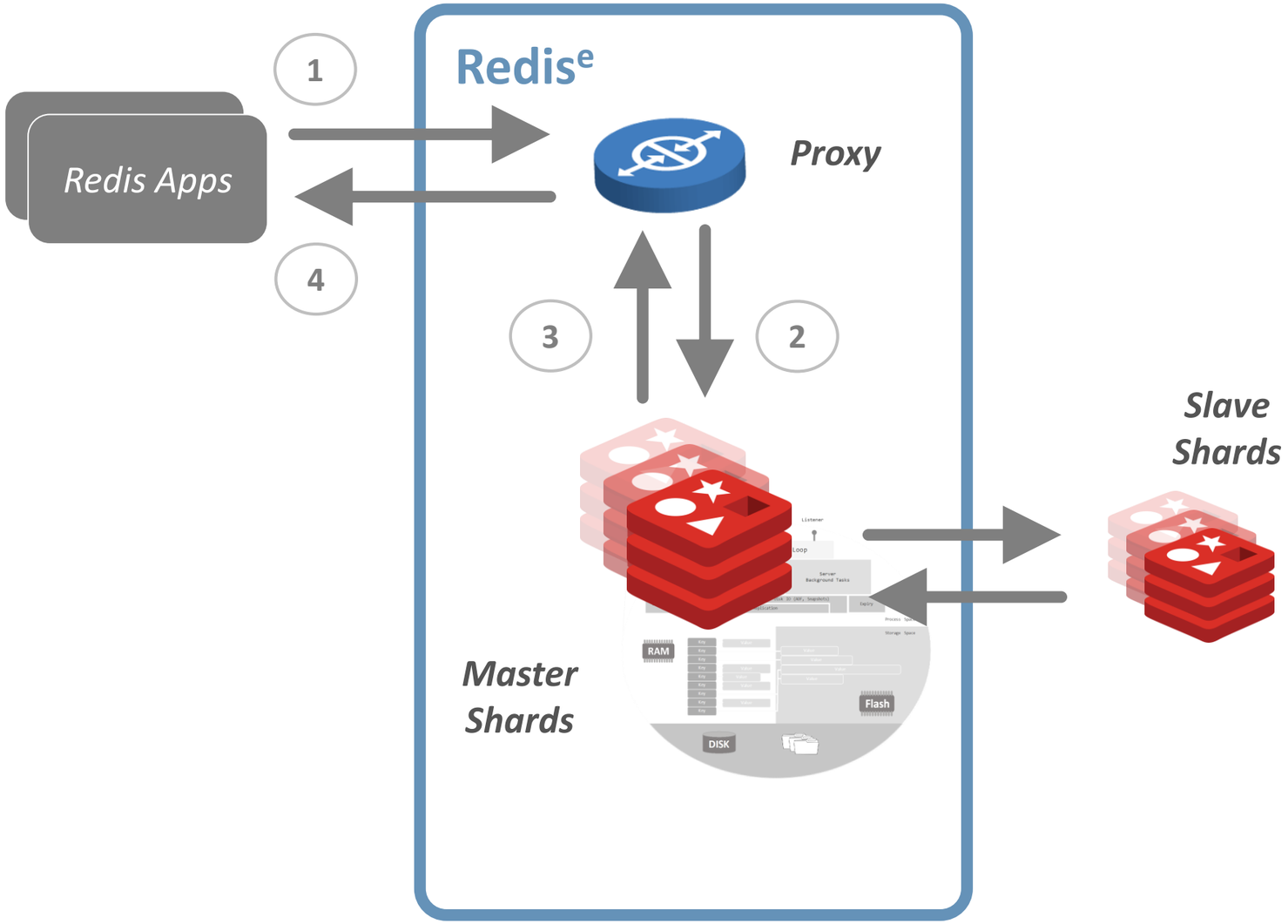
Read/Write Operation with Redis^{e}

1. App submits the operation to one of the Proxies in Redis^{e}
   - *Single Key Ops (GET,STRLEN,HSTRLEN etc)*
   - *Multi Key Ops (MGET, BRPOP, EXISTS, TOUCH, etc)*
2. Proxy distributes the operations to the corresponding shards in parallel
3. All involved shards return data to proxy
   a) Replicate writes to slave shards
4. Proxy assemble responses back to App



Read/Write Operation with Redis^{e} Flash

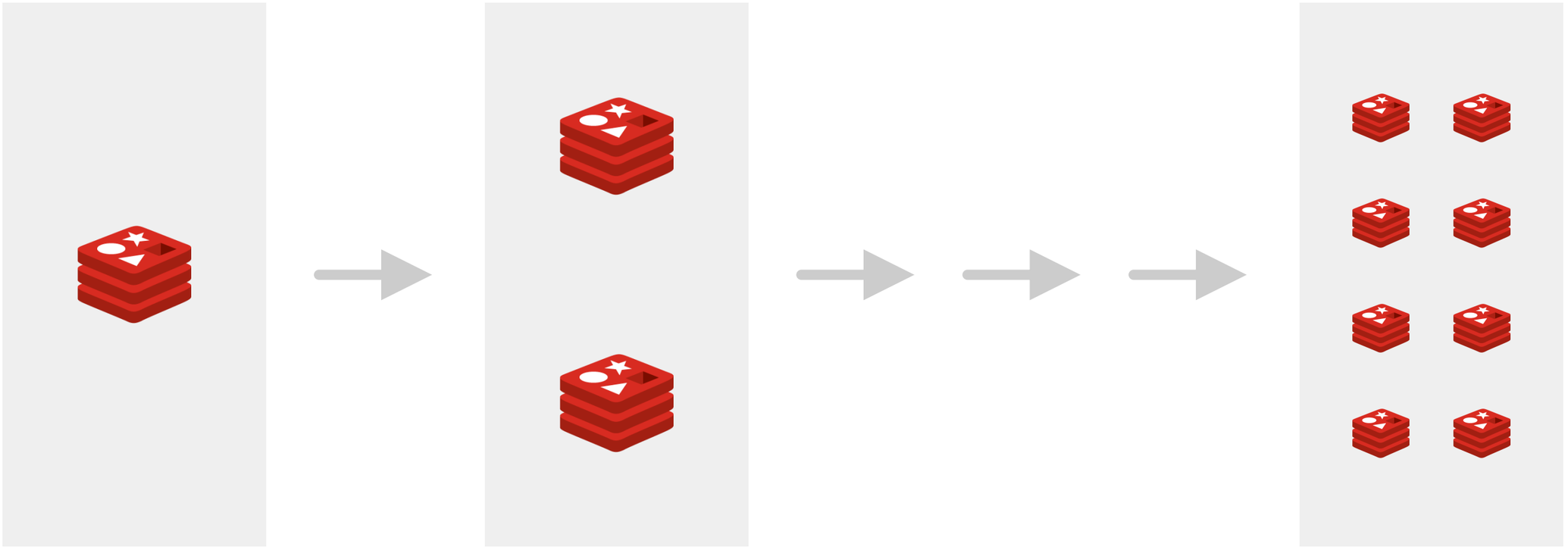
1. App submits the operation to one of the Proxies in Redis^{e}
   - *Single Key Ops (GET,STRLEN,HSTRLEN etc)*
   - *Multi Key Ops (MGET, BRPOP, EXISTS, TOUCH, etc)*
2. Proxy distributes the operations to the corresponding shards in parallel
3. All involved shards return data to proxy
   a) Fetch value from flash if not in RAM
   b) Replicate writes to slave shards
4. Proxy assemble responses back to App



Scaling Applications with Redis[e]

Scaling Throughput

Redis^{e} Shards



1 core

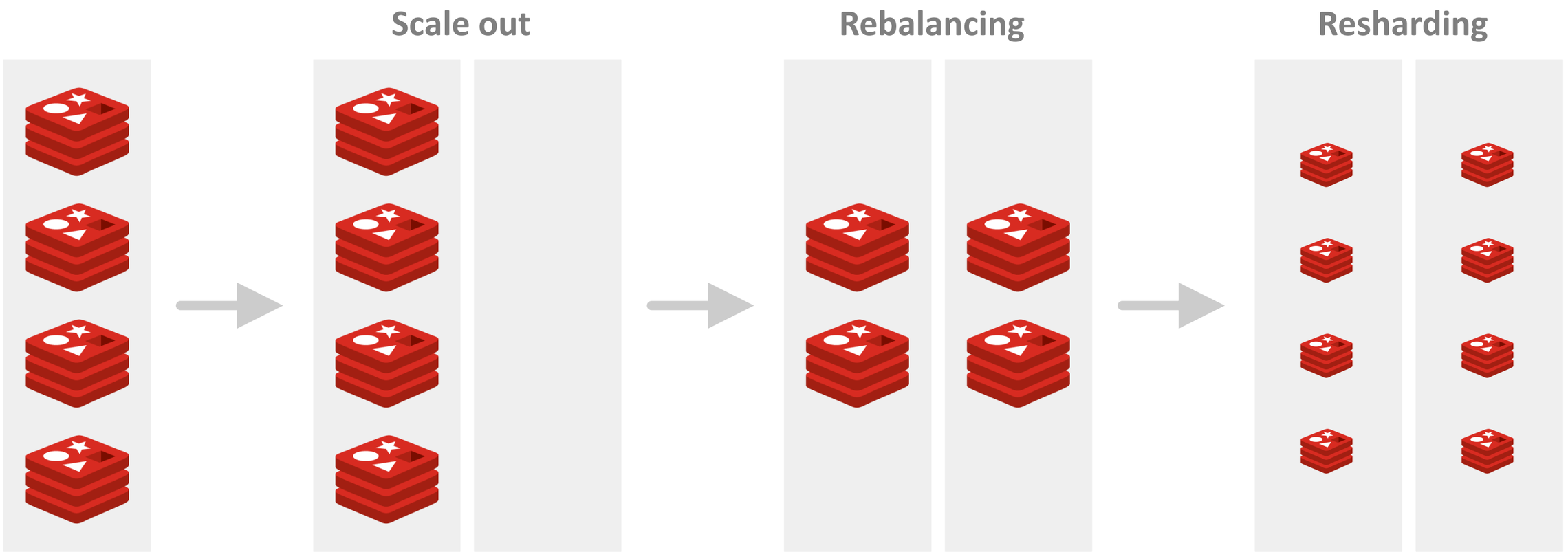
2 cores

N cores

Database is sharded by the Cluster Manager.
Each shard runs on a separate process.

Scaling Up and Scaling Out

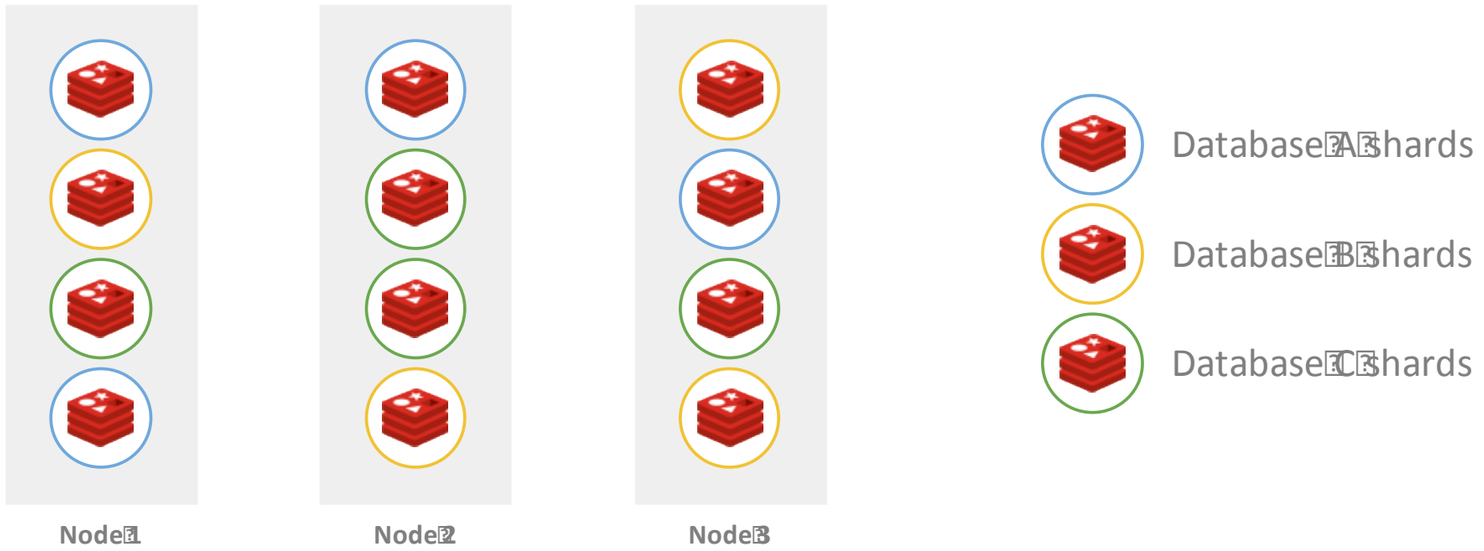
Rebalancing & Resharding without Downtime



Multi Tenancy

Built for Ground up for Multi-tenancy

- Simple Tenant Manageability,
- Tenancy Model for the Best Economics.
  - Mixing Many Small Tenants & Isolated Large Tenants



How is Scaling Managed

Redis^{e} Cloud

- Fully-automated

Redis^{e} Pack

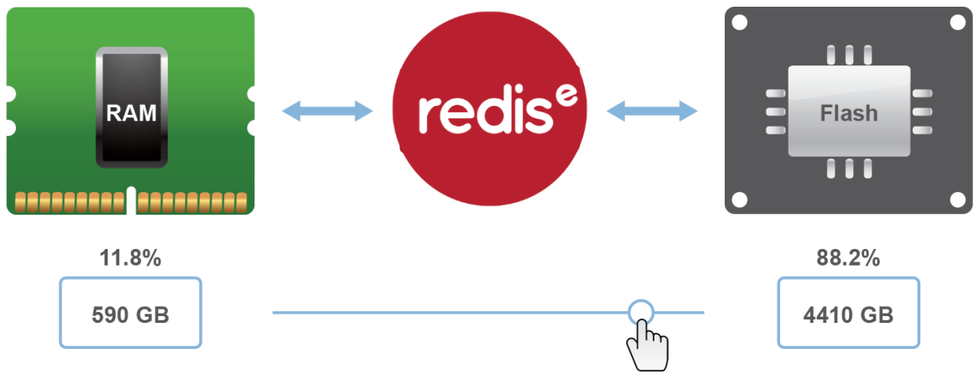
- GUI
- CLI
- API

Scaling Data Size

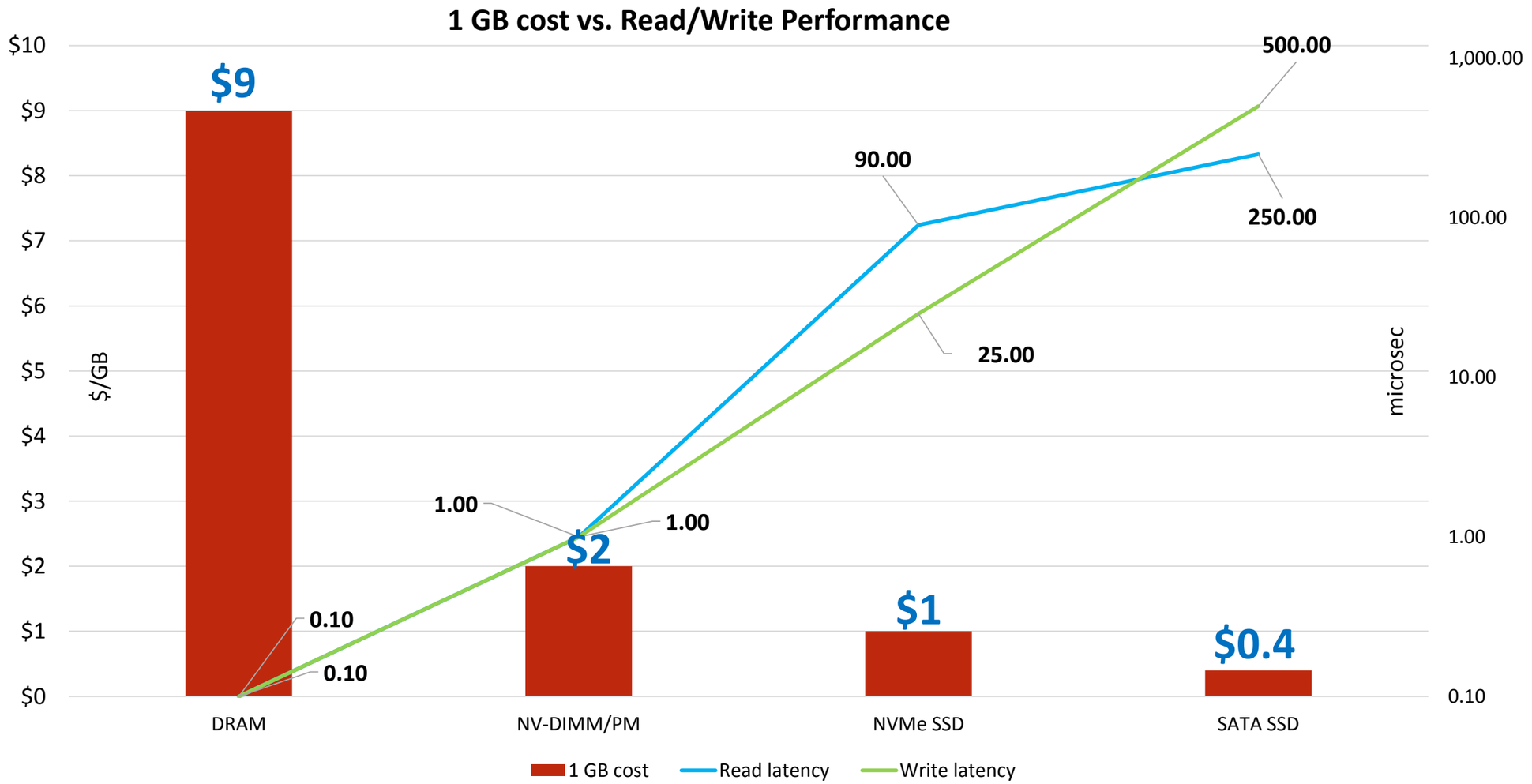
Redis^{e} Flash

Massive Datasets with Near-Ram Latency at a Drastically Lower Cost

- Optimized Read/Writes with RAM-Extension approach
- Gain speed with smart caching between RAM and Flash
- Future proof for upcoming *persisted-memory* technology

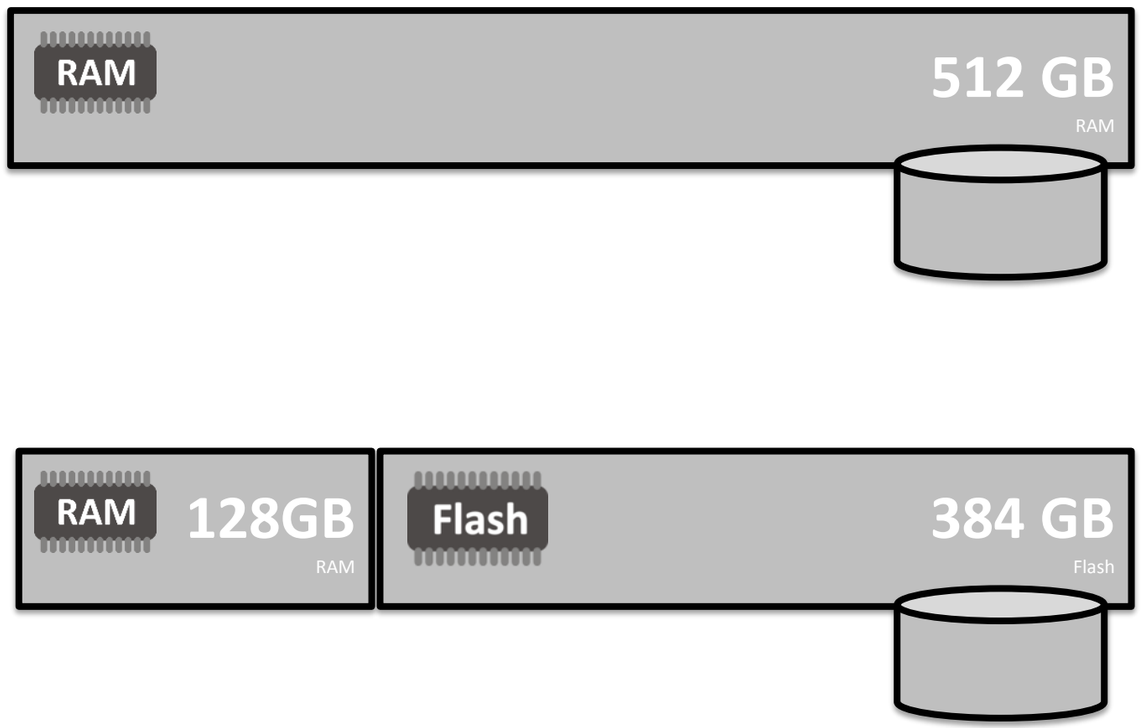


Price/Performance – Memory Technology



Why Redis^{e} Flash?

Lower cost for Larger data sets



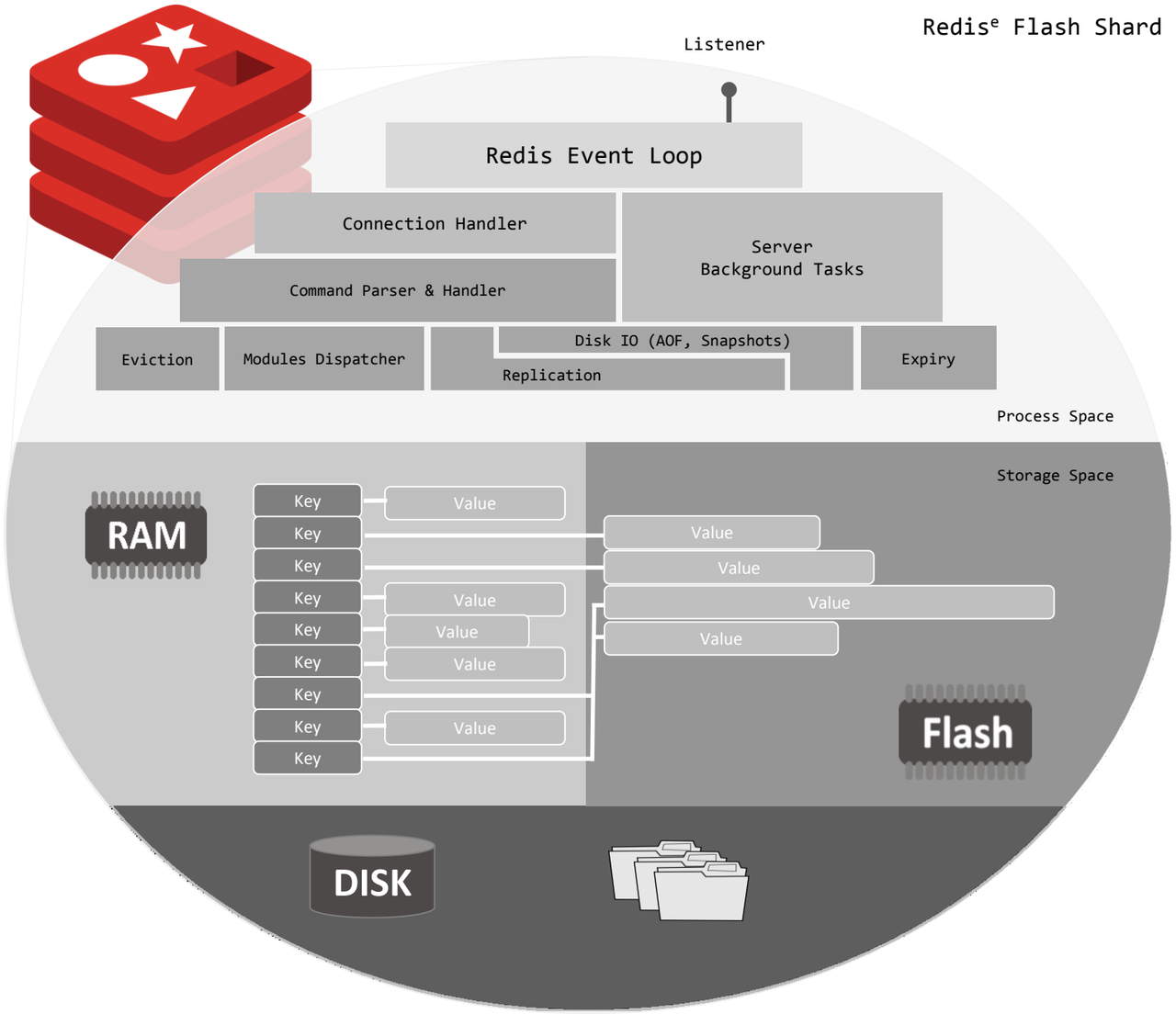
>80%

Lower Cost with RAM+Flash

Compared to all-in-RAM

Redis Shards in Redis^{e}

- Redis^{e} Flash Shard
  - Ability to extend RAM to Flash for cheaper storage of data (Redis^{e} Flash)
  - Advanced threading for better parallelism
  -



Redis^{e} Flash vs Disk Based Databases?

	Redis on Flash	Disk Based Databases
Hot Value Handling	**No IO Required** Keep hot values in RAM	**Heavy IO Required** Keeps writing to disk
Write Performance	**Faster Writes** Non-Durable Writes with Ram Extension approach	**Slower Writes** Durable Writes (WAL, Redo logs etc)
Cloud Optimized	**Fast Local Writes to Ephemeral Drive** Utilizes the Ephemeral Drive for fast local IO and Network IO for durability	**Slow Writes to Network Attached Storage** *CANNOT* Utilizes the Ephemeral Drive for fast local IO
Future Proof	**Ready for Persistent Memory** Systems like Intel 3D-XPoint	**Needs Re-Architecting**

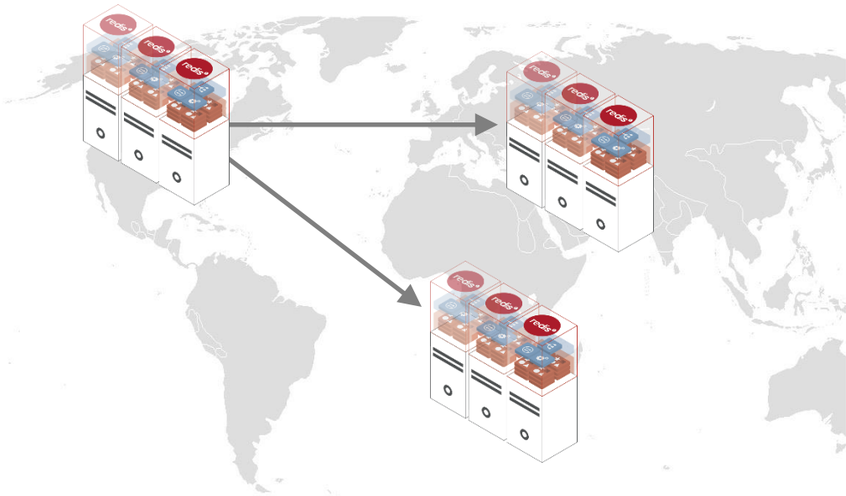
High Availability with Redis[e]

Redis^{e} Replication Architecture

Replication with Redis^{e}

Low Replication Lag & High Replication Throughput

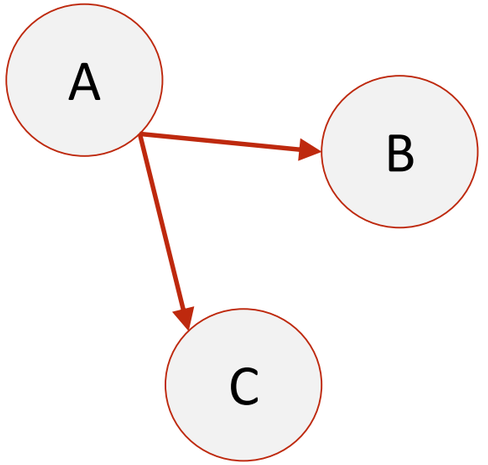
- Local Replication: Built for LAN
  - Higher bandwidth
  - Lower latency
  - High quality links susceptible to fewer failures and retransmits



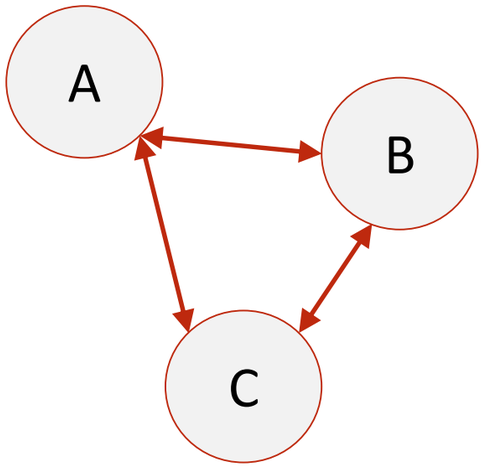
- Cross-Geo Replication: Built for WAN
  - Lower bandwidth
  - Higher latency
  - “Noisier” network quality susceptible to more failures and retransmits

Cross-Geo Replication

- Unidirectional Replication
  - ***Replica Of*** - Source DB to Destination DB across WAN
    - Content Distribution to Various Geographies for low latency, local reads
    - Geo-distribution for Hot Standby protecting against regional failures

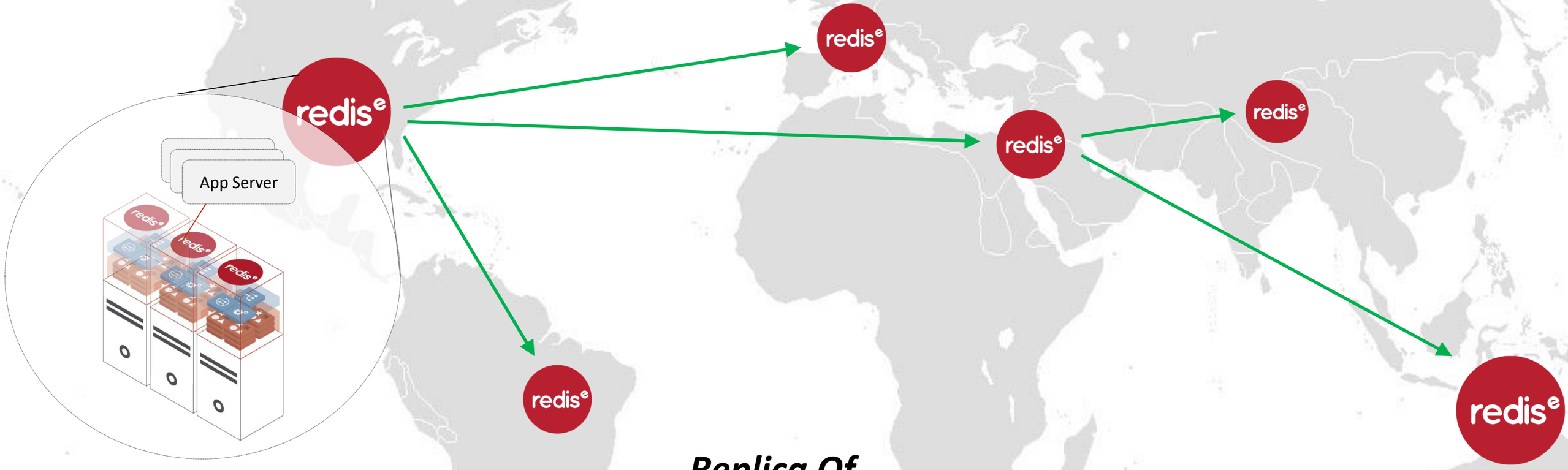


- Multi-Master Replication *(coming soon in v5.0)*
  - ***Multi-Master Databases*** with Concurrent Active-Active Writes using CRDTs
    - Database spanning multiple geographies and clusters
    - Smart handling of conflicting concurrent writes with strong eventual consistency



Common Redis^{e} Geo Distribution Topologies

Topology #1
Geo Distribution for Fast Local Data Access



Replica Of

Geo Distribution for Local Data Access (CDN Like)

Great for…

Distribute Content Close to Your Users

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

Avoid Write Conflicts - Achieve Higher Consistency

- All writes go to the master cluster, source database

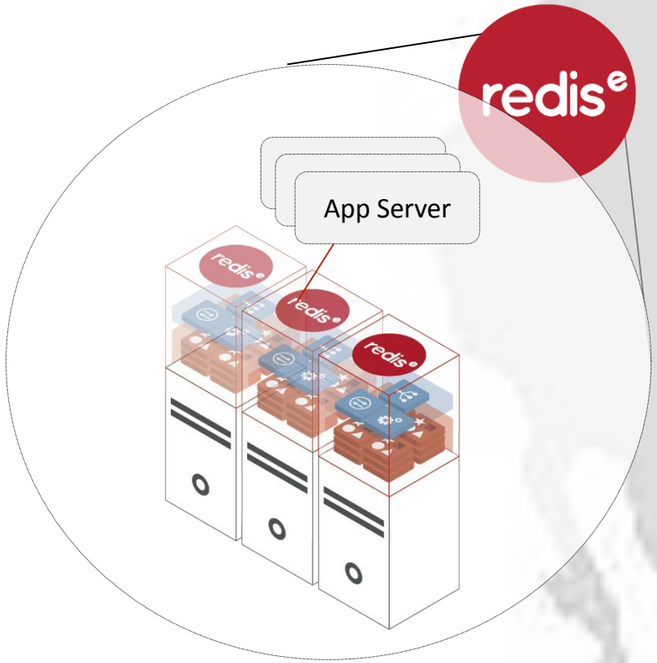
Increase Read-Availability through Multiple Read-Replicas

- Multiple data sources to read from maintained up to date with fast replication

Independently Size Each Region's Cluster

- Independently build enough capacity to handle local workloads in each region

Topology #2
Geo Distribution for Warm Stand-by



DB1 : [*West Coast Users*]

DB2 : [*East Coast Users*]

Replica Of

Geo Distribution for Hot Stand-By

Great for…

Protect against Regional Failures

- Place a hot standby server to a different geo
- Increase Read and Write-Availability through Active-Passive Distribution

Avoid Write Conflicts - Achieve Higher Consistency

- All writes go to the master cluster with the source database

Independently Size Each Region's Cluster

- Independently build enough capacity to handle local workloads in each region

Recap

Redis and Redis^{e} (Redis Enterprise) Architecture

- Shards, Nodes, Clusters
- Replication in Redis^{e}
- Anatomy of Read/Write Operation

Scaling Applications with Redis^{e}

- Scaling Throughput – Redis^{e} Resharding and Rebalancing
- Scaling Connections - Redis^{e} Proxy
- Scaling Data Size – Redis^{e} Flash

High Availability with Redise

- Replication Architecture in Redis^{e}
  - Local Replication across the LAN
  - Cross-Geo Replication across the WAN

DEMO

Performance and HA with Redis^{e} Pack Cluster