Welcome! Quick checklist while you settle in

1. Access the Github repo with the Workshop labs and deck

   https://tinyurl.com/workshop-labs

2. Start Lab 1: Get yourself a lab environment

Real Time Search Workshop

SA Specialist Workshop Series

<NAME>
<ROLE>

Agenda

09:30 a.m.	Check-in, Introductions & Opening Remarks
10:00 a.m.	Redis Overview and Use Cases
	RedisMart Demonstration
	Get your Redis Environment (LAB 1)
11:00 a.m.	Real Time Search and JSON
	Basic JSON Operations (LAB 2) & Basic Search Operations (LAB 3)
12:00 p.m.	Lunch Break (continue through labs/networking)
01:00 p.m.	Real Time Search Technical Deep Dive
	Advanced JSON (LAB 4) & Advanced Search (LAB 5)
02:00 p.m.	Real Time Search in Production
	Deploying a 99.999 Redis Environment (LAB 6)
03:00 p.m.	Program Concludes

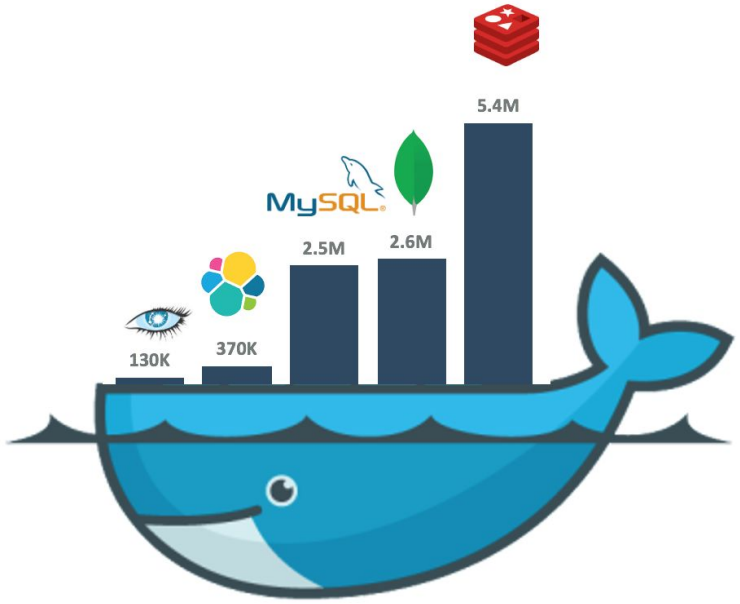
Introduction to Redis

Background on **Redis**

Company	Redis is home of Redis the world's most popular **In-memory open-source database**, and commercial provider of Redis Enterprise
Background	Founded in 2011, private company, raised **$555M** from **12** investors including Bain Capital Ventures, Francisco Partners, Goldman Sachs Growth, Viola Ventures, Dell Technologies Capital, TCV, Softbank, and Tiger Global.
Technology	An **In-memory open-source NoSQL database**, supporting a variety high performance operational, analytics or hybrid use case.
Partnerships	Google Cloud, Amazon Web Services, Microsoft Azure — Strategic partnerships with leading **cloud partners** with integration and support of Redis Enterprise

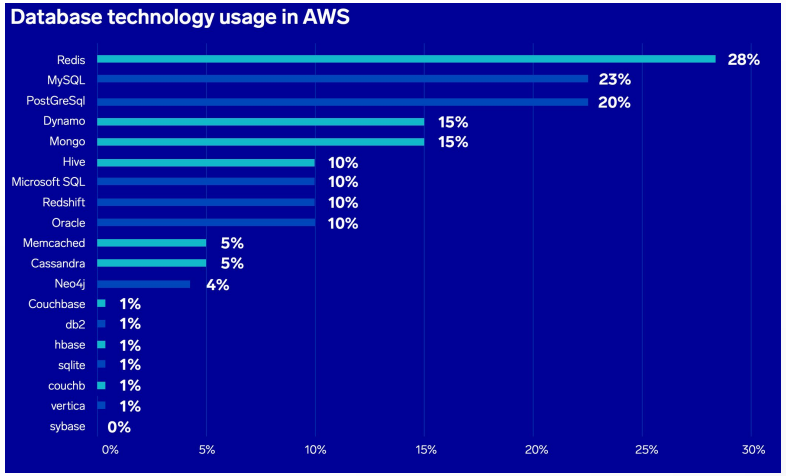
REDIS IS LOVED BY DEVELOPERS

MOST LAUNCHED

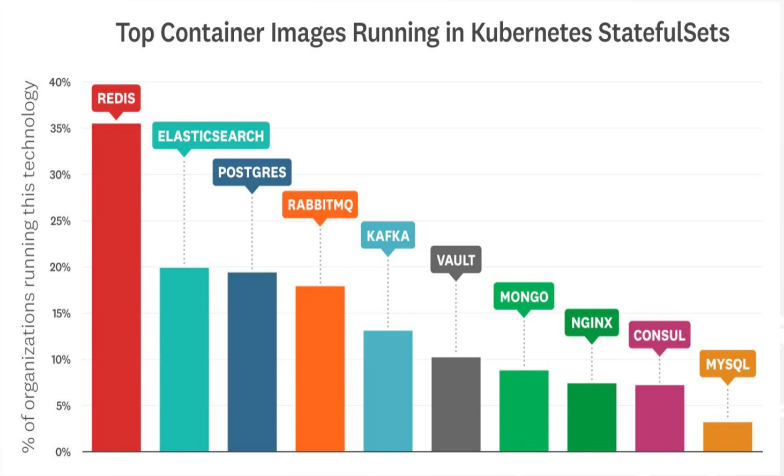


Launches/day: Docker Hub

MOST USED

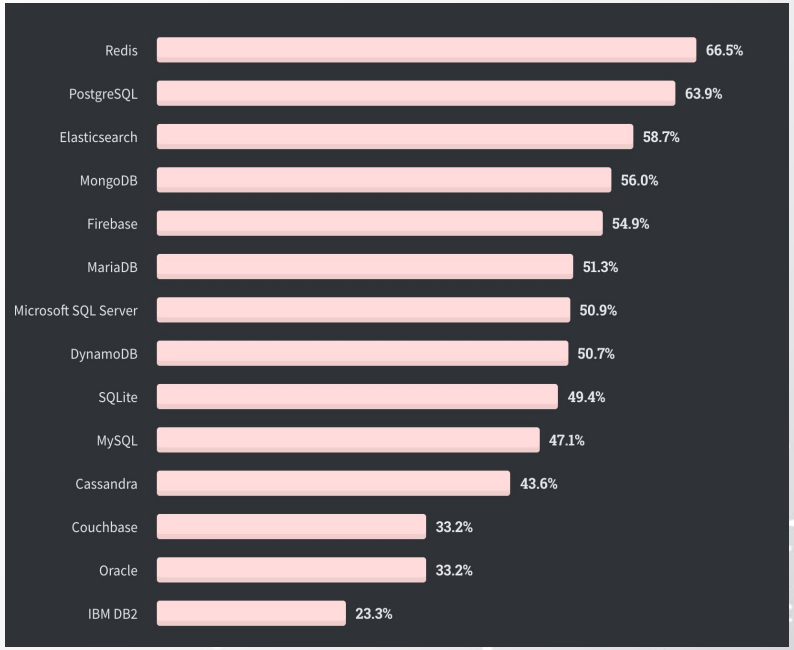


Sumo Logic, Nov. 2020



Datadog, Nov. 2020

MOST LOVED



Stack Overflow, 2017, 2018, 2019, 2020, 2021


redis


telesign


ifood

zapier


viacom 18


ULTA
BEAUTY


Remind


gojek


ON24


RAYMOND JAMES


YEXT


TELUS


DUNZO


MYTEAM11


plivo

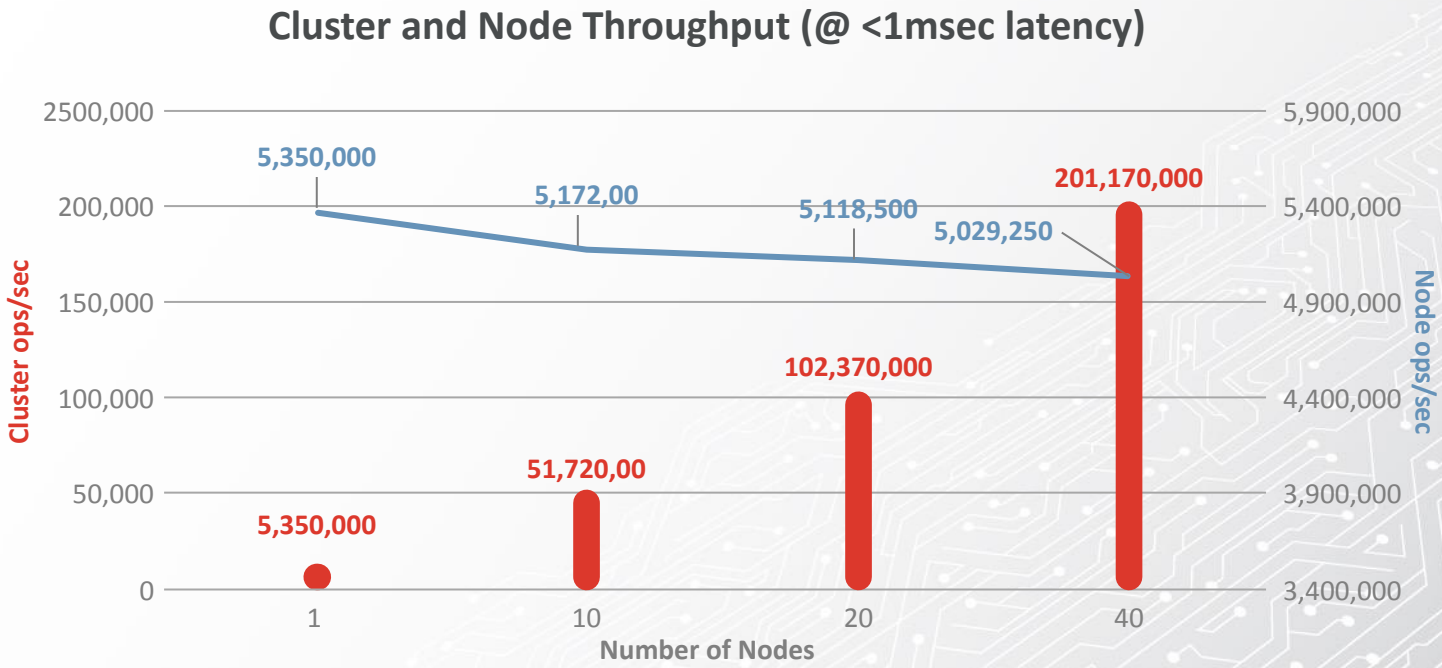

HealthStream

Redis Enterprise brings the Most Loved Database to Petabyte Scale

Shared-nothing architecture to enable **near linear scaling**

Fully utilizes **multi-core** / **multi-node** architecture for vertical and horizontal scaling

Scaling and **resharding** without downtime



Redis Enterprise with **200M ops/sec** @ **<1ms** latency on only 40 AWS instances

Real-time data platform

Enterprise Grade Capabilities

- Linear Scalability
- High Availability
- Durability
- Backup & Restore
- Geo-Distribution
- Tiered-memory Access
- Multi-tenant
- Security

Data Processing Engines

- Event Streaming & Data Integration
- Query & Search
- Server-side functions, write-through, write-behind

Developer Experience

RedisInsight

Fully Supported Client Libraries

Object Mapping Libraries

- Jedis
- node-redis
- redis-py

Data Structures

- Strings
- BitFields
- Sorted Sets
- Geospatial
- Lists
- Bitmaps
- Hashes
- Sets
- HyperLog
- Streams
- JSON
- Probabilistic

Open Source Core

Uniquely Suited to Modern Use Cases


User Session Store


Content Caching


Real Time Data Ingest


High Speed Transactions


Job & Queue Management


Auto-complete


Time Series Data


Complex Statistical Analysis


Notifications

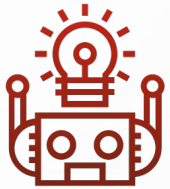

Distributed Lock


Publish/ Subscribe


Fraud Mitigation


Geospatial Data


Streaming Data

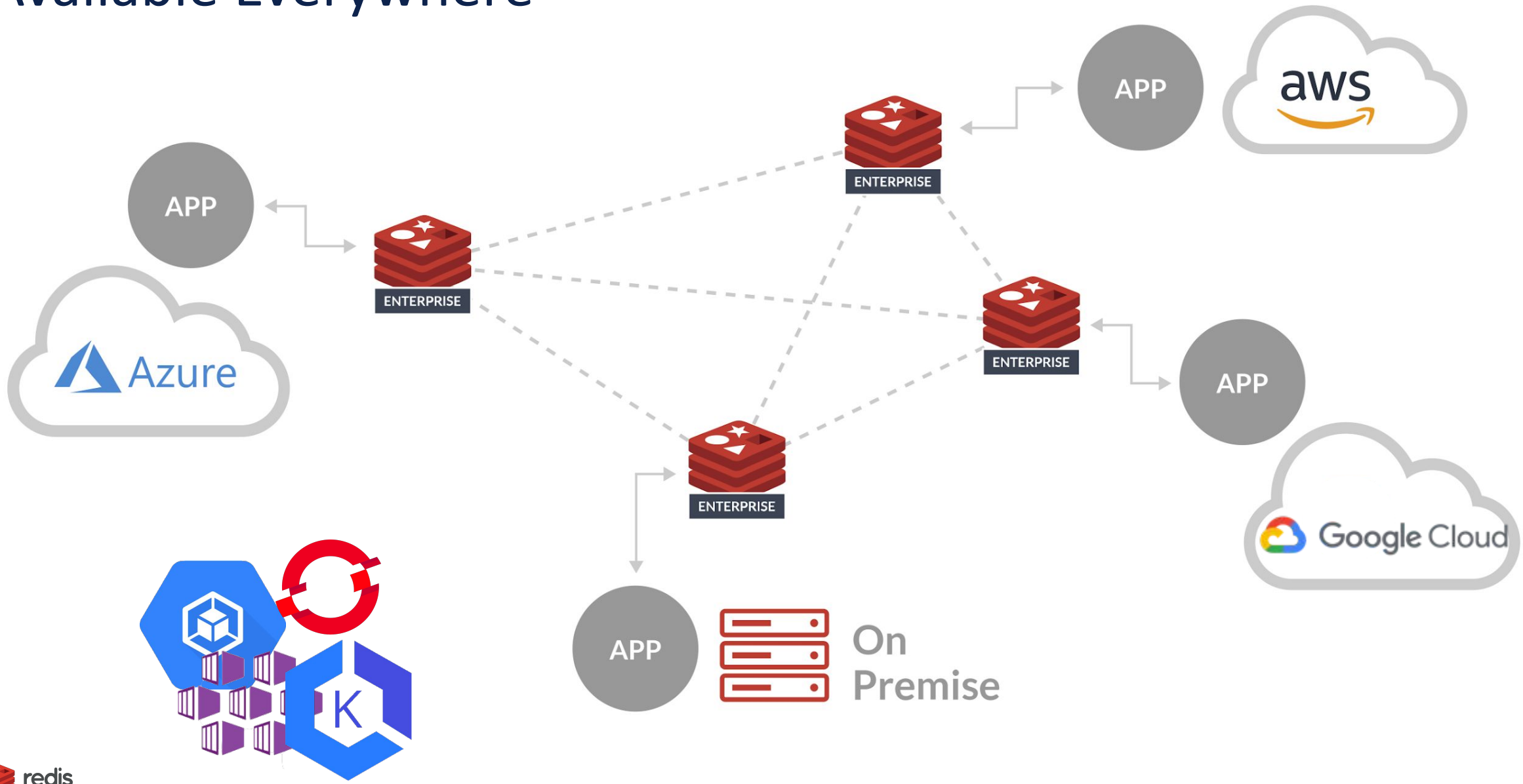

Machine Learning


Search


Geo- Distributed Cache


Leaderboards

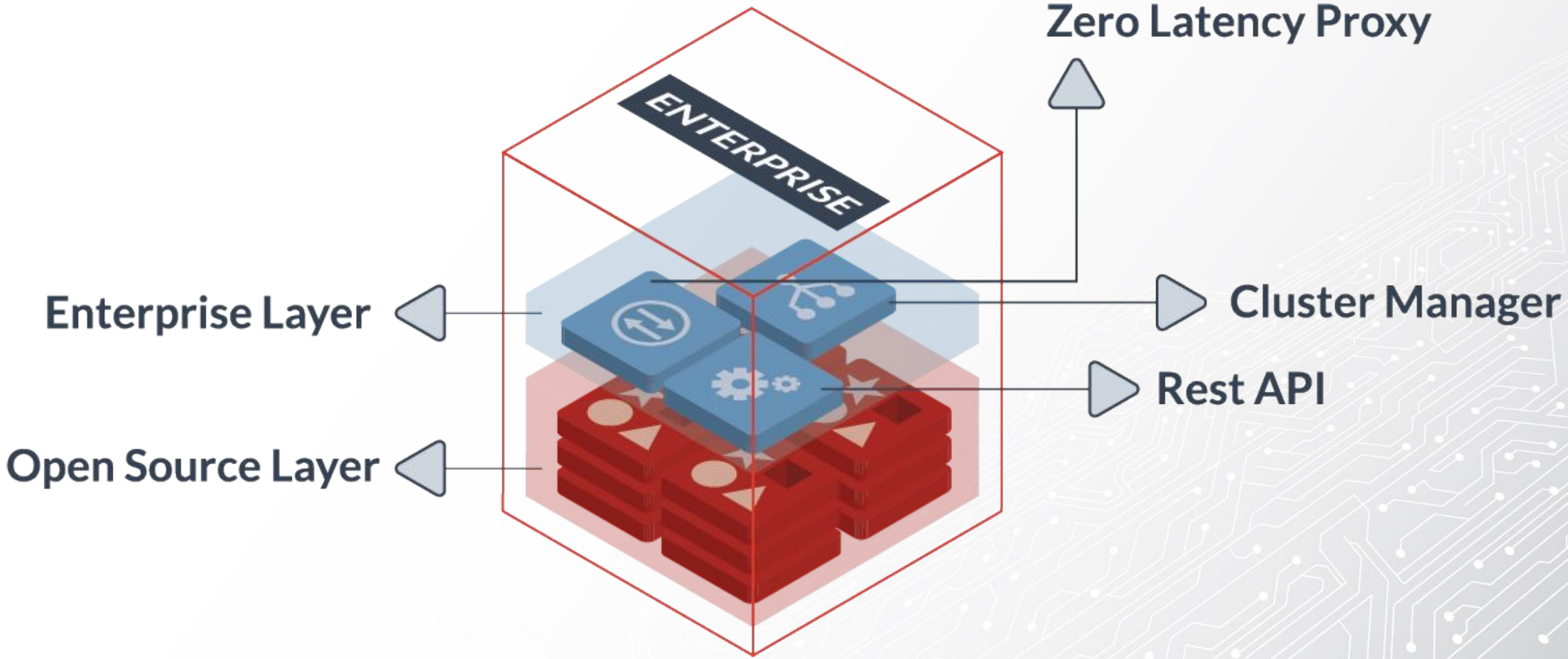
Available Everywhere



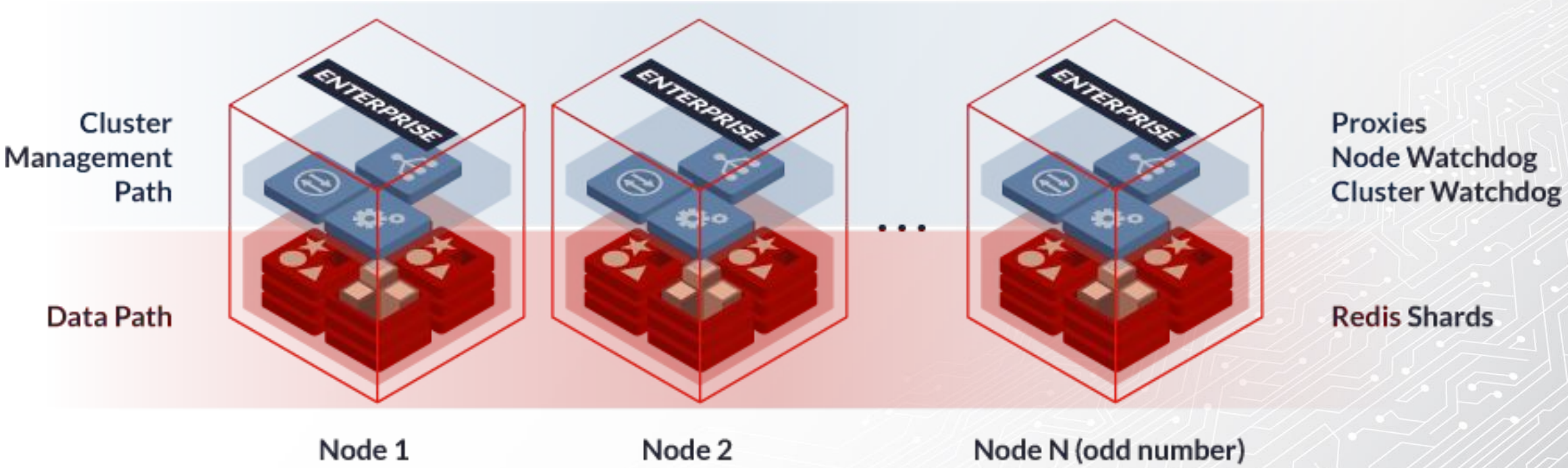
Redis Enterprise Technical Overview

Redis Shard

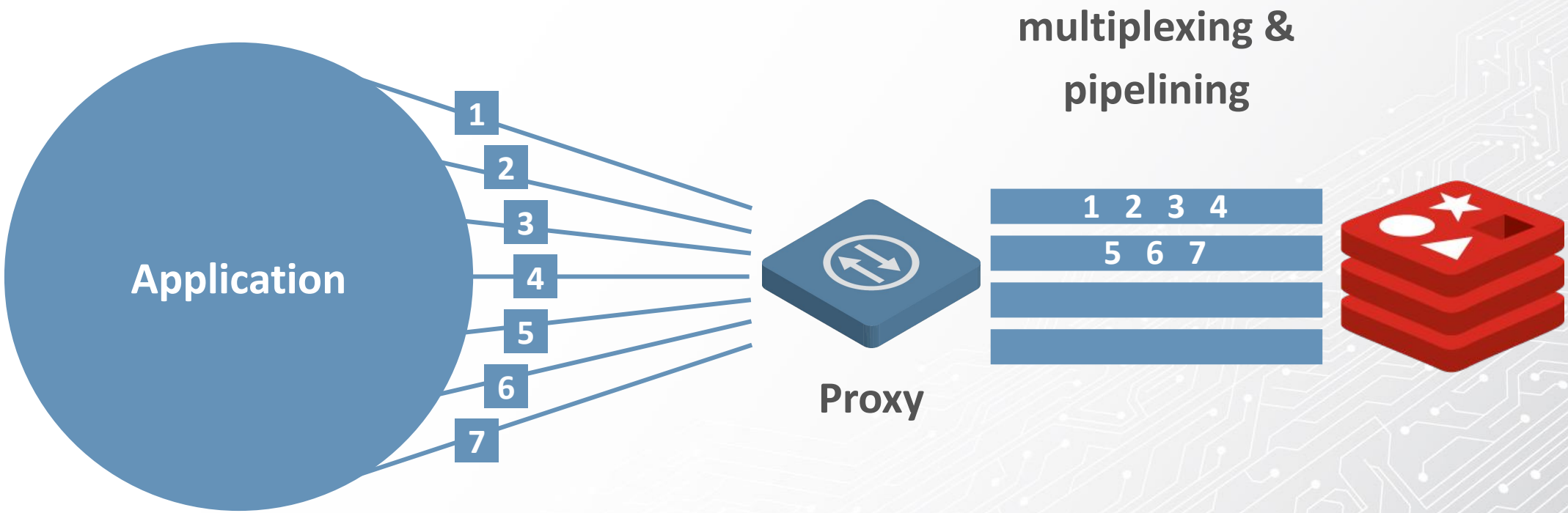
Redis Enterprise Node



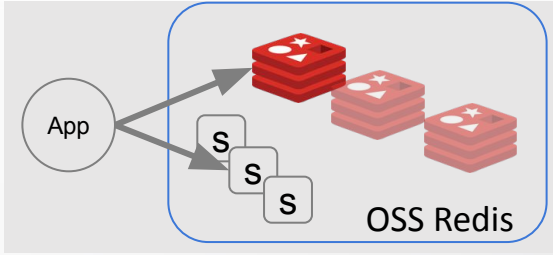
Redis Enterprise Cluster

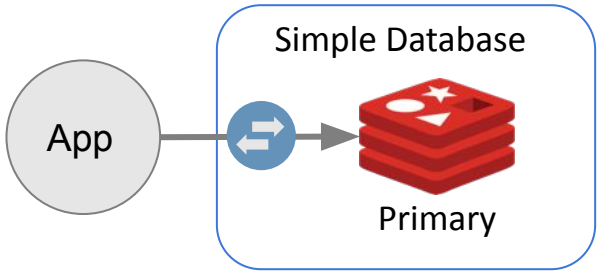


Redis Enterprise Proxy

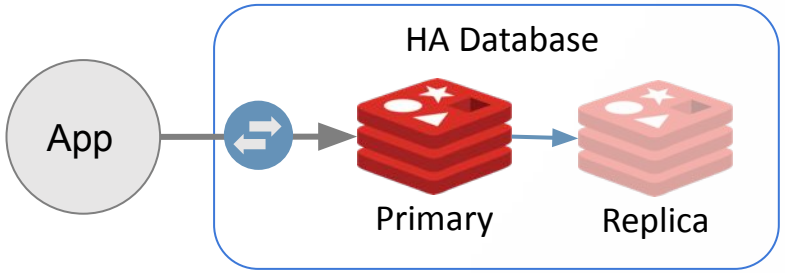


Redis Enterprise Topologies

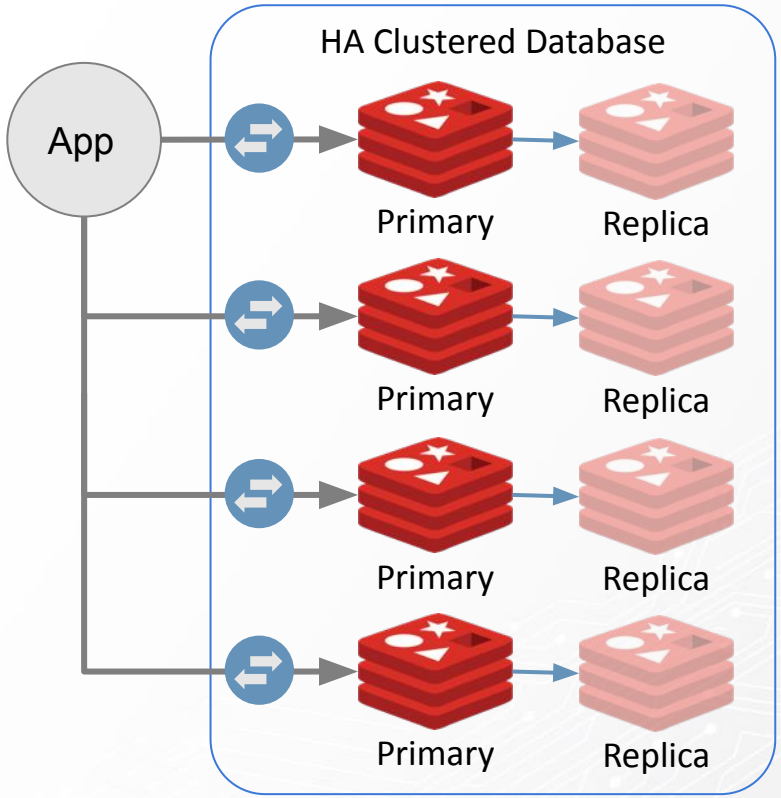




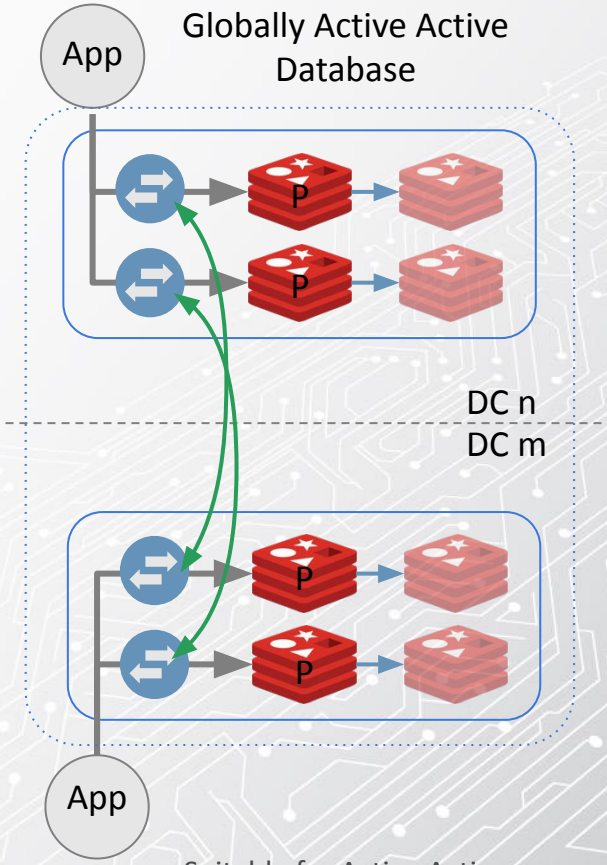
Suitable for :
Dev/Test/MVP



Suitable for Production:
Most internal apps (< 100k users)
(< 20GB, < 20k Ops/sec @ 1ms)



Suitable for Production:
External Apps w > 100k users
(> 25GB, > 25k Ops/sec @ 1ms)

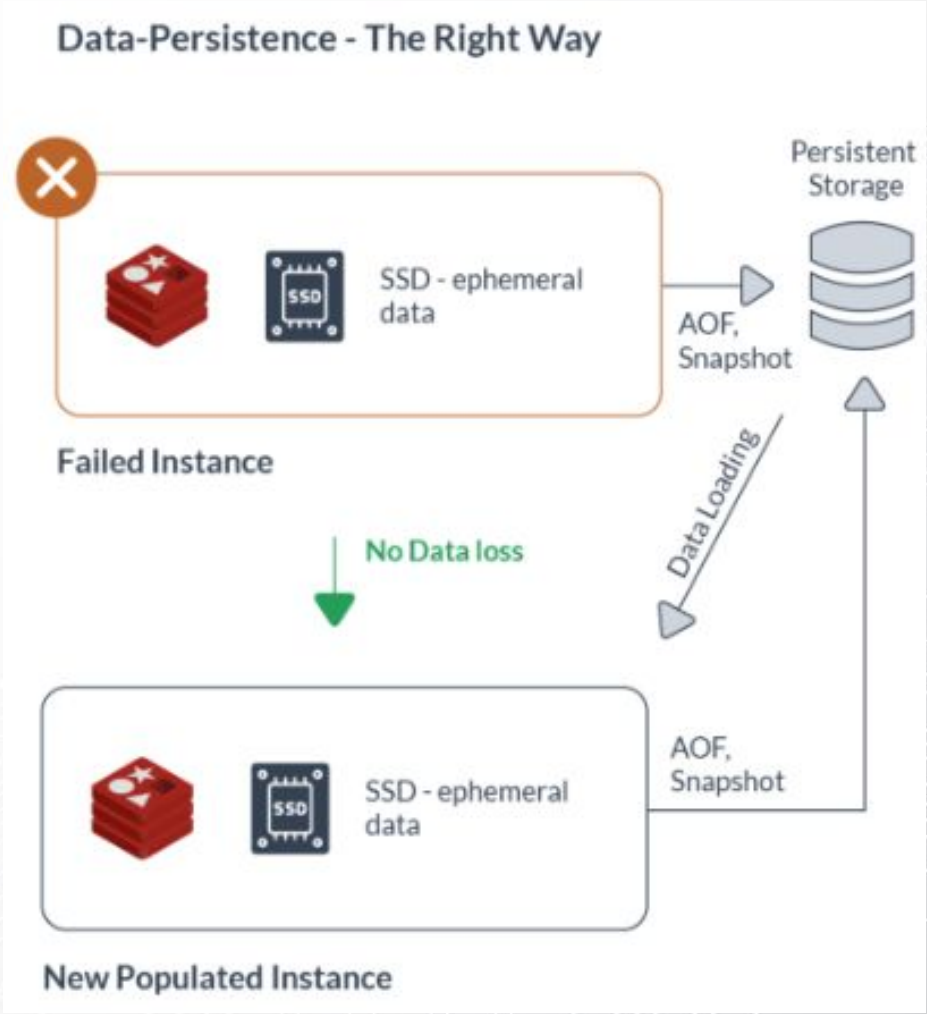


Suitable for Active Active
Production Applications

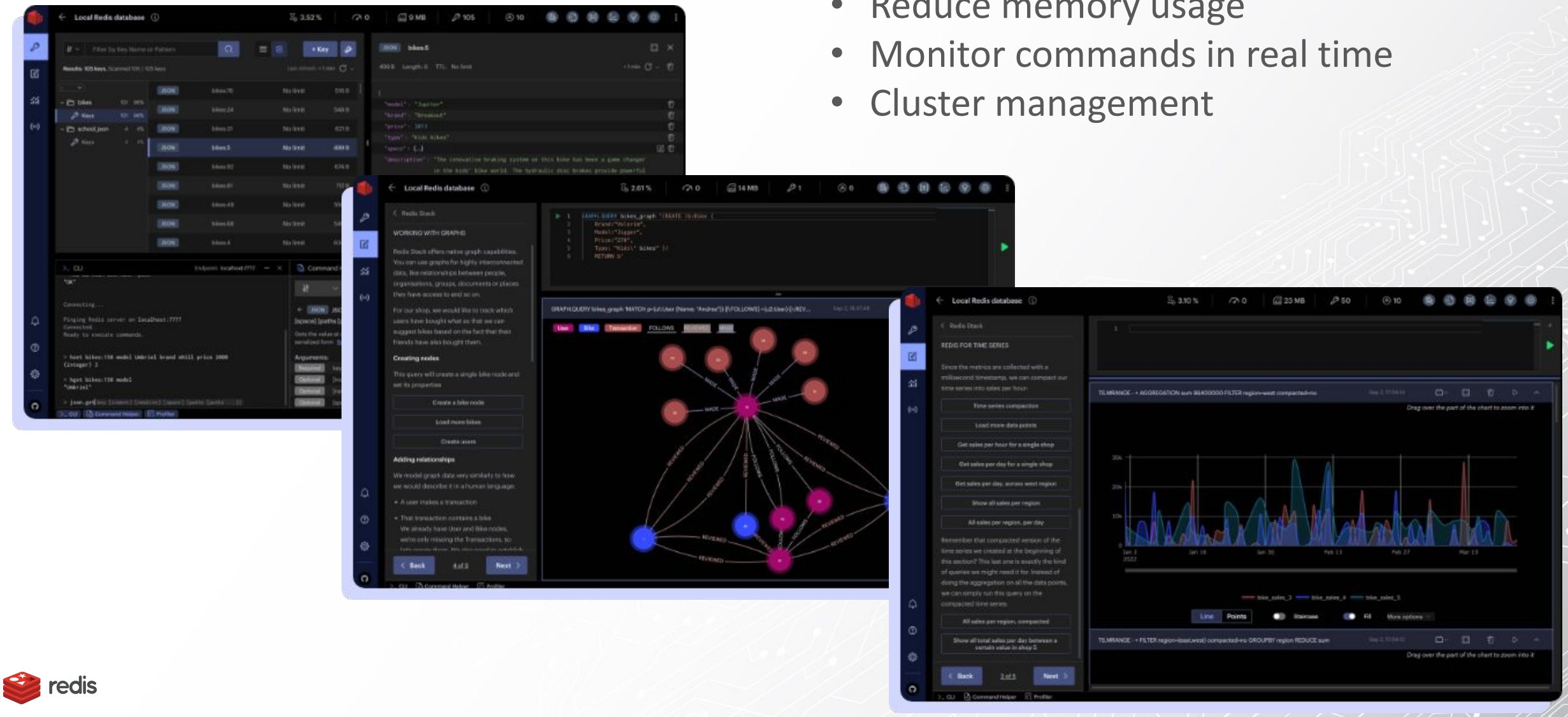
Redis Enterprise Persistence and Backup

- Redis Enterprise provides full durability through two persistence mechanisms
  - Append-Only file (AOF) per write or per second
  - Snapshots (point-in-time copy of the entire database with configurable time intervals)

- Redis Enterprise cluster is designed to work with network-attached storage for data persistence.



RedisInsight

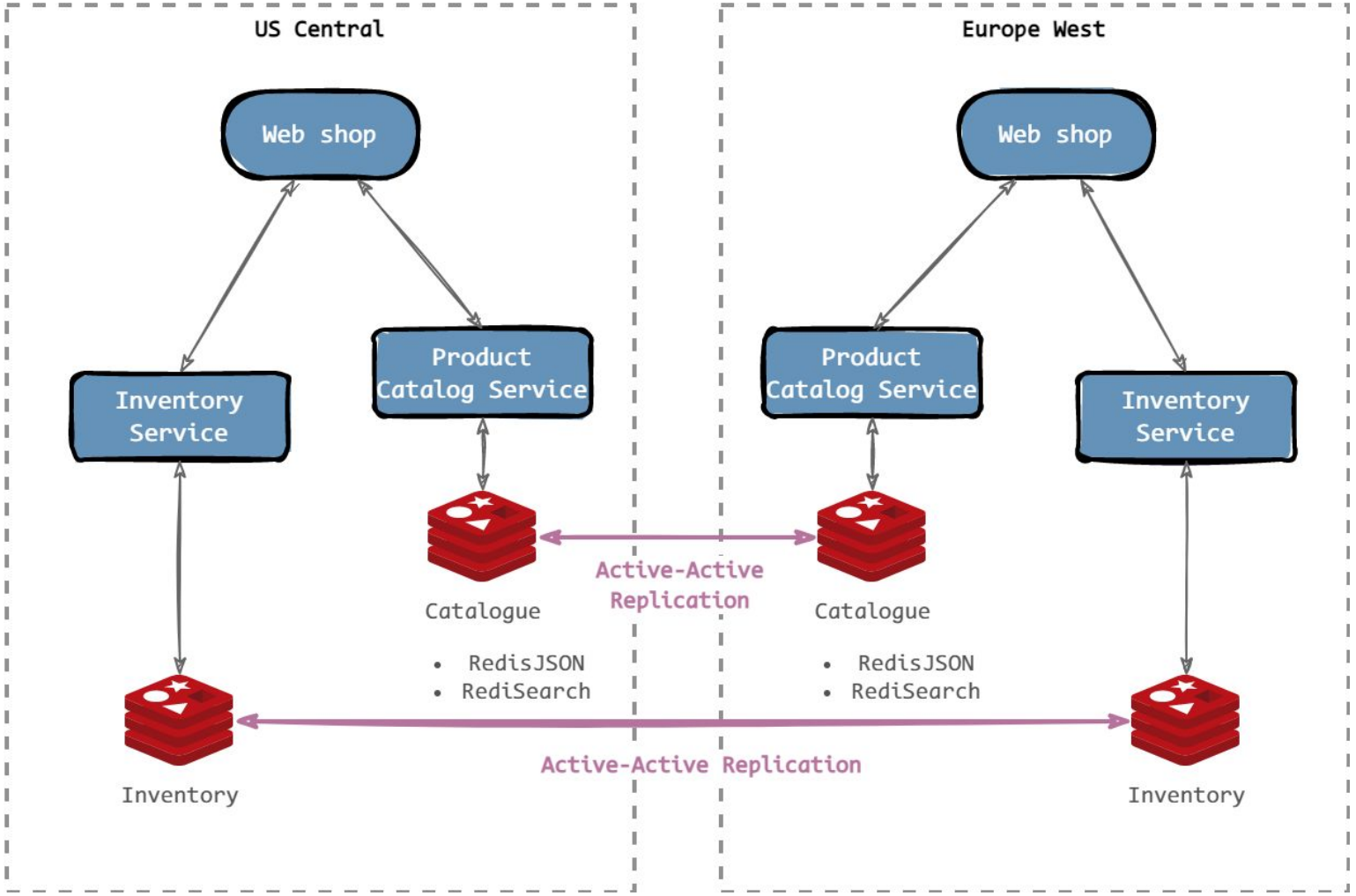


- Explore and interact with your data
- Reduce memory usage
- Monitor commands in real time
- Cluster management

redis

RedisMart Demo

RedisMart Demo Architecture



Redis Real Time Search and JSON

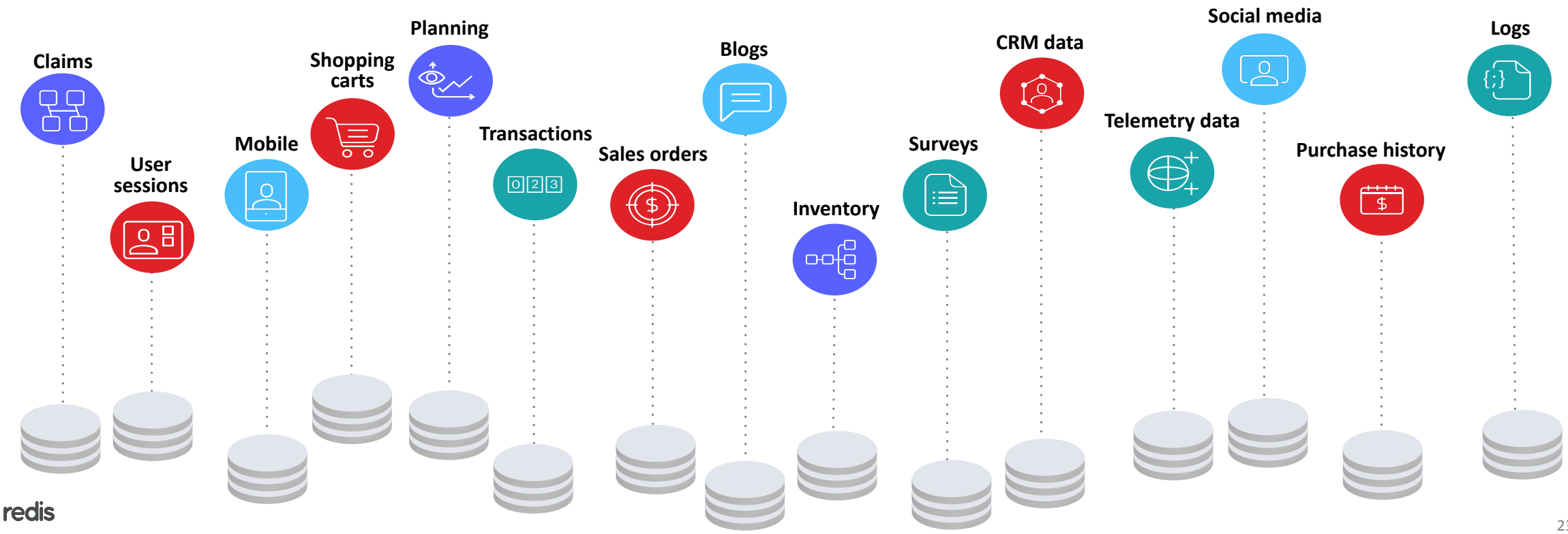
The age of real time data

Businesses are creating unique customer experiences

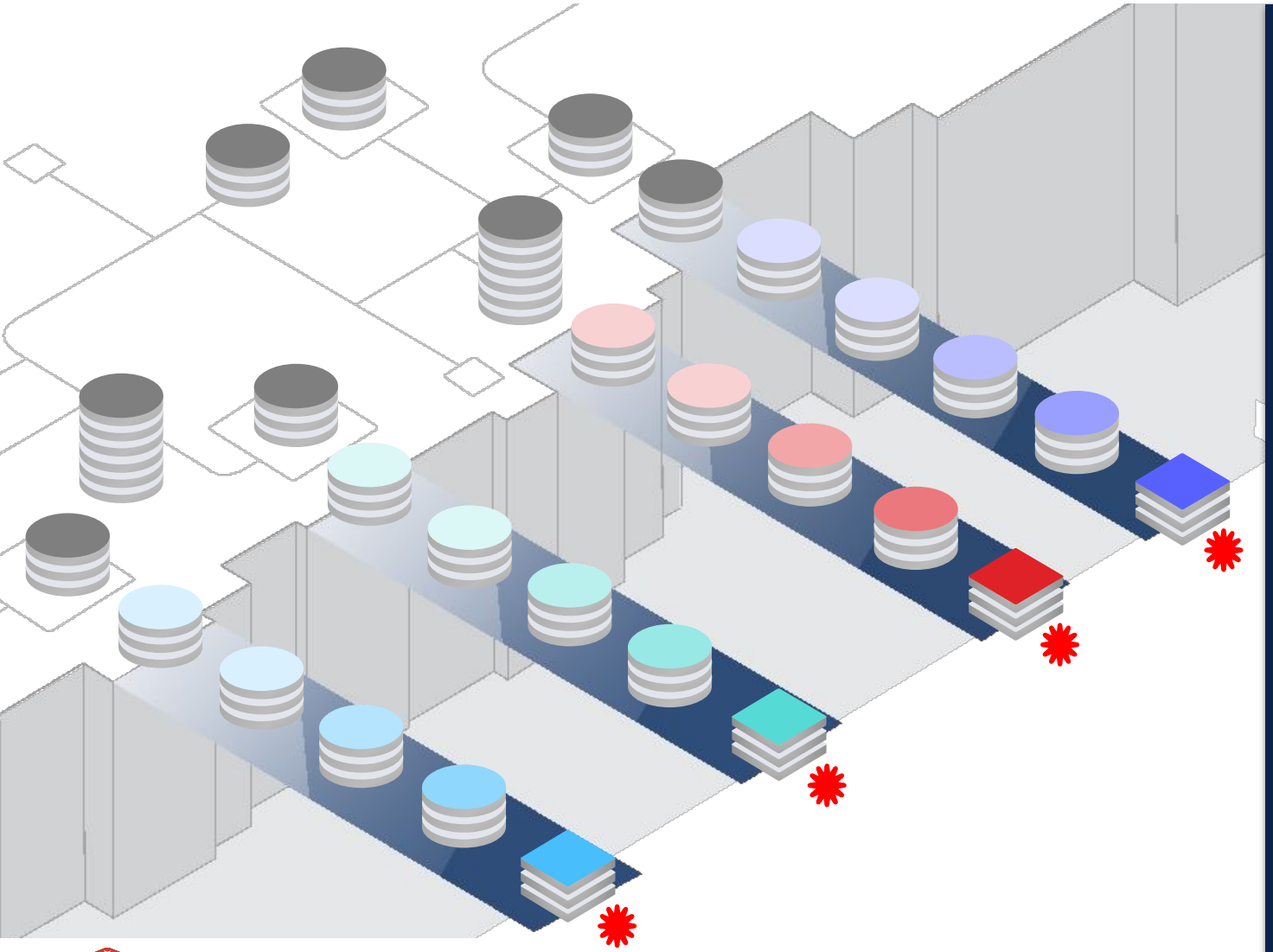
- Analyzing, making predictions, and taking actions against data in real-time enables new use cases
- Latency is a growth killer

According to a recent Customer Experience survey from Emplifi

- **61**% of customers say they would switch after one bad experience
- **86%** of customers would switch after as few as two poor experiences



Data is trapped in slow databases



Difficult for users to search and applications run too slow

Customer experience suffers

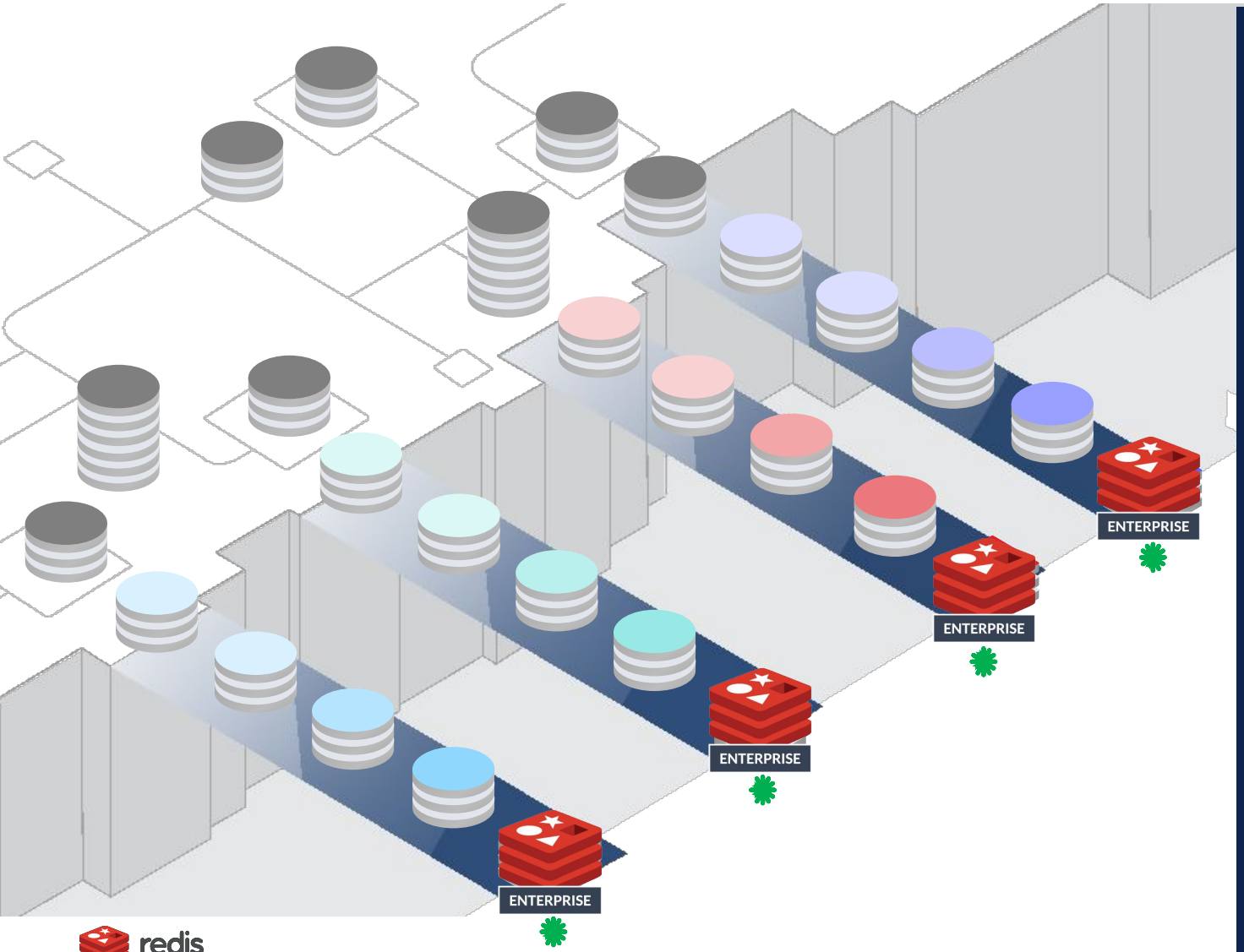
Architecture is too slow to consolidate, aggregate and query data

Stale data loses value

Cannot scale to support serverless or microservices frameworks

Legacy applications hinder growth

Redis search and query unlocks real time data



Sub-millisecond query results even for frequently updated, extremely large datasets

Immediate search results keep customers engaged

Query by properties, full text, numeric ranges, geography, and aggregate in real time

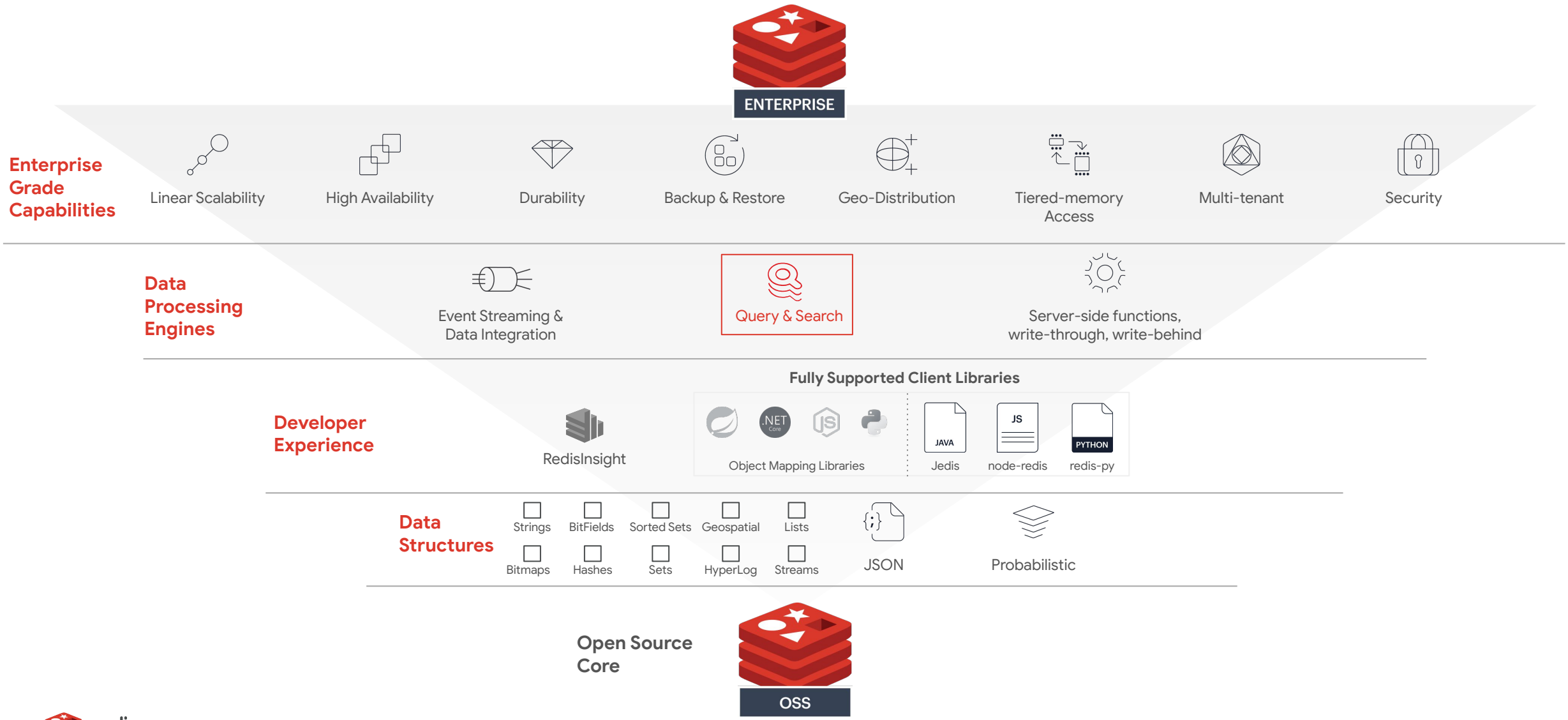
Data is accurate and fresh enabling new use cases

Scalable global data platform supports serverless and microservices frameworks

Enables worldwide business growth and enhanced customer experiences

redis

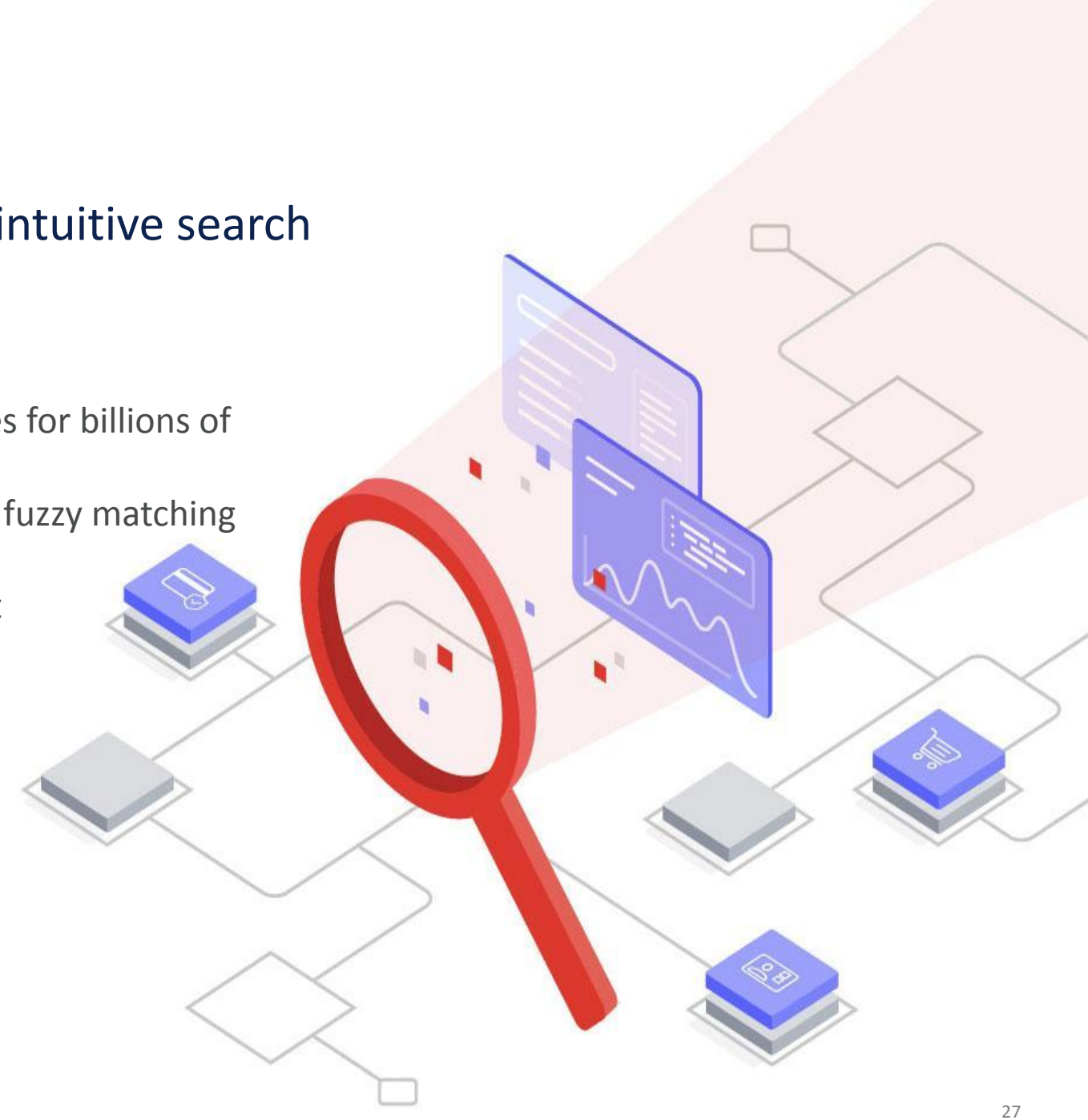
Redis Enterprise includes a real time search engine



Real time search

Better customer experience through fast, intuitive search

- 4x faster results than competitive search solutions
- Scales to easily create primary and secondary indexes for billions of Hash and JSON documents
- Full text search and auto-complete suggestions with fuzzy matching helps users find what they need fast
- Auto-indexing speeds time to market with less effort

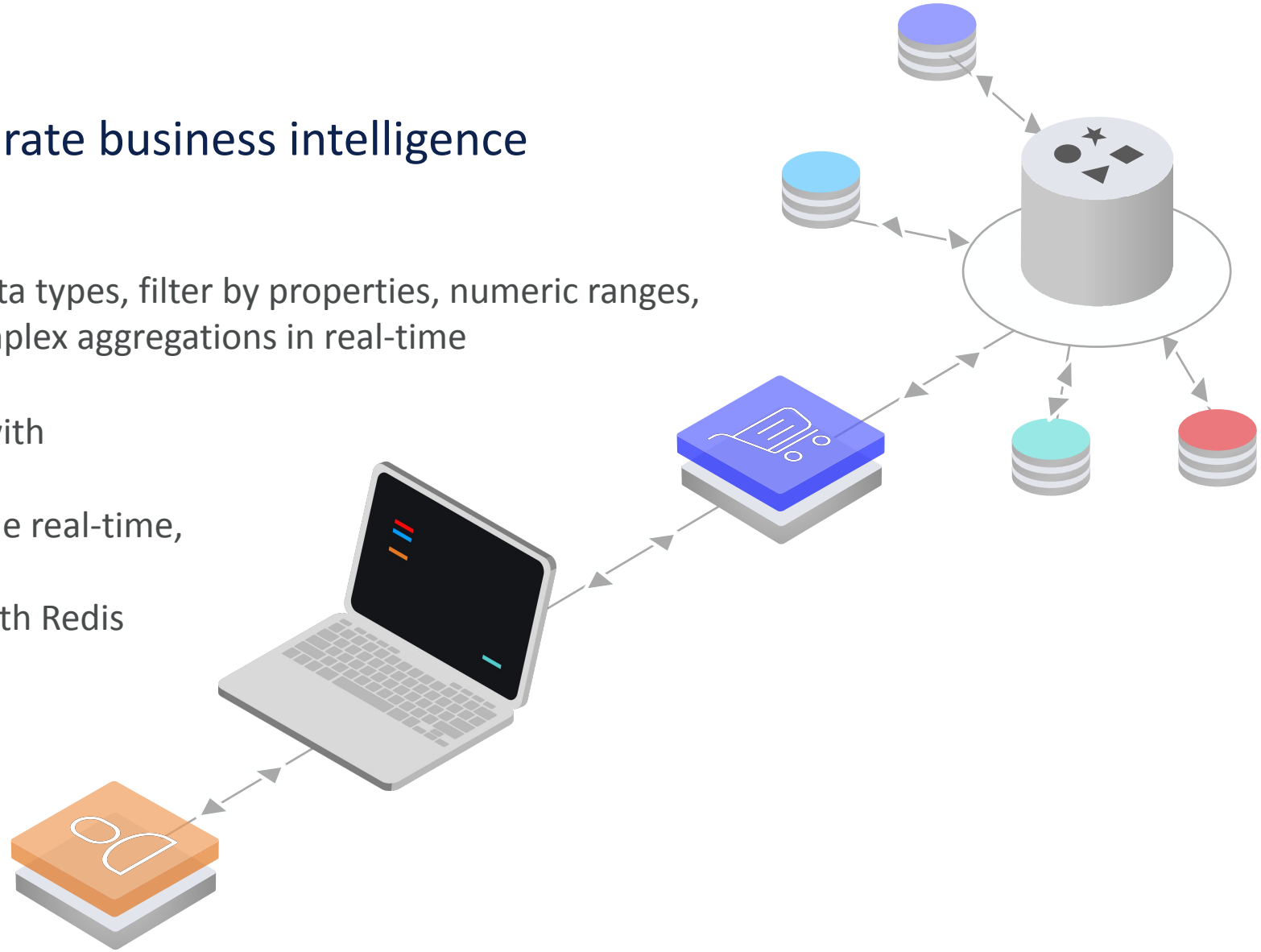


High speed analytics

Deliver the freshest data for accurate business intelligence

Deliver the freshest data, query multiple data types, filter by properties, numeric ranges, and geographical distance and perform complex aggregations in real-time

- Higher quality insight from fresh data with in-memory, sub-millisecond response
- Consolidate external databases onto one real-time, multi-model platform
- Easy analytics dashboard integration with Redis Data Source for Grafana plug-in

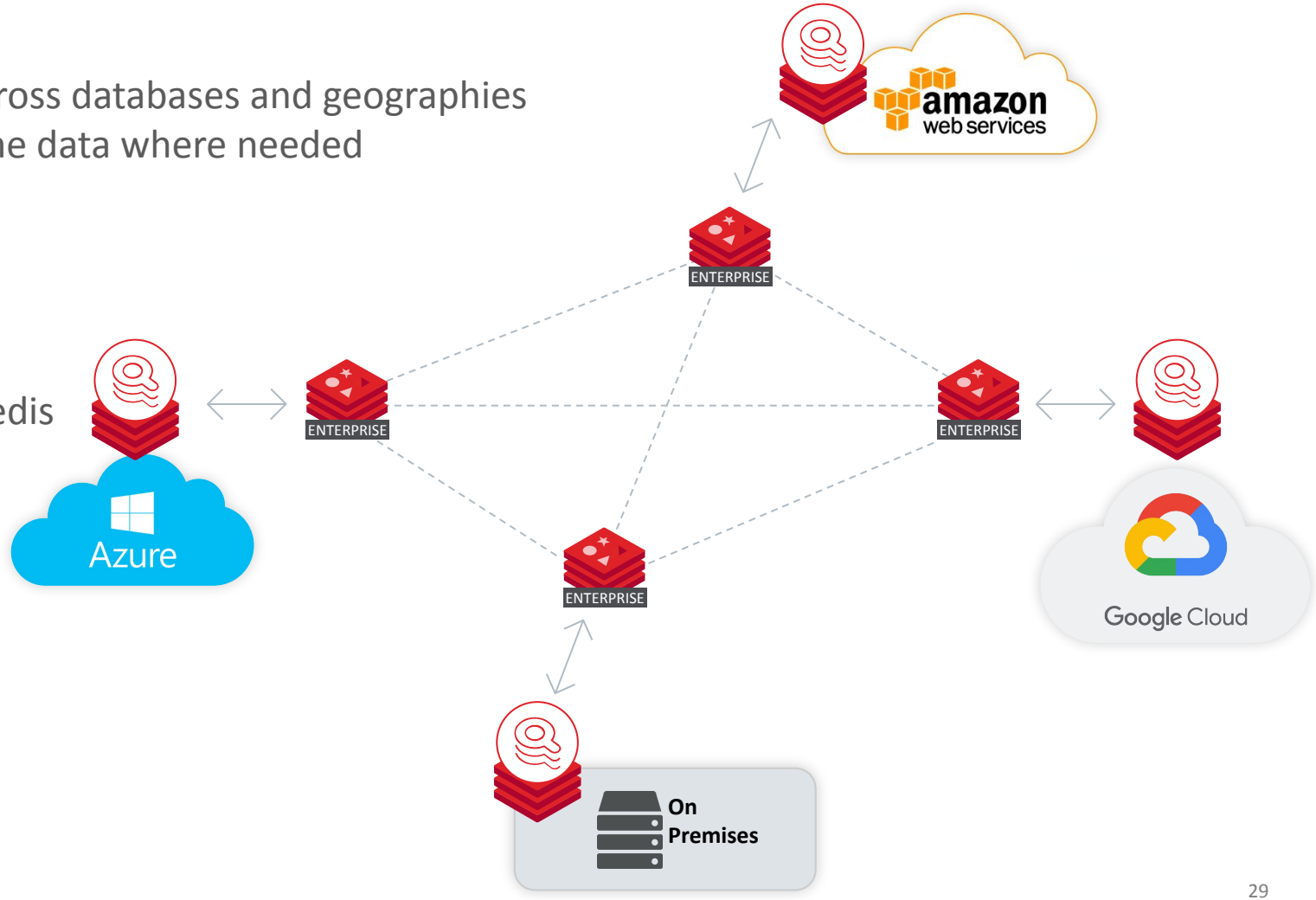


redis

Geographically distributed search

Deliver scale and performance exactly where you need it

1. Active-active database topology:
   - Scales by distributing search load across databases and geographies
   - Stores data locally to deliver real-time data where needed
2. Runs on premises, private or public cloud
3. Available as a fully managed service by Redis
4. Small resource footprint and low latency is ideal for serverless and microservices framework use cases



Fast creation and automatic indexing of secondary keys

- Index any field in the database in real time for faster search results and unique data views
- Multi-field queries with no application code changes
- Once defined indexes are automatically updated, never manage indexes again

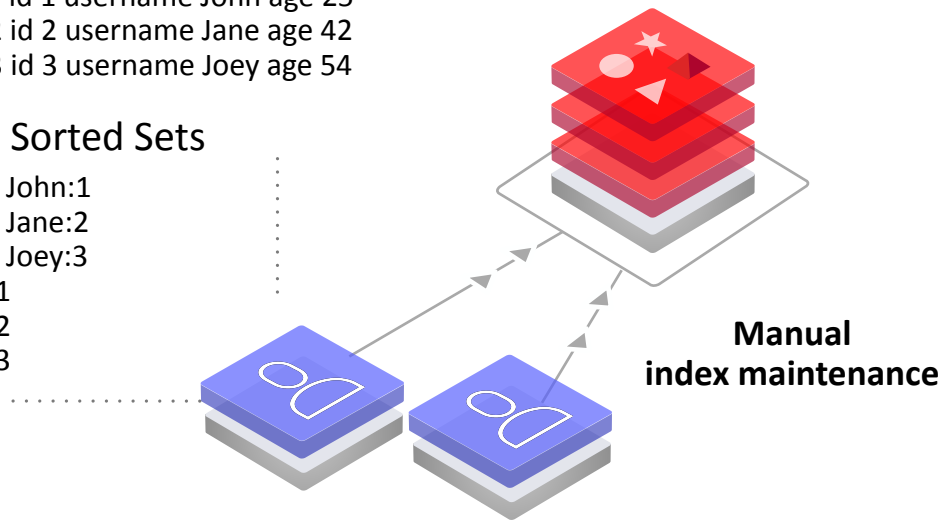
Without RediSearch

Data stored via Hash

HMSET person:1 id 1 username John age 25
HMSET person:2 id 2 username Jane age 42
HMSET person:3 id 3 username Joey age 54

Multiple Indices via Sorted Sets

ZADD nameIdx 0 John:1
ZADD nameIdx 0 Jane:2
ZADD nameIdx 0 Joey:3
ZADD ageIdx 25 1
ZADD ageIdx 42 2
ZADD ageIdx 54 3



Manual index maintenance

Cumbersome multi-key search

ZRANGE ageIdx 20 30 **BYSCORE**
ZRANGE nameIdx [Jo + **BYLEX**
… plus logic to perform set intersection…

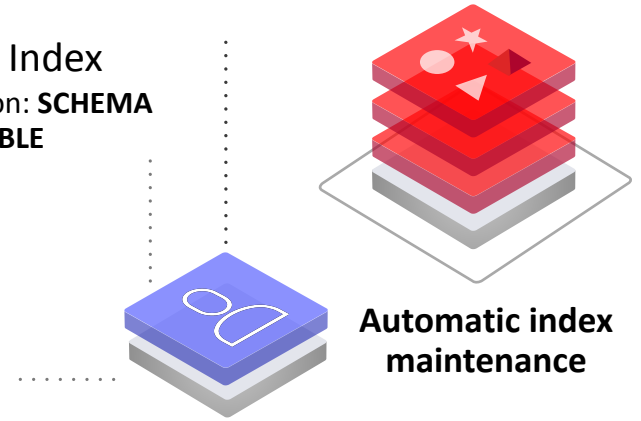
With RediSearch

Native JSON storage

JSON.SET person:1 $ '{"username": "John", "age": 25}'
JSON.SET person:2 $ '{"username": "Jane", "age": 42}'
JSON.SET person:3 $ '{"username": "Joey", "age": 54}'

Multiple secondary keys, Native Index

FT.CREATE idx **ON JSON PREFIX 1** person: **SCHEMA**
$.username **AS** username **TEXT SORTABLE**
$.age **AS** age **NUMERIC SORTABLE**



Automatic index maintenance

Native, optimized search

FT.SEARCH idx "@age:[20 30] @username:Jo*"

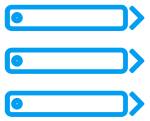
Redis Enterprise real time search capabilities

Indexing

- Secondary index structures for numeric data, text values (tags), and geo locations
- Indexing on multiple fields in docs via a single index
- Declarative indexes
- Incremental **synchronous indexing**
- Inverted index structure for full-text
- Document deletion and updating with index
- Garbage collection

Querying

- Multi fields queries
- Numeric filters and range queries
- Geo radius queries
- Complex Boolean queries with *AND, OR, NOT* operators between sub-queries
- Optional query clauses
- Ask for full document content or just ids
- Aggregations across shards



Full-text search

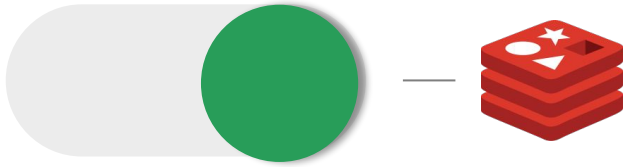
- Document ranking & field weights
- Prefix based searches
- Exact phrase search, or slop based search
- Stemming based query expansion in many languages
- Support for custom functions for query
- Expansion and scoring
- Limiting searches to specific doc fields
- Spell-checking and auto-completion dictionaries

Lab 2: Basic JSON Operations

Lab 3: Basic Search Operations

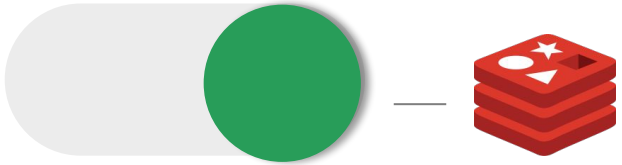
Redis Real Time Search Use Cases

Redis real time search and query use cases



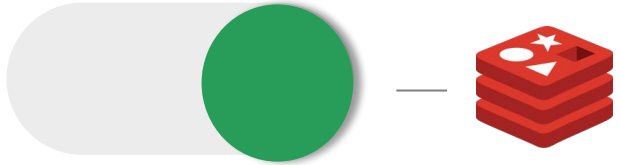
Master data table lookups

Eliminate application latency from frequently-used master tables



Real time analytics

Deliver fresh data for accurate operational analytics and business intelligence



Customer 360°

Create a unified real-time view to enhance customer service response and support real-time transactions

Master data table lookups

Enhance customer experience by eliminating application latency

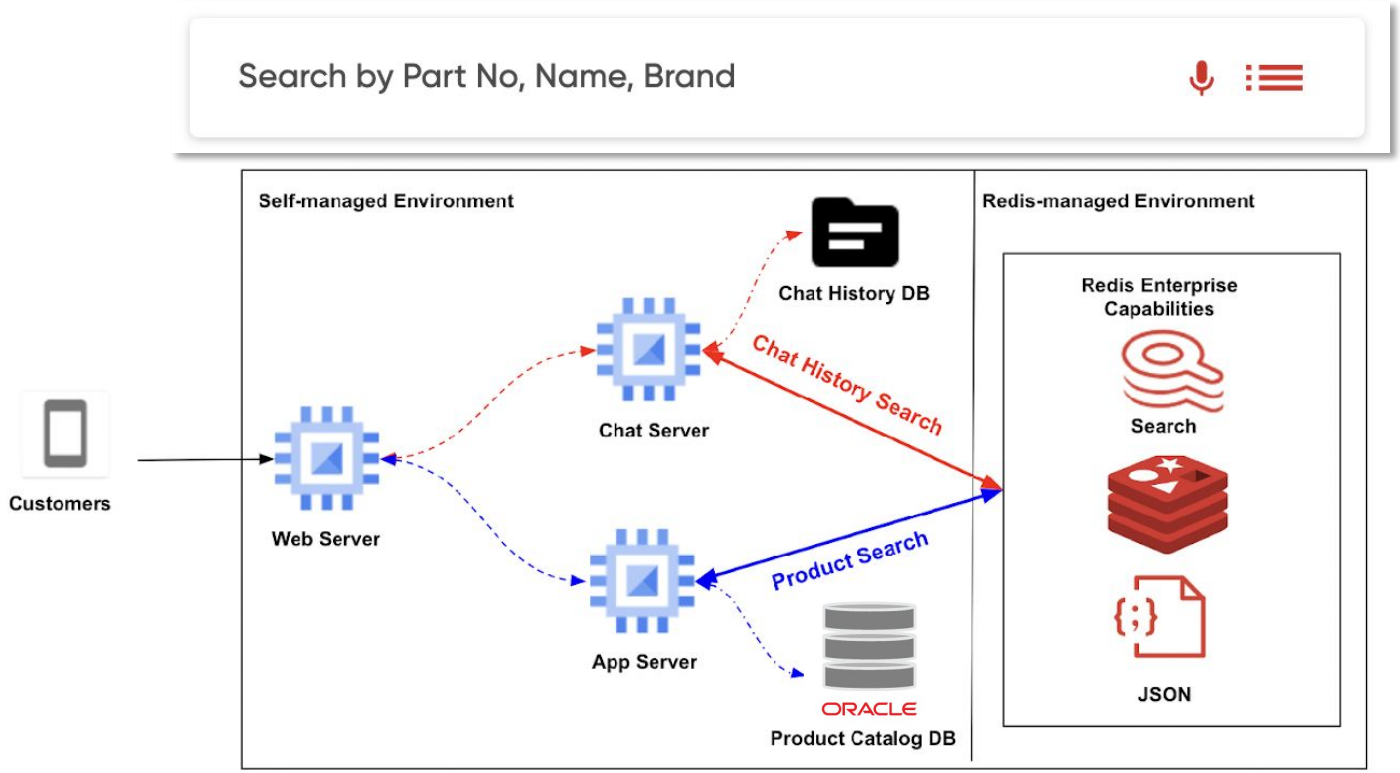
Challenges

- **Application performance problems** from large master data table lookups
- **Slow user search** on tens of millions of primary and secondary keys creates poor user experience

Redis provides

- **Better application user experience** by eliminating master table performance bottlenecks with sub-millisecond response
- **Supports query of millions of primary and secondary keys** delivering flexible, real-time query and exact match performance
- **Increase ROI** by deferring costly infrastructure enhancements due to application performance and scaling issues

Redis delivers real-time search of millions of parts



CASE STUDY

Master data tables lookup of over 10M parts

National Auto Retailer operates thousands of car service centers

Business challenges

- Retail customer search of over 10M parts became too slow
- Customer service team needed real-time access to massive online chat history

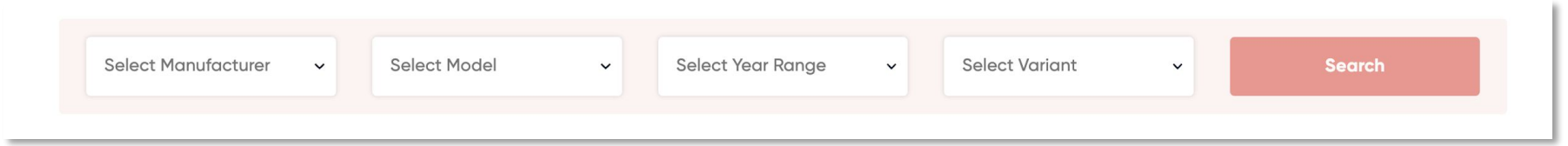
Technical challenges

- Performance of ElasticSearch got too slow as parts list grew
- Searching chat histories using JavaScript became slow and cumbersome

Redis Benefits

Enhanced customer experience through real time search

- Search response improved 5x by eliminating master table bottlenecks
- Chat history is now searchable in real time by Customer Service
- Faceted and auto-complete search using fuzzy matching happens in less than 30ms



Increase ROI with RediSearch

- 5x the speed, half the cost of Elasticsearch
- Cost effective to scale globally

"We needed a fast data layer to search across 10 million spare parts and power our customer service chats."

– Vice President
of Engineering, National Auto Retailer

Real time analytics

Deliver higher quality data for analytics, machine learning, and business intelligence

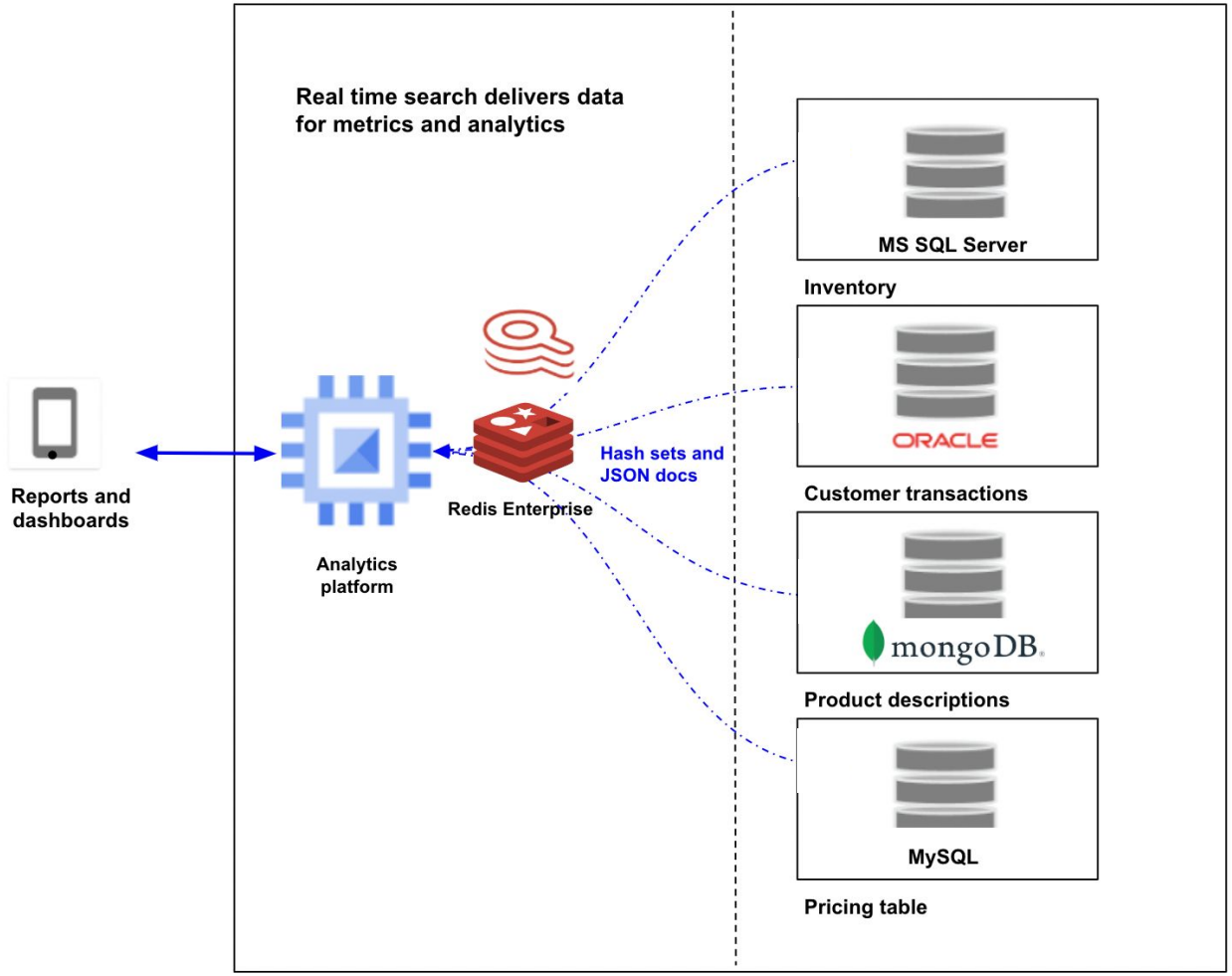
Challenges

- **Slow queries** deliver poor quality metrics for analysis for fraud, gaming, financial services, etc
- **Data is spread** across multiple databases
- **Limited search and query** capabilities due to complex data infrastructure

Redis provides

- **Performance** for accurate Insights from more timely data
- **Consolidation** of multiple systems of record into one real-time global data platform
- **Flexibility** to search real-time on any field with flexible secondary indexes

"What is the top product purchased in the past 10 minutes?"



CASE STUDY

Real time data for AI-based Market Analytics

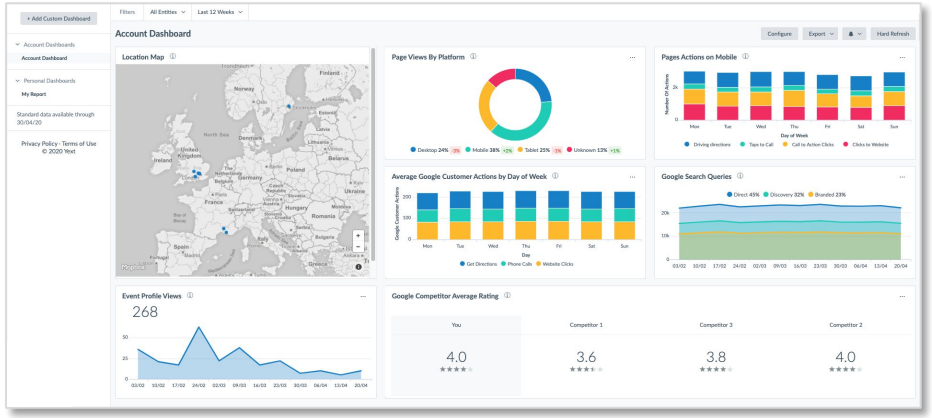
Global leader in marketing and website marketing intelligence

Business challenges

- Require low latency queries to support AI-based marketing and web operational analytics
- Need to provide full text search across multiple continents

Technical challenges

- Consolidate data from multiple systems of record as fast as possible for accurate analytics
- Running 1,000+ microservices so additional query latency is unacceptable



Redis benefits

- **Enhanced customer experience** with auto-suggestion and faceted search in real time of millions of records
- **Consolidated data globally** with multicloud, active-active database topology
- **Low latency search and query scales** to support modern microservices framework

"Because Redis…is pushing out features very important to us. We use the hosted version of Redis search and because they're able to operate in multiple clouds like we are, we can use them anywhere we have a point of presence."

– Vice President of Engineering

Customer 360°

Deliver exceptional customer service and personalization

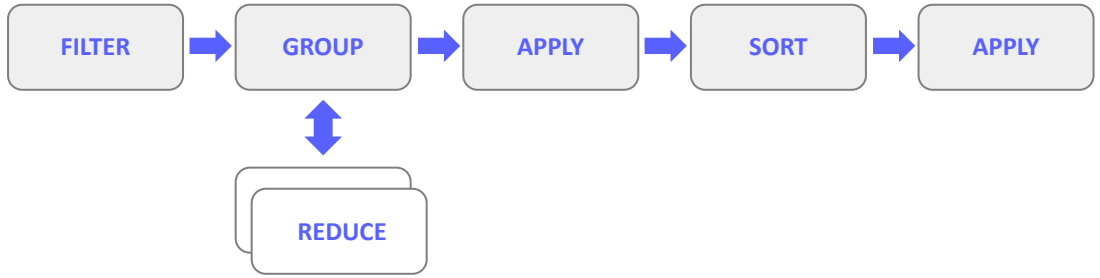
Challenges

- **Poor service** due to slow customer lookup of exact matches for email, mobile #, or government ID
- **Unhappy customers** don't like to wait for answers from customer service or be put on hold

Redis provides

- **Scalable real time customer lookups** for extremely large datasets and enhanced customer service
- **Low-latency data aggregation** in real time to create new customer views of dynamic data
- **Deeper personalization** with time to analyze user session, profile, recent actions, and transaction history to predict best upsell and cross sell possibilities

AGGREGATION SUPPORTS DATA TRANSFORMATION PIPELINE



GROUPBY {Userid}

Accounts table
Userid:7773777
Account1:000329991
Account2:043944
Account3:94440003304
Account4:0034485
Userid:7773778
Account…
Account…

```
FT.AGGREGATE
  {index_name:string}
  {query_string:string}
  [WITHSCHEMA] [VERBATIM]
  [LOAD {nargs:integer} {property:string} ...]
  [GROUPBY
    {nargs:integer} {property:string} ...
    REDUCE
      {FUNC:string}
      {nargs:integer} {arg:string} ...
      [AS {name:string}]
    ...
  ] ...
  [SORTBY
    {nargs:integer} {string} ...
    [MAX {num:integer}] ...
  ] ...
  [APPLY
    {EXPR:string}
    AS {name:string}
  ] ...
  [FILTER {EXPR:string}] ...
  [LIMIT {offset:integer} {num:integer} ] ...
```

CASE STUDY

Customer 360° for Real Time Banking

Large retail bank with $10B in annual revenue and 70,000 employees

Business challenges

- The bank needed to scale Customer 360° real-time application for over 28M customers
- Current Oracle solution could not scale to 6 Billion transactions per month

Technical challenges

- Oracle complex data structures were too slow and costly for core jobs and processes
- Require front-end database layer to take load away from Oracle RDBMS

Redis benefits

- **A huge performance boost**: the new application scales 1000x better than Oracle to support real-time search and transactions for millions of customers
- **Aggregates customer data** from disparate data sources to create dynamic real-time summary reports for tens of thousands of simultaneous users

"Started using Redis for performance, then as a scalable search engine and aggregator. It provides us a transaction platform that can scale our bank infrastructure years into the future."

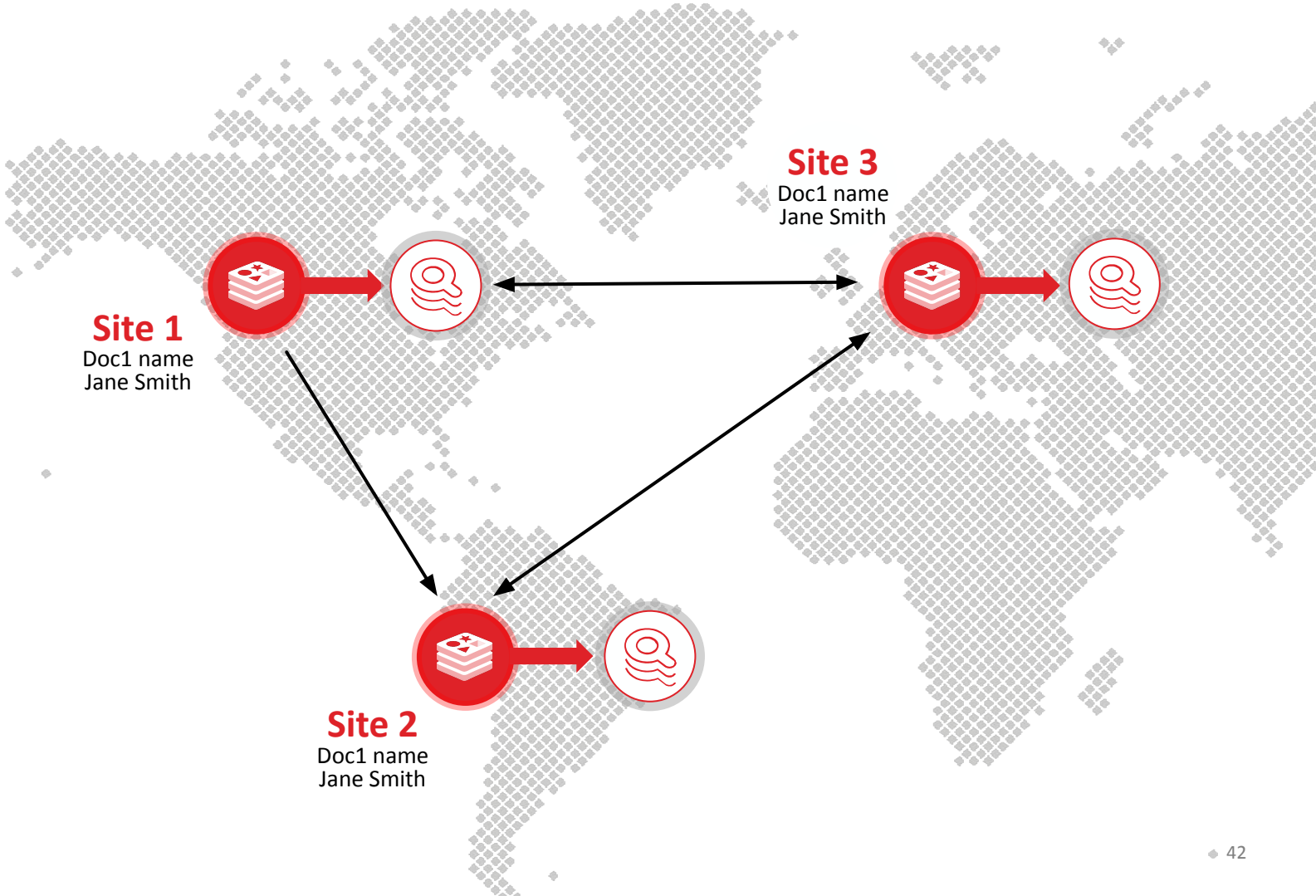
–Vice President
of Engineering, Large Online Bank

Enterprise-grade availability, scalability, and security

Redis Enterprise delivers multi-site, active-active, real time search engine

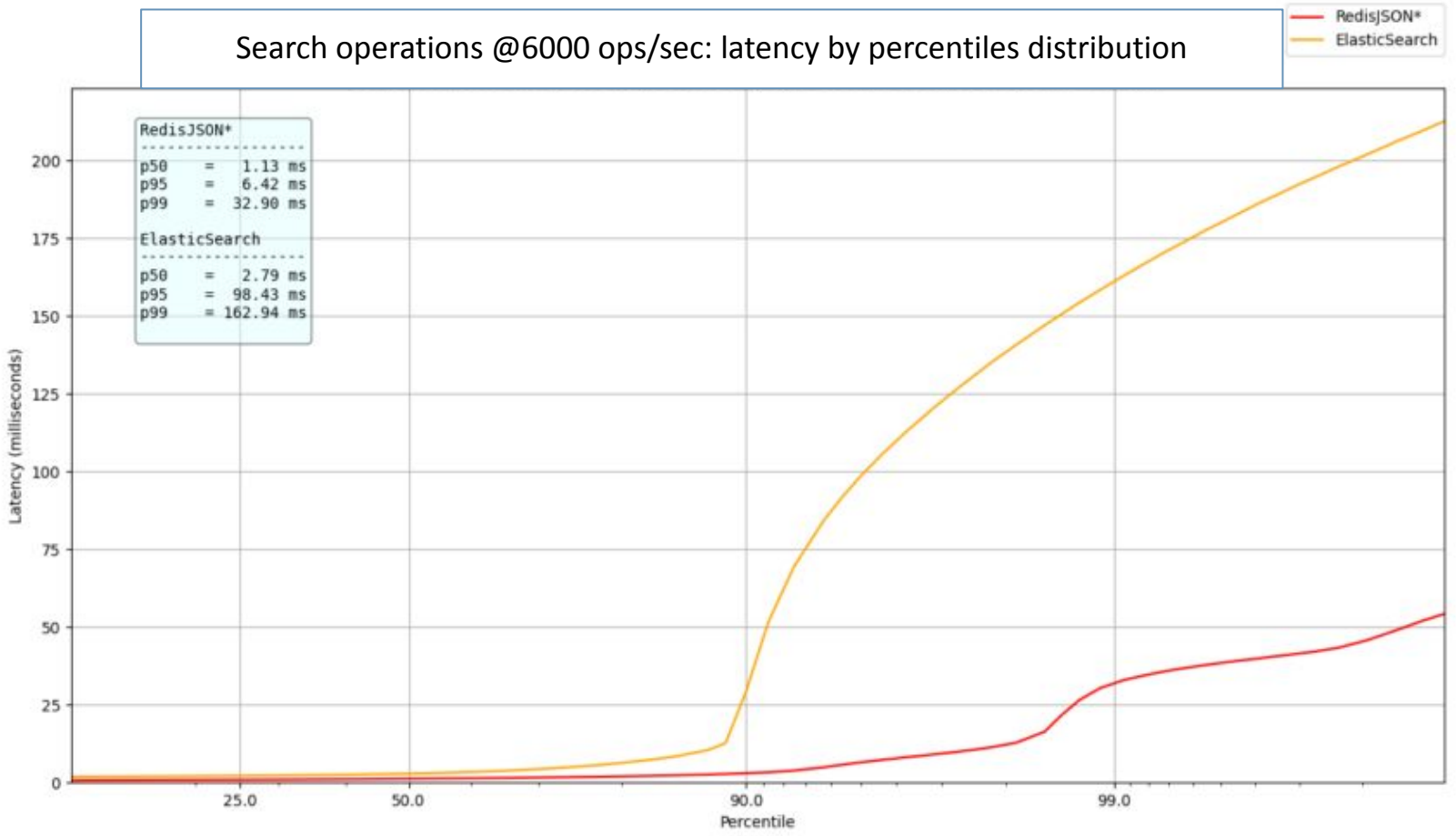
Support real time needs anywhere in the world

- Deliver applications anywhere with local latencies
- Ensure global data consistency with seamless conflict resolution via conflict-free replicated datatypes (CRDTs)
- Scale real-time federated search across the globe



Scalable real time search performance

Build lightening fast applications with JSON

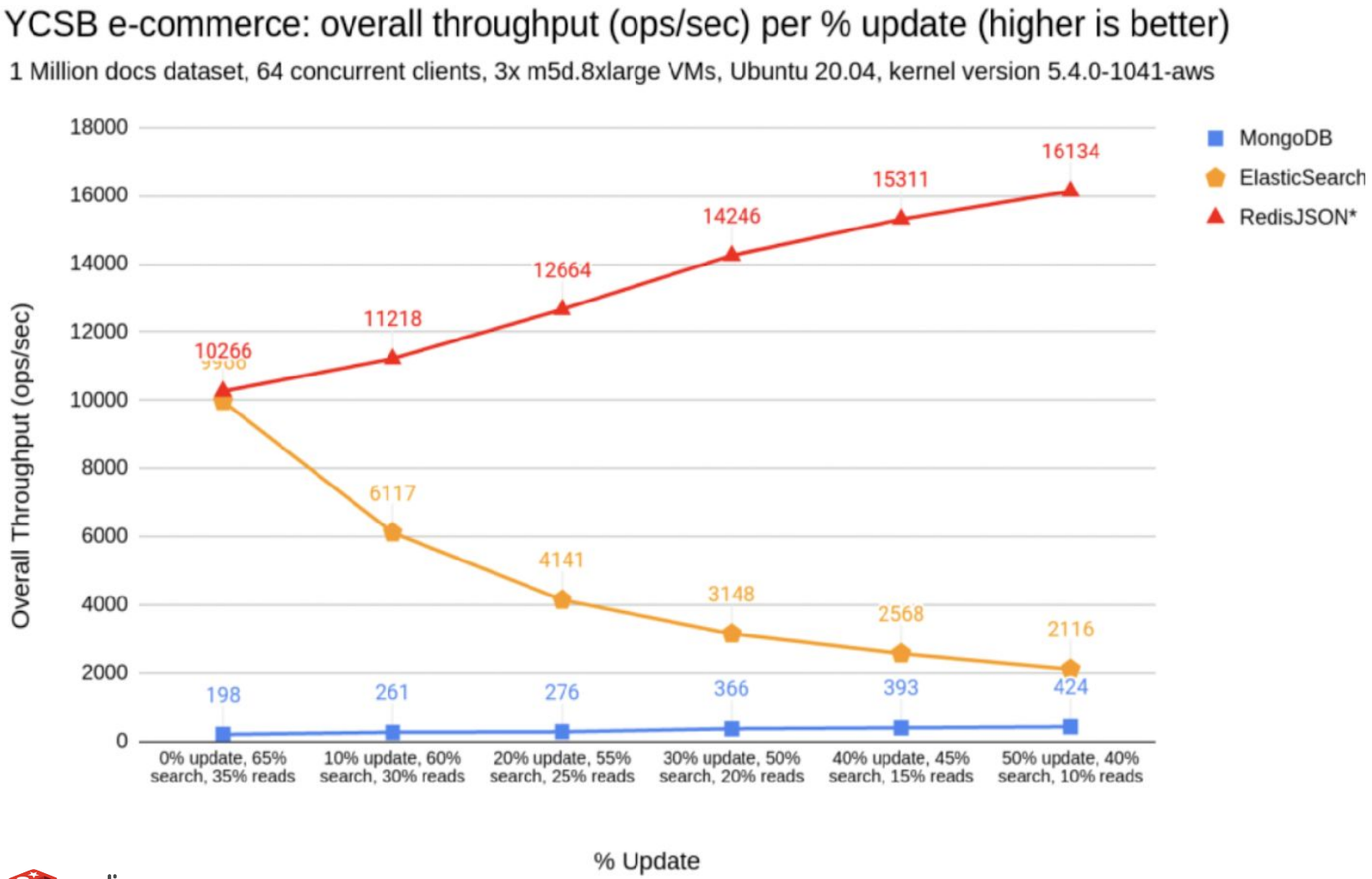


https://redis.com/blog/redisjson-public-preview-performance-benchmarking//

- **Redis is written in C** providing better performance at scale, Elasticsearch is mostly written in slower Java
- **Redis is 500% faster** with a p99 below 33ms vs. a much higher 163ms for Elasticsearch
- **Elasticsearch latency dramatically increases** perhaps due to suffering garbage collection and cache misses on the >= p90 percentiles

Highest throughput in mixed workloads

Redis outperforms the competition in mixed workloads



- **Elasticsearch throughput *shrinks* 45% in mixed workloads** where Redis goes up (10% update, 60% search, 30% read)
- **Redis throughput scales 800%** greater then Elasticsearch as updates increase (50% update, 40% search, 10% read)
- **Redis low latency is ideal** for real world use cases where data is both updated and searched at the same time

https://redis.com/blog/redisjson-public-preview-performance-benchmarking//

Redis Enterprise built-in search capabilities

Deliver scalable search, better intelligence and high-speed global applications

Real time search and query

Sub-millisecond results for even frequently updated, extremely large datasets

Immediate results keep customers engaged

High speed analytics

Data is accurate and fresh enabling new use cases

Data is accurate and fresh enabling new use cases

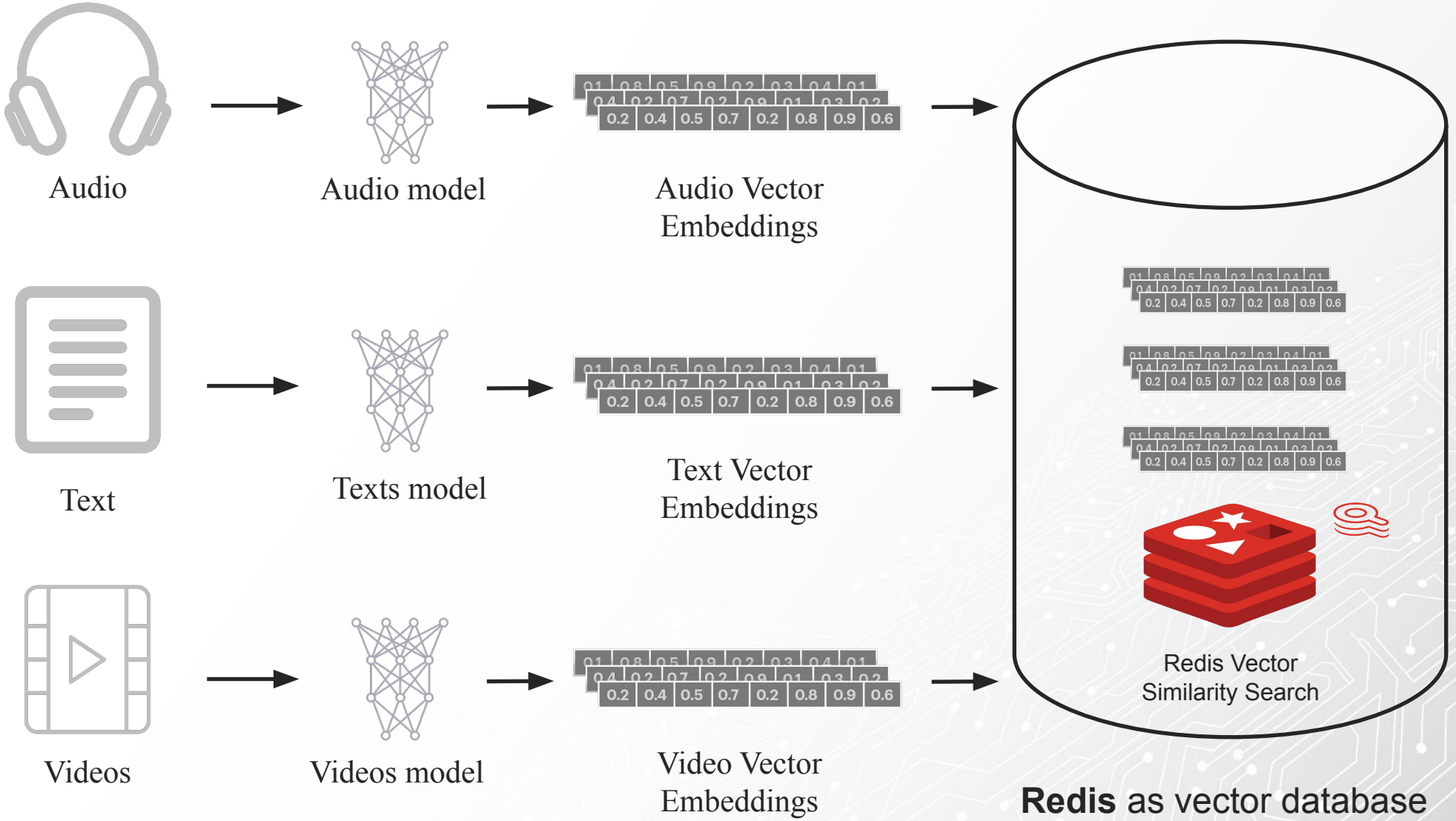
Globally distributed search

Federated search across globally distributed databases located anywhere, on-premises or multicloud. Supports serverless and microservices frameworks.

Supports business growth and enhanced customer experiences

Redis as a vector database for similarity searches

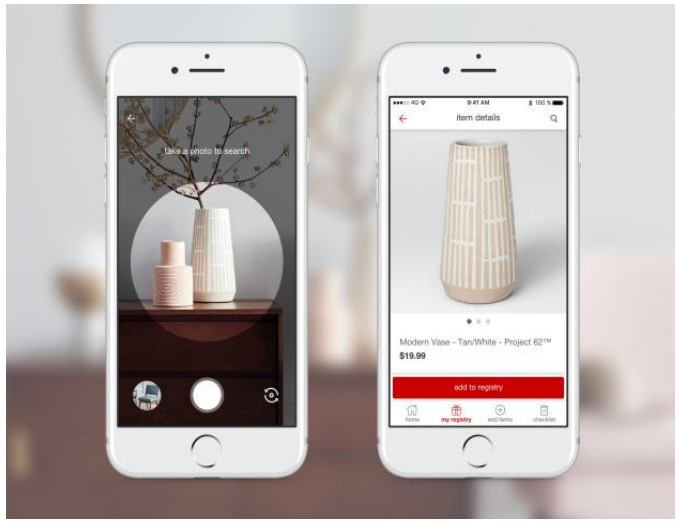
Redis as a vector database for similarity searches



Redis as vector database for vector similarity search

Everyday use cases powered by similarity searches

Visual Search



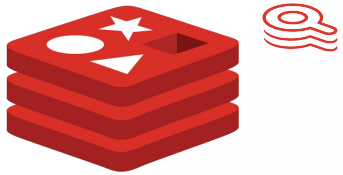
Natural Language Search

Customer Service: semantic search, Q&A, content recommendation

Recommenders

Recommend: Similar products, brands, properties

Vector Similarity Search Capabilities



Redis Vector Similarity Search as part of Public Preview – RedisSearch 2.4

Storing Vectors as **BLOBs** in Redis Hashes

Two indexing methods supported:

Hierarchical Navigable Small World (HNSW) or **ANN**

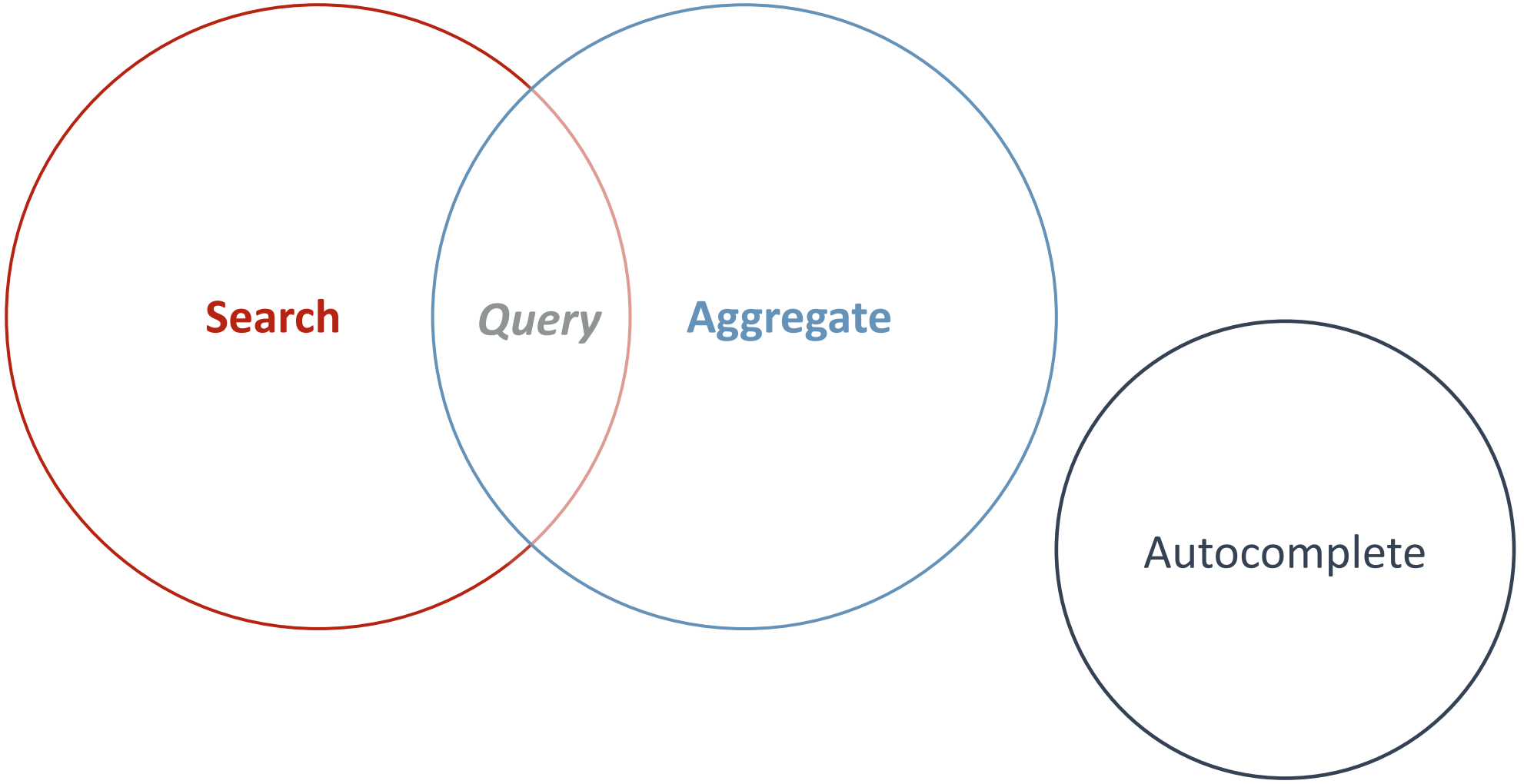
Flat or **KNN**

Three distance metrics:
Euclidean
Cosine
Internal Product

Hybrid Queries
Combine vector similarity with filtering on hash fields as GEO, TEXT, etc

Redis Real Time Search Technical Deep Dive

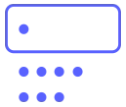
Redis Search can do three things.



Primary features

Search and filter

- Full-text indexing of multiple fields in document
- Geo filtering with Redis geo-commands
- Retrieval of either full document content or IDs
- Faceted search
- Field weighted or ad-hoc document ranking



Auto-complete

- Suggestions with fuzzy prefix matching
- Phonetic matching and synonyms

Aggregate

- Aggregate across millions of documents
- Intuitive pipes that can be combined in any order
- Pipes can group by/reduce, map, sort and filter

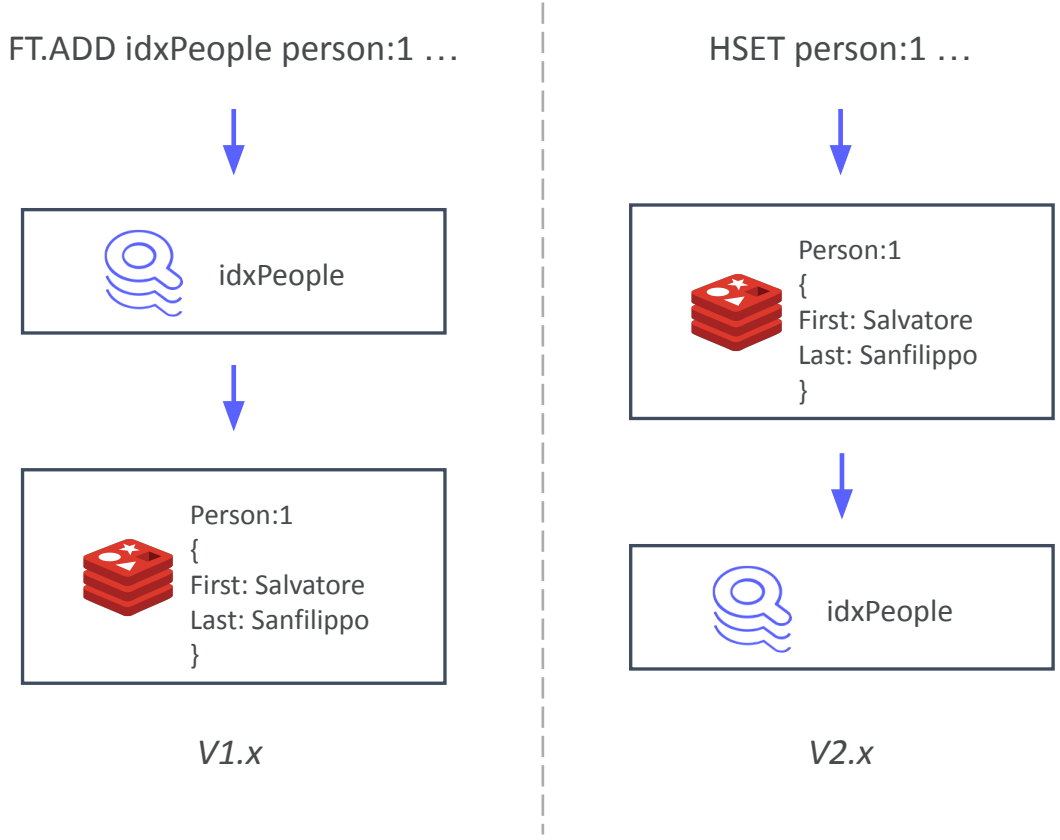
Real-time insertion

- Partial and conditional document indexing in atomic operations
- Document deletion and updating with index garbage collection

Instantly search any Redis database

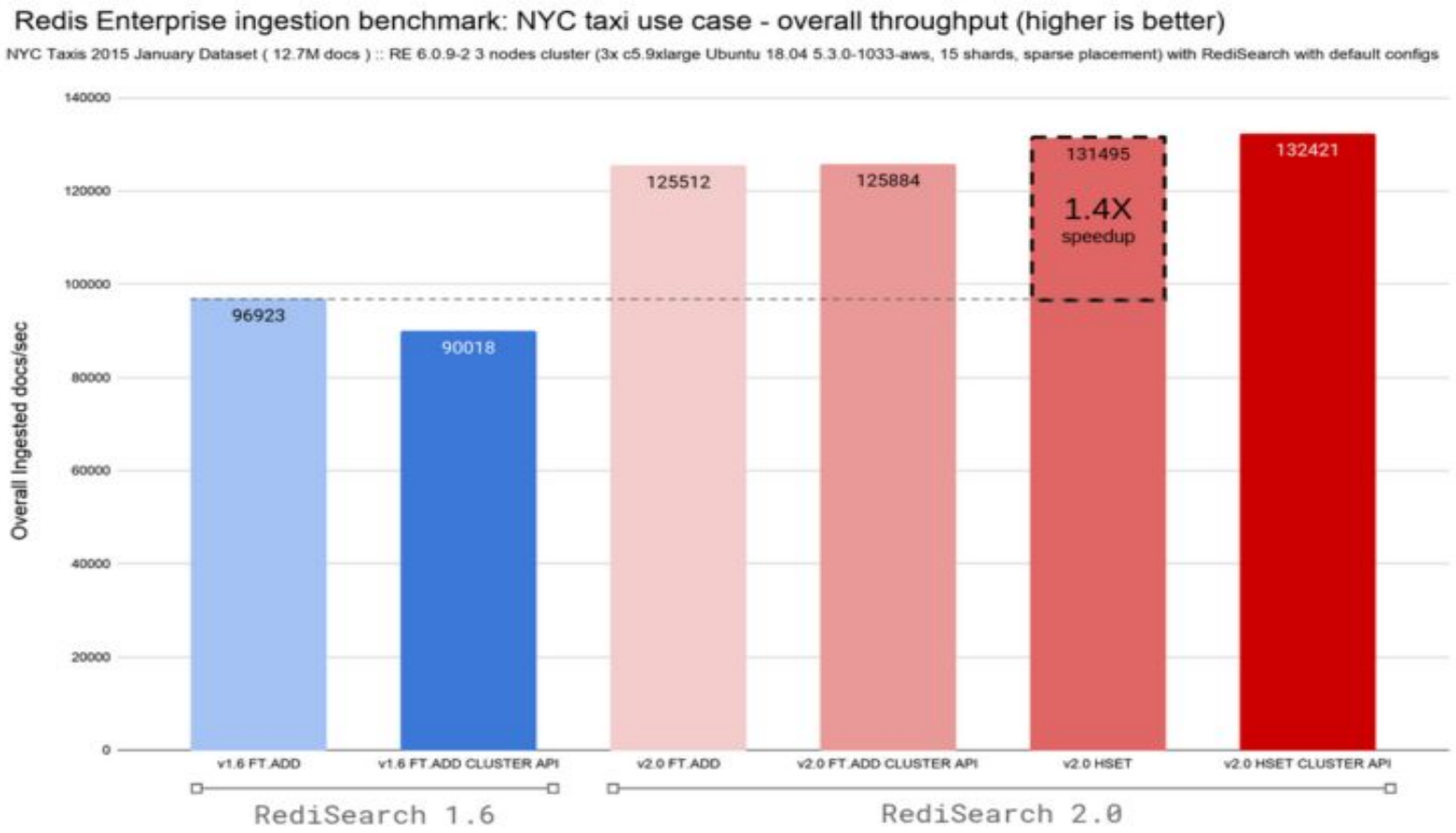
New architecture makes it easy to add search and indexing

- Follows data written in Redis and automatically indexes existing data with no performance degradation
- Add indexing to Redis without changes to application code
- Documents are automatically deleted out of the index on expiry



The fastest full text search

132K docs/sec at an overall p50 ingestion latency of 0.4ms



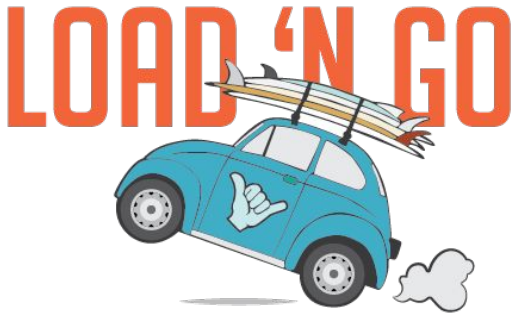
- Overall throughput and latency improvements are 2.4x times faster than RediSearch 1.6
- 4x faster in querying two-word searches, and 58% faster in indexing compared to ElasticSearch;

It’s easy!

You have multiple ways to run RediSearch, and clients are available for almost all languages:

- building from sources and installing it inside an existing Redis Instance
- Redis Cloud
- Redis Enterprise
- Docker for raw Redis Commands API

```
> docker run -it --rm --name redis-search-2 \
   -p 6379:6379 \
   redislabs/redisearch:2.0.2
```



Currently available Libraries

Language	Library	Author	License	Stars
Python	redis-py	Redis	BSD	Star Unable to select next Github token from pool
Python	redis-om	Redis	BSD-3-Clause	Star 617
Java	Jedis	Redis	MIT	Star 11k
Java	redis-om-spring	Redis	BSD-3-Clause	Star 319
Java (Lettuce based)	LettuceMod	Redis	Apache-2.0	Star 30
Java	redis-modules-java	dengliming	Apache-2.0	Star 71
Go	redisearch-go	Redis	BSD	Star 226

And more…

Create index - FT.CREATE

Basic index:

```
FT.CREATE idx:movie ON hash PREFIX 1 "movie:" SCHEMA title TEXT SORTABLE
release_year NUMERIC SORTABLE rating NUMERIC SORTABLE genre TAG SORTABLE
```

Indexing with filter:

```
FT.CREATE idx:drama ON hash PREFIX 1 "movie:"
FILTER "@genre=='Drama' && @release_year>=2010 && @release_year<2020"
SCHEMA title TEXT SORTABLE release_year NUMERIC SORTABLE
```

Ephemeral indexing:

B2B shopping cart session data :
Implemented by a leading sport apparel for
Manufacturing use cases

```
FT.CREATE idx:drama ON hash PREFIX 1 "movie:"
FILTER "@genre=='Drama' && @release_year>=2010 && @release_year<2020"
TEMPORARY 60
SCHEMA title TEXT SORTABLE release_year NUMERIC SORTABLE
```

Query data - FT.SEARCH

basic word matching:

```
FT.SEARCH "idx:movie" “fast”

FT.SEARCH "idx:movie" @title:“fast”
```

negation:

```
FT.SEARCH idx:movie "Star wars -jedi" RETURN 2 title release_year
```

condition, sorting, pagination:

```
FT.SEARCH "idx:movie"
"@genre:{Action|Thriller}"  @release_year: [2017 2020] LIMIT 0 10
SORTBY release_year ASC RETURN 3 title release_year genre
```

Query data - FT.SEARCH

Geospatial:

```
> FT.CREATE idx:theater ON hash PREFIX 1 "theater:" SCHEMA name TEXT SORTABLE location GEO
OK
> FT.SEARCH "idx:theater" "@location:[-73.9798156 40.7614367 400 m]" RETURN 2 name address
 1) (integer) 5
 2) "theater:88"
 3) 1) "name"
    2) "Snapple Theater Center"
    3) "address"
    4) "1627 Broadway"
 4) "theater:30"
 5) 1) "name"
    2) "Ed Sullivan Theater"
    3) "address"
    4) "1697 Broadway"
 6) "theater:115"
 7) ....
```

Aggregate data - FT.AGGREGATE

Group by
Reduce
Sort by

```
> FT.AGGREGATE "idx:movie" "*"
GROUPBY 1 @release_year
REDUCE COUNT 0 AS nb_of_movies
SORTBY 2 @release_year DESC

 1) (integer) 60
 2) 1) "release_year"
    2) "2019"
    3) "nb_of_movies"
    4) "12"
 3) 1) "release_year"
    2) "2018"
    3) "nb_of_movies"
    4) "14"
 4) 1) "release_year"
    2) "2017"
    3) "nb_of_movies"
    4) "12"
 5) 1) "release_year"
    2) "2016"
    3) "nb_of_movies"
    4) "88"
 ……
```

Grouping & Manipulation

- **Reducers**:
  - COUNT
  - COUNT_DISTINCT
  - COUNT_DISTINCTISH
  - SUM
  - MIN
  - MAX
  - AVG
  - STDDEV
  - QUANTILE
  - TOLIST
  - FIRST_VALUE
  - RANDOM_SAMPLE

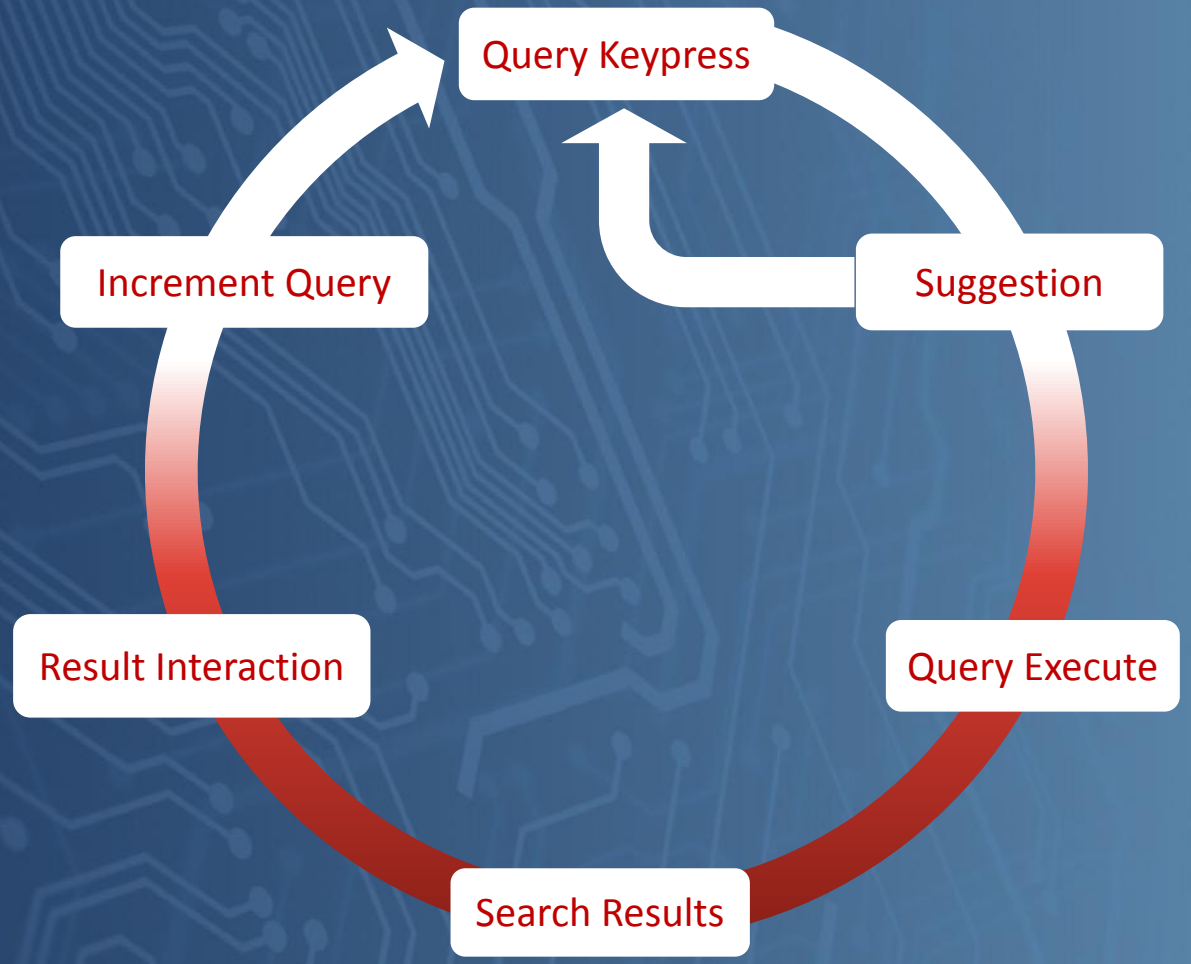
- **Manipulate**
  - Strings
    - substr(upper('hello world'),0,3) -> “HEL”
  - Numbers w/ Arithmetic
    - sqrt(log(foo) * floor(@bar/baz)) + (3^@qaz % 6)
  - Timestamp to Calendar
    - timefmt(@mytimestamp, "%b %d %Y %H:%M:%S”) -> Feb 24 2018 00:05:48

Full-text search

- Prefix: `hel* world`
- Infix/suffix: `*sun* *ing`
- Wildcard: `"w'foo*bar?'"`
- Stemming: going -> go, gone
- Phonetic: on many different languages
- Scoring: assigning score to your subclass on your weight
- Fuzzy search:

```
> FT.SEARCH idx:movie " %gdfather% " RETURN 2 title release_year
1) (integer) 1
2) "movie:11003"
3) 1) "title"
   2) "The Godfather"
   3) "release_year"
   4) "1972"
```

User-Direct Search Autocomplete Cycle



Auto-completion

Simple API:
FT.SUGADD
FT.SUGGET
FT.SUGDEL

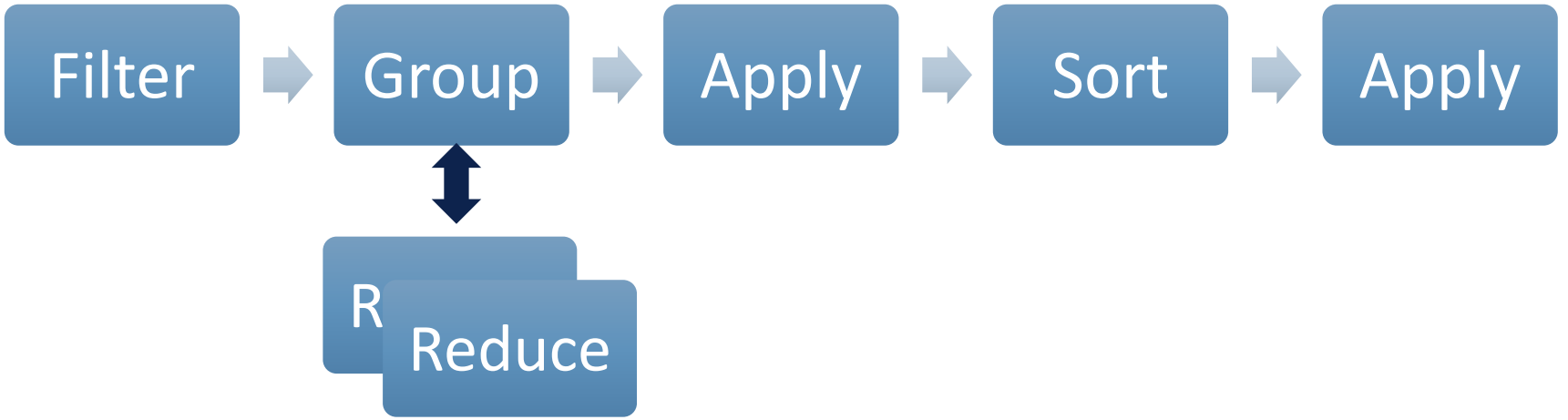
```
> FT.SUGADD ac "Star Wars: Episode V - The Empire Strikes Back" 1
(integer) 2
> FT.SUGADD ac "Star Wars: Episode IV - A New Hope" 2
(integer) 3
> FT.SUGADD ac "Star Wars: Episode VII - The Force Awakens" 3
(integer) 4

> FT.SUGGET ac "star wars"
1) "Star Wars: Episode VII - The Force Awakens"
2) "Star Wars: Episode IV - A New Hope"
3) "Star Wars: Episode V - The Empire Strikes Back"

> FT.SUGGET ac "star wars" MAX 2 WITHSCORES
1) "Star Wars: Episode VII - The Force Awakens"
2) "0.5144957304000854 5"
3) "Star Wars: Episode IV - A New Hope"
4) "0.3922322690486908"
```

Aggregations

- Processes and transforms
- Same query language as search
- Can group, sort and apply transformations
- Follows pipeline of composable actions:



Redis Search on Redis JSON

Storage Layer Requirements

Data Shape

The data comes in as a nested structure that looks like

```
{
     id: ,
     org_id: ,
     schema_name: user,
     data: {
          id: 1, // queryable
          first_name: Greg, // queryable
          external_json: "", // NOT queryable
          foreign_key: 6
     }, // need to query fields inside
     metadata: { agent, provenance } // json blobs - need to query
fields inside
}
```

Outreach -
#1 Sales Engagement platform

Data size:

Our current production cluster stores 11.5M -> grow to up to 30M records, up to 5MB in size. 90-180 day per-record TTL

SLA:

Writing

- 100-1000 writes per minute
- records are mutable (compliance updates)
- records are deletable (customer exit)

Reading

- ~100 reads per minute

Creating Index

```
FT.CREATE idx:user1
ON JSON
  PREFIX 1 user:
SCHEMA
  $.data.first_name AS firstName TEXT
  $.data.last_name AS lastName TEXT
  $.data.foreign_key AS foreignKey TEXT
  $.metadata.agent.addr AS ipAddr TEXT SORTABLE
  $.metadata.agent.port AS agentport NUMERIC SORTABLE
  $.metadata.agent.email AS email TEXT SORTABLE
  $.metadata.agent.tags.dc AS dc NUMERIC SORTABLE
  $.metadata.agent.tags.role AS agentRole TEXT SORTABLE
  $.metadata.provenance.extension.*.url AS provenanceURL TEXT
  $.metadata.provenance.countryCode AS countryCd TEXT SORTABLE
  $.activeStatus AS status TAG SORTABLE
  $.age AS age NUMERIC SORTABLE
  $.metadata.provenance.lastUpdated AS provDate NUMERIC SORTABLE
```

- **Binary filters** - exact match on strings e.g. email or model type
- **Text search** - text search on fields inside data block

```
FT.SEARCH "idx:user1" @email:"brandiguzman/@/example/./org"
FT.SEARCH "idx:user1" @ipAddr:"210/.153/.22/.235"
```

- **Range filters** - filter on date ranges inside data or metadata blocks
- **Pagination** - limiting result sets and being able to navigate with a window

```
FT.SEARCH "idx:user1" @provDate:[1627061856,1627062856] LIMIT 0 10 return 1 firstName
FT.SEARCH "idx:user1" @firstName:Jason LIMIT 0 10 RETURN 1 $.data.external_json.payload
```

- **Sort** - sort results by fields in data and metadata blocks

  FT.SEARCH "idx:user1" "@foreignKey:285 @dc:[1 5]" SORTBY provDate DESC

- **Counting** - Need to know numbers of certain event types, based on filters listed above

  FT.AGGREGATE "idx:user1" @age:[25,50] GROUPBY 1 @agentRole REDUCE COUNT 0 as num SORTBY 2 @num DESC

  FT.AGGREGATE "idx:user1" @agentRole:"trader" GROUPBY 2 @agentRole @dc REDUCE COUNT 0 AS num SORTBY 2 @num DESC

Real world testimony of Redis Search

Redis Search powers business-critical applications with real-time performance

American worldwide clothing and accessories retailer

Fast searching and aggregations over millions of store/sku combinations enables real-time inventory management and order fulfillment

American health insurance provider

Provider search filters based on geographic location, spoken language, and specialty, as well as use on claim fraud detection

And many more testimonies from industries in Travel, Public services, etc.

"We wanted a fast layer to serve our data, we wanted to be able to search millions and millions of records really quickly…Redis Labs checked all the boxes for us."

IT Director of Product Architecture, American worldwide clothing and accessories retailer.

Lab 4: Advanced JSON

Lab 5: Advanced Search

Redis Real Time Search in Production

Cloud Provides Enormous Value

- Agility: **build new digital apps quicker** to gain new revenue streams
- **Consolidate data silos** to gain new visibility and streamline business
- **Simplify operations** and reduce TCO
- **Scale** your business up and down instantaneously with elasticity.

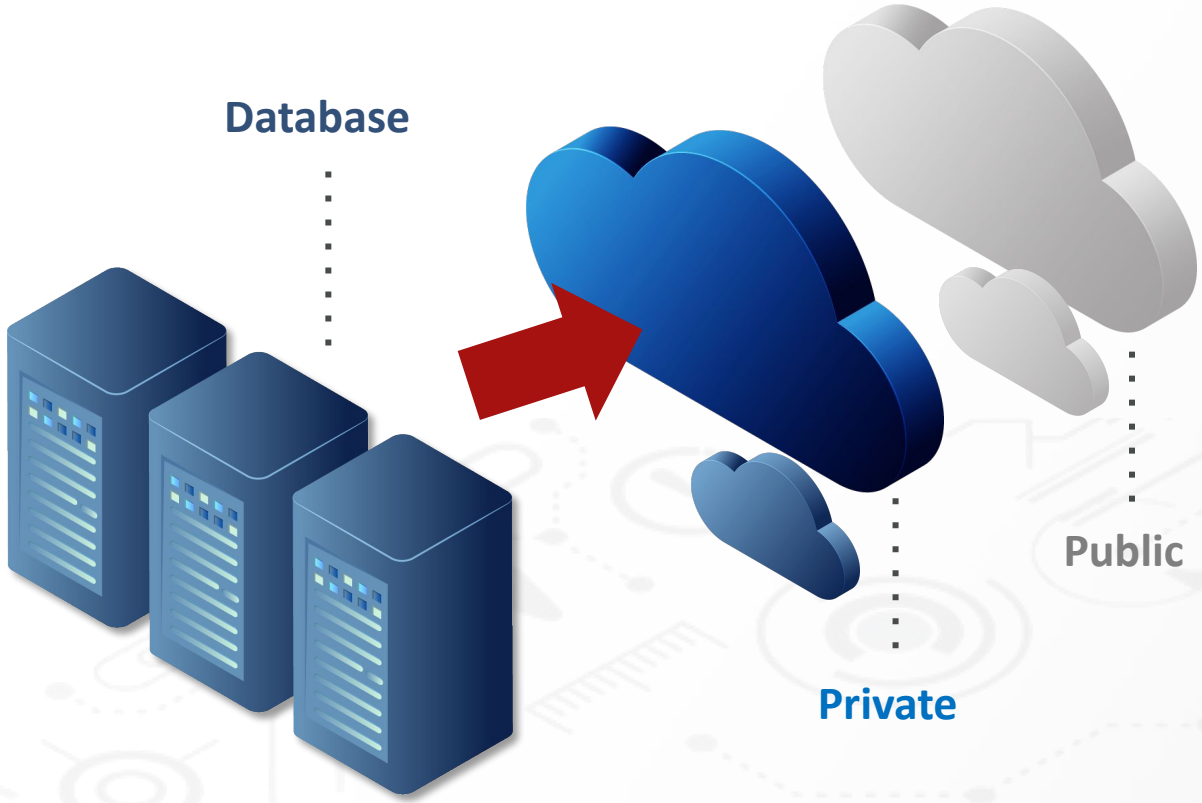
On-Ramps to the Cloud

Cloud Migration

Building Cloud-Native Apps

Cloud Migration is Rife With Challenges

Migration

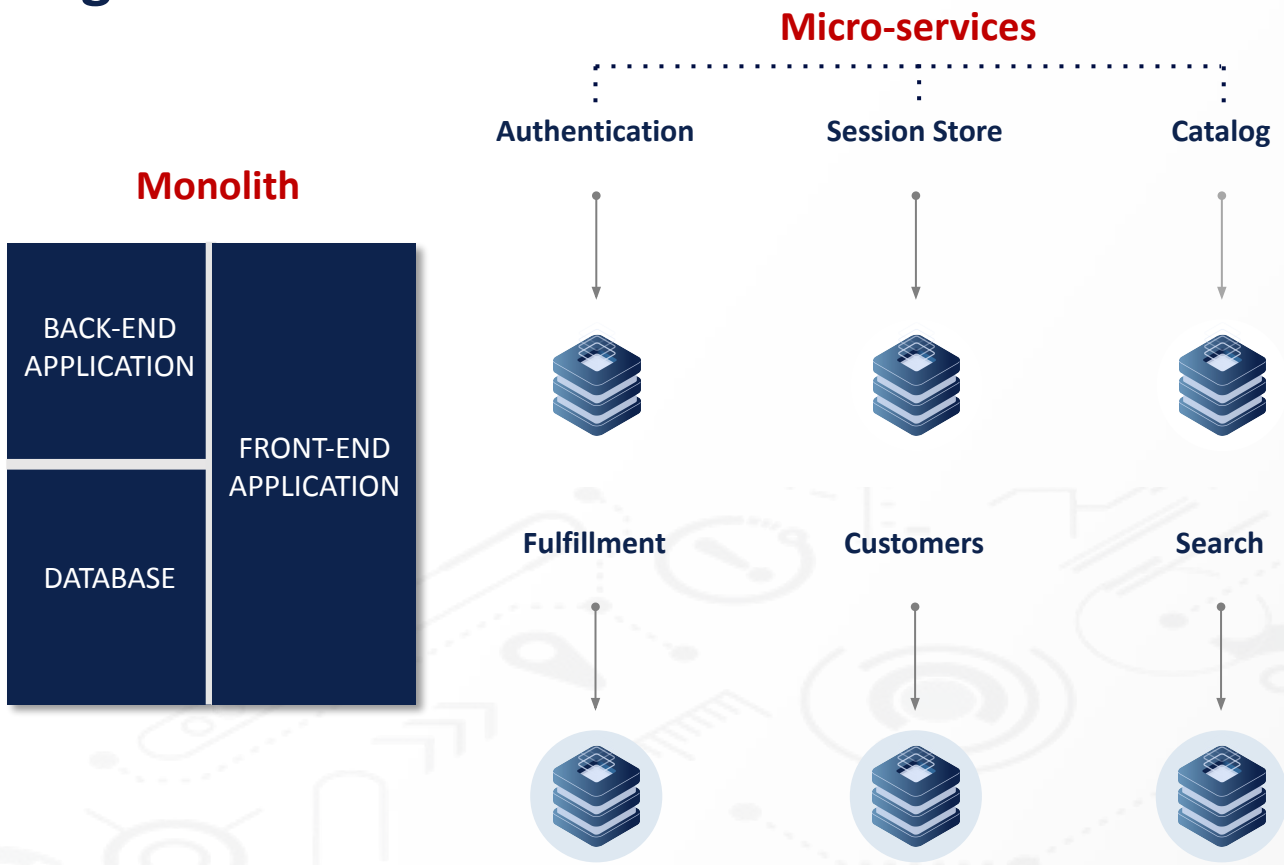


CHALLENGES

- Hybrid Compatibility
- Data Silos
- Availability
- Cost

Microservices Architecture Compounds Complexity

Migration



CHALLENGES

- Expensive
- Complex
- Slow

Cloud-Native Brings New Sets of Challenges

Cloud-Native

- **Microservices**
- **Containers**
- **Serverless**

CHALLENGES

- Hyper-scale and elasticity
- Hybrid
- Multi-cloud
- Global data distribution

Most Data Layers Are Not Built for The Full Cloud Journey

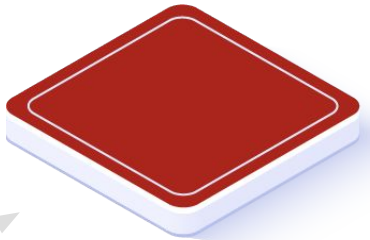
Non Real-Time

Claim Real-Time

Not Built for Hybrid and Multi-Cloud

Do More With Redis Enterprise Cloud

Applications for Data Economy

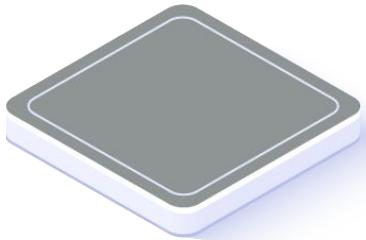


- AI-serving
- Programmable data operations across data models
- AI-powered real-time applications

Data Models for Application Modernization

- JSON, Timeseries, Streams, Graph, and BloomFilter
- Extensive indexing and search capabilities
- Unprecedented performance with single unified interface

Real-time DB and Enterprise Caching

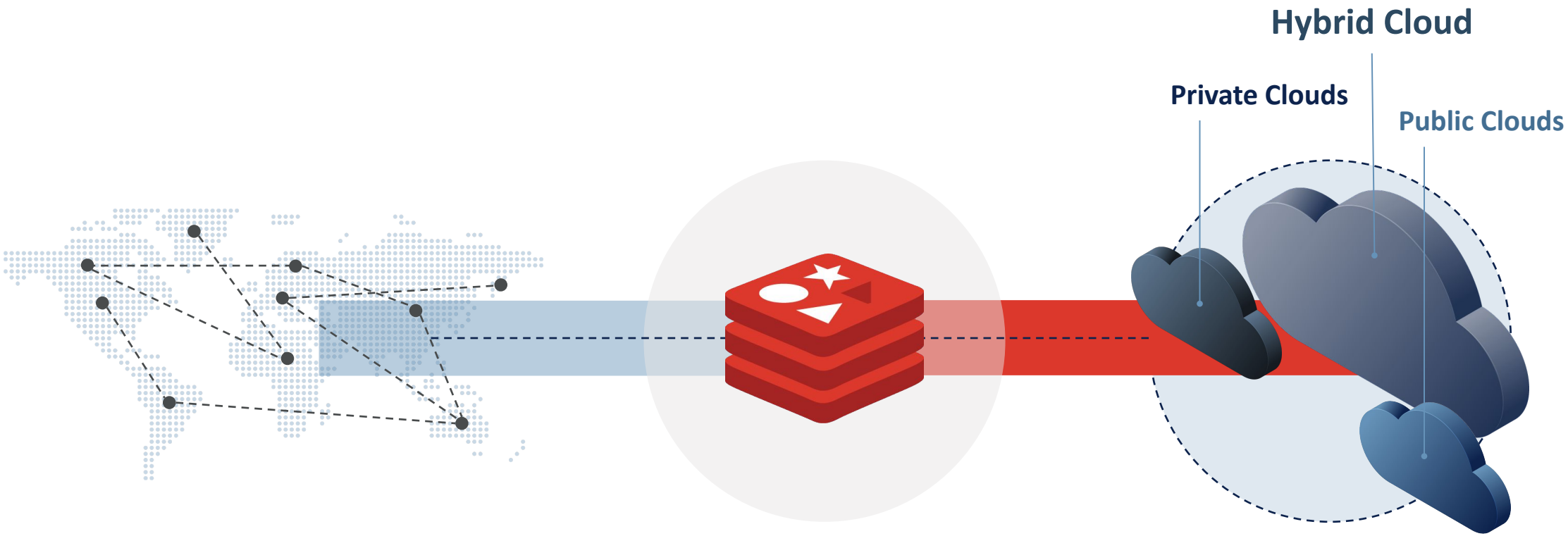


- Sub-millisecond latency under any load
- Five-nines (99.999%) uptime with active-active
- Effective TCO with multi-tenancy and tiered memory access

Redis Enterprise Is Built for Every Stage of Your Journey

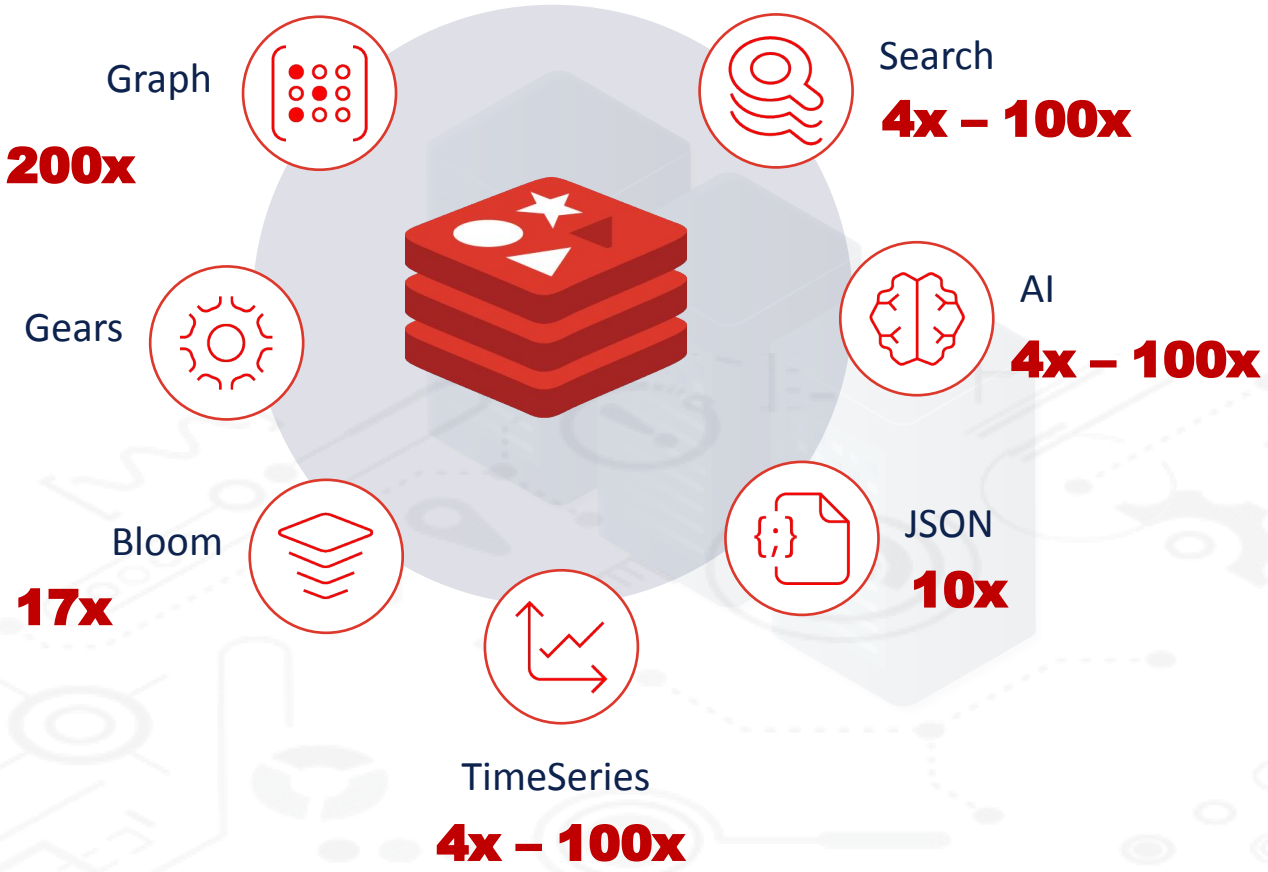
Non Real-Time	Claim Real-Time	Real-time Hybrid and Multi-Cloud Data Layer
redis	redis	redis
>100 ms	10-100 ms	<1 msc
NEEDS CACHE	NEEDS CACHE	BUILT FOR SPEED

Unify Your Data in Real-Time Across The Globe and Clouds



Build with Freedom and Operational Flexibility in the Cloud

Redis Enterprise Cloud



- Search, JSON, Graph, TimeSeries, and Bloom enable new modern use-cases at the Redis speed
- Reducing complexity and improving TCO with a single data platform for all your real-time data processing needs

Redis Enterprise Geo Replication Strategies

Active-Passive

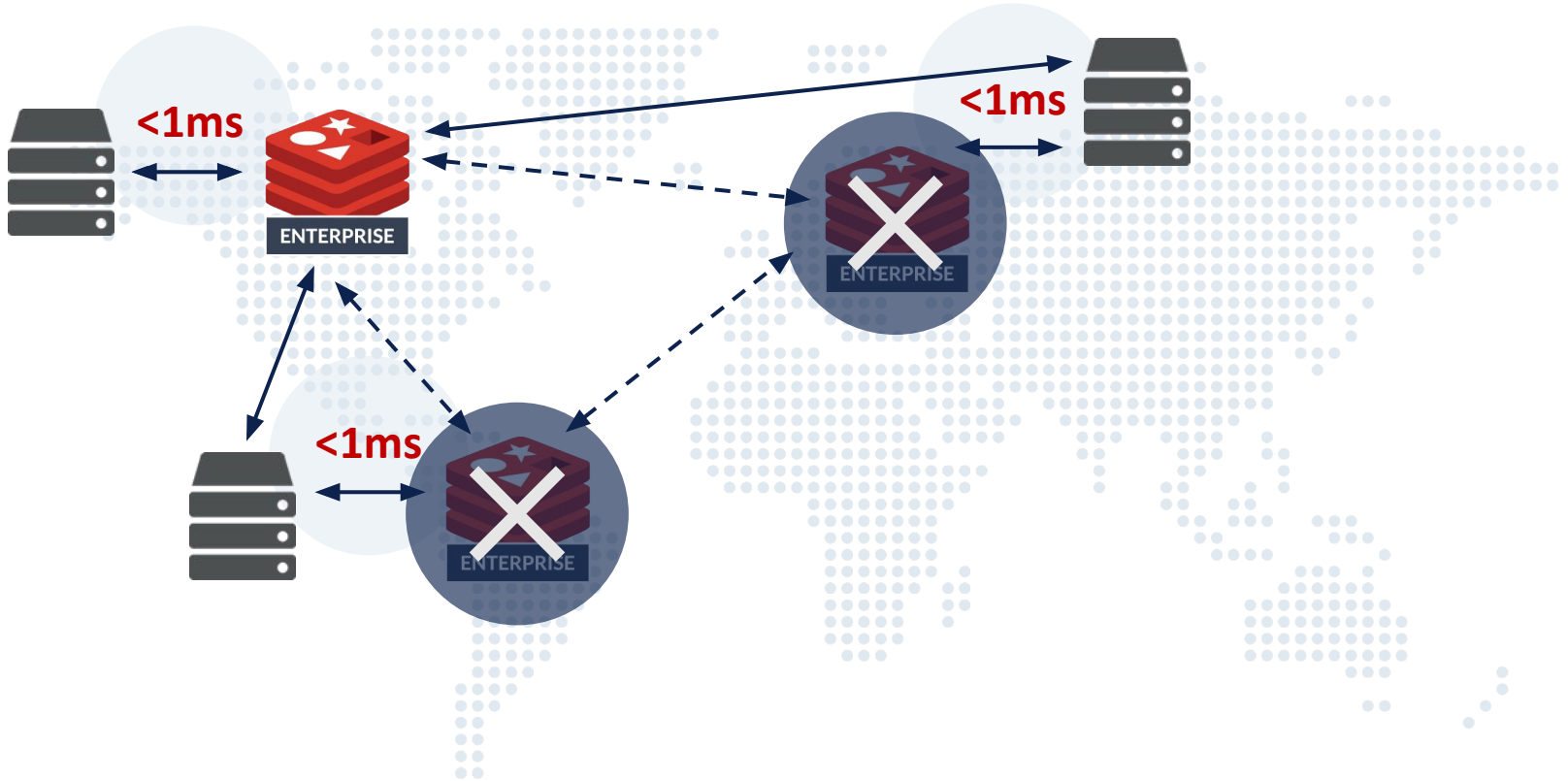
- Improves application **response time**
- Enables highly available standby nodes in **disaster recovery** scenarios
- Provides flexible cluster configuration to **manage cost**

Active-Active

- Delivers **Local** latency across geos
- Provides **Instant** failover without data loss
- **Unifies** your data layer across environments through **Seamless** conflict resolution

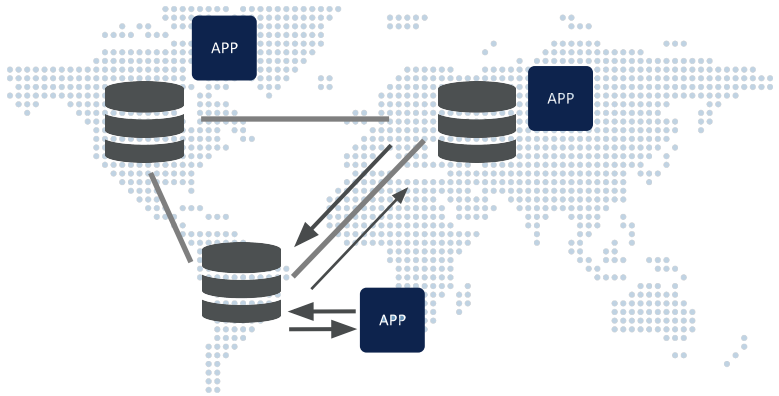
CRDT based Active-Active Delivers Major Service Levels

- Local <1ms latency
- CRDT-based conflict resolution
- Strong resiliency based on consensus-free protocol



Sub-millisecond Performance Delivers Unique Value

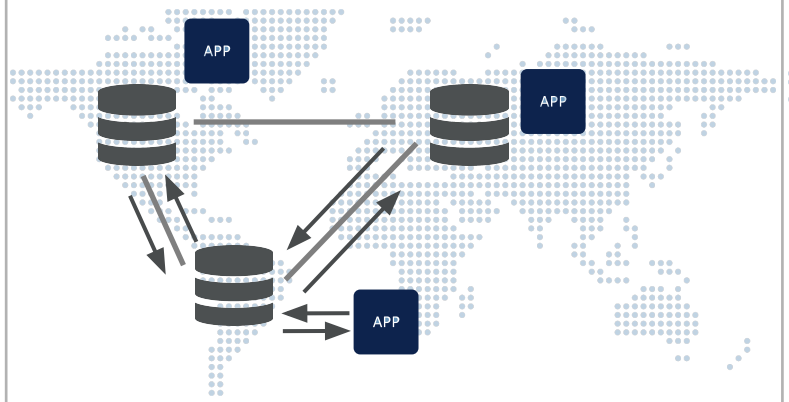
NoSQL DB



Eventual Consistency

100ms

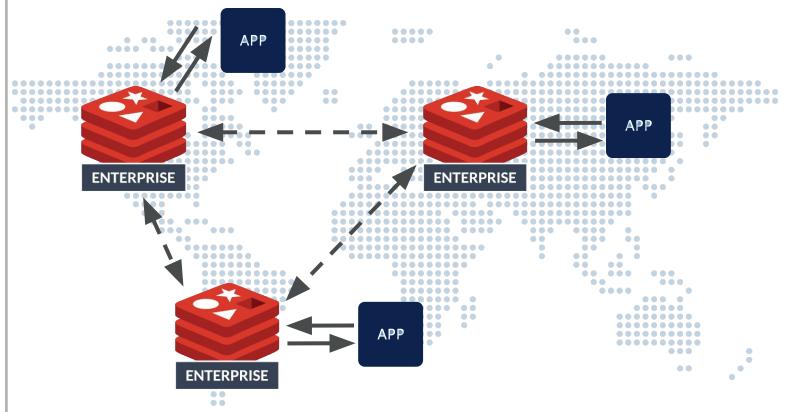
NoSQL DB



Strong Consistency

200ms

Redis CRDTs

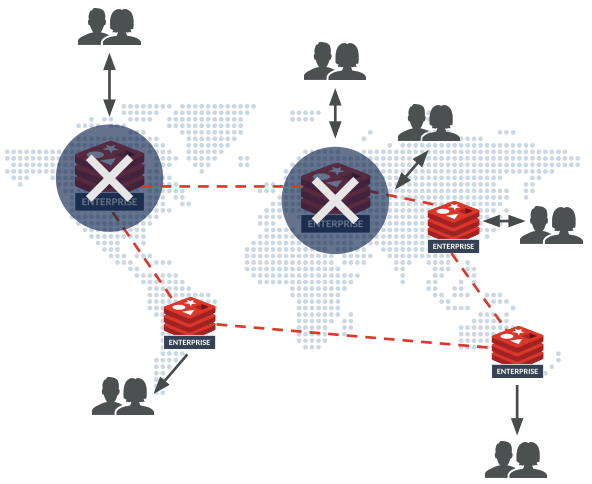


Strong Eventual Consistency, Causal Consistency

<1ms

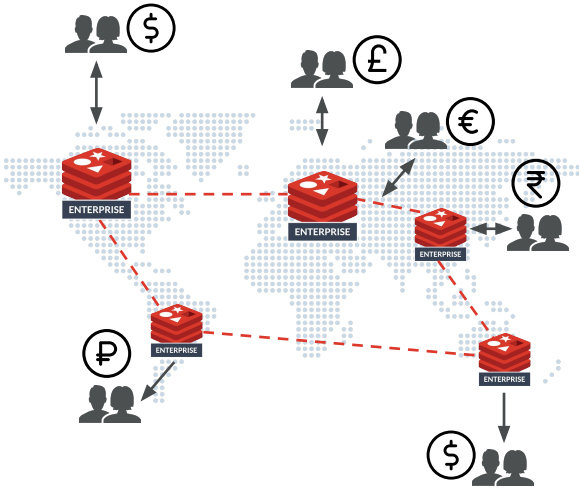
Business Challenges Active-Active Solves

1. Disaster Recovery

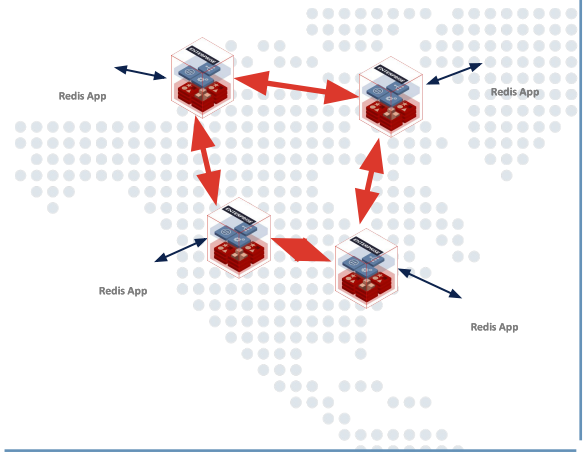


2. Global Session Management

3. Globally Distributed Apps

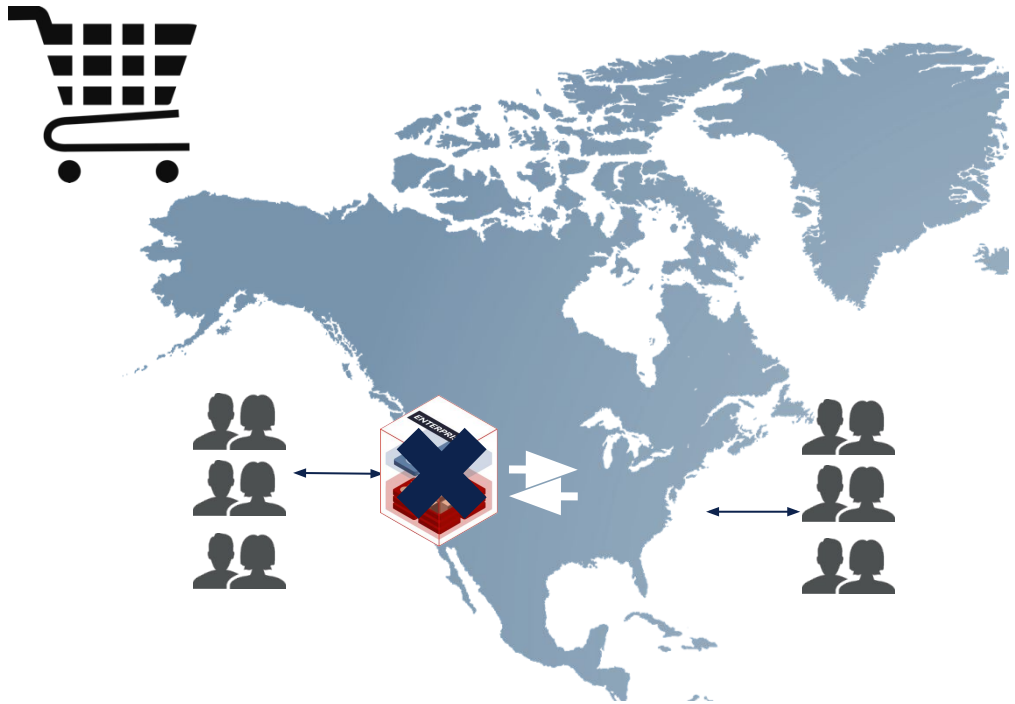


4. Data Consolidation



1. Disaster Recovery

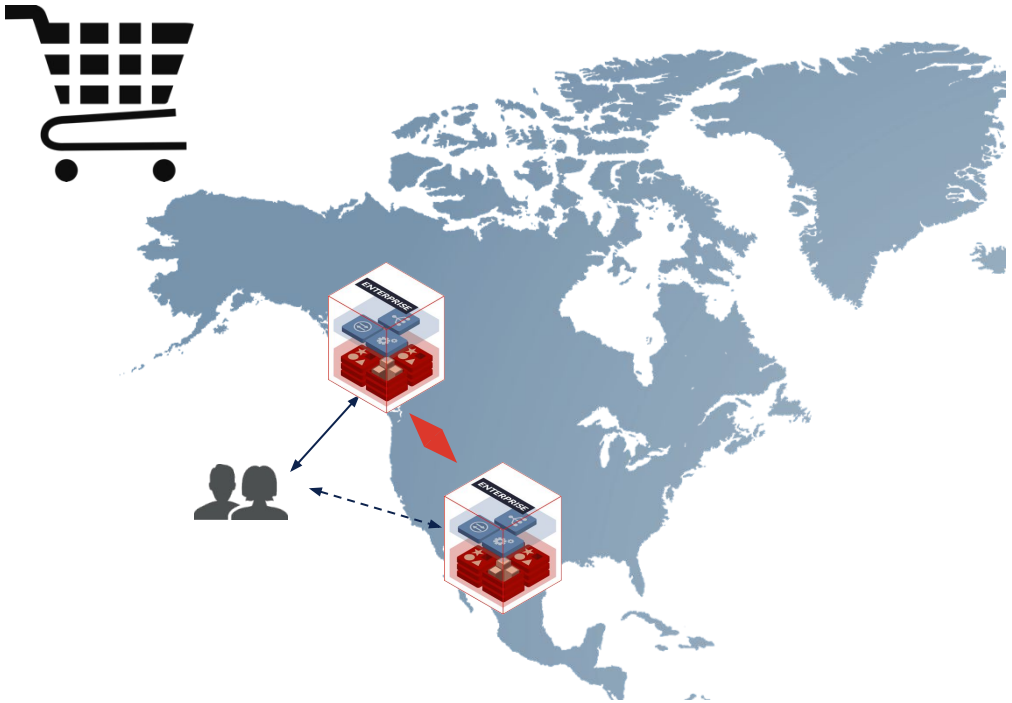
Handling data-center failures



1. Failure in one datacenter, needs sessions to move over to the other data center
2. All the session states (example: items in shopping carts) need to be exactly the same
3. Once restored, any changes need to show up in first datacenter

2. Global Session Management

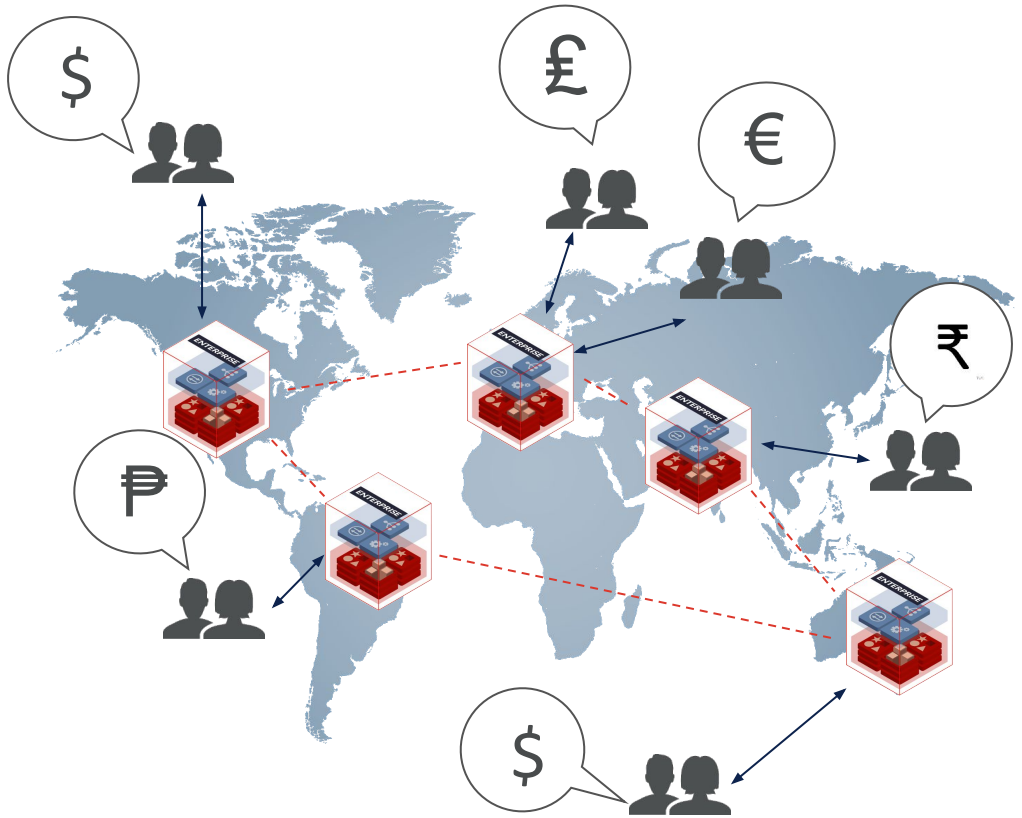
Migrating user sessions across data centers



1. Customers get routed from one datacenter to another on the fly or while moving from one location to another
2. All the session states (example: items in shopping carts) need to be exactly the same
3. If routed back, any changes need to be synced between datacenters

3. Globally Distributed Apps

Delivering local latencies for geographically distributed apps across servers, clouds, and regions

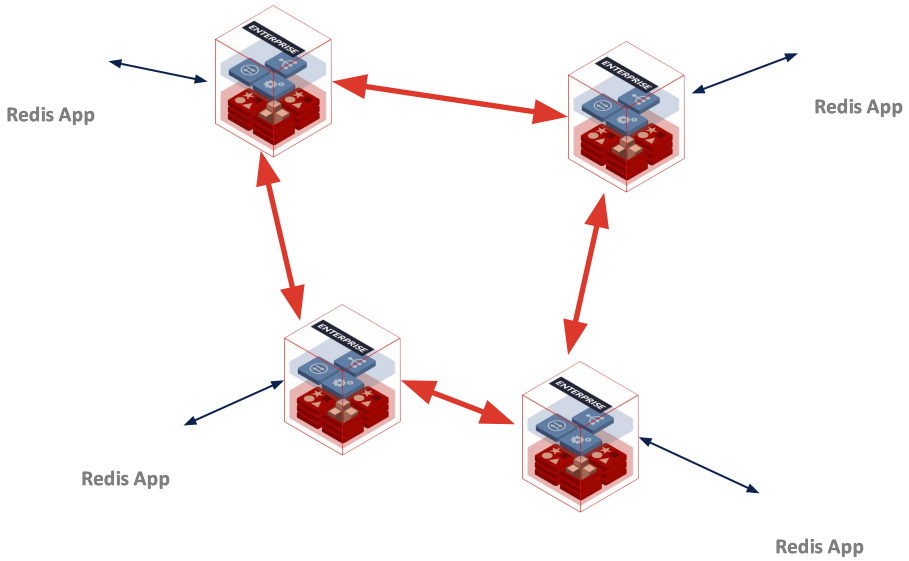


Geo distributed datacenters

1. High frequency writes/reads in many regions
2. Complex data types, not just key-value
3. Database needs to reflect current position
4. Customer needs simple ways to resolve simultaneous updates – delegate to databases

4. Data consolidation

Active Active supports a unified data layer



1. In a microservices environment, apps connect to their own databases
2. However, apps share data structures/tables and require data to be consistent

Active-Active Is Critical for Real-time Use Cases

Fraud Mitigation

Geo Distributed Event Tracking:

Sets Gathering Geo Distributed Events

Geo Distributed Trading/Bidding

Auctions, Bids/Asks:

Lists/Sorted Sets tracking Bids and Asks

Social Engagement Apps

Encoding Social Engagement:

Distributed Counters for “Likes”, “Shares”, “Retweets”

Dashboards & Scoreboards

Tracking Geo Distributed Scoreboards:

Sorted Sets tracking ordered scores

Collaboration Apps

Constructing Smart Timelines, Instant Messaging & Conversation Tracking, distributed streams, and distributed search

Real-time Metering Apps

Tracking Usage/Consumption:

Sets/Lists Tracking Consumption Events

Leading Enterprises Rely on Active-Active to Deliver Business Success

Unify Your Data Layer Across Clouds and Regions

- Consistent datasets for any environment (on-prem, hybrid, or cross-cloud)
- Distribute your data globally

Faster Time to Market

- Simpler to develop geo distributed apps
- Results in high performance, consistent and highly responsive app

Easier to Deploy and Maintain

- Database does all the heavy lifting
- Simpler code means easier to change
- Simpler architectures are easier to implement and manage

Leading Edge

- Based on the latest thinking in computer science (CRDT)
- Complex data types included in conflict resolution

Lab 6: Deploying a 99.999 Redis Environment

Redis Cluster URL: https://tinyurl.com/workshop-cluster
User: admin@admin.com
Password: dryrun

Thank You