The background image is a high-angle aerial photograph of the Boston skyline. The city is built along the Charles River, which curves from the bottom left towards the top center. The river is lined with green parks and several bridges. The city's dense urban landscape is filled with numerous skyscrapers of varying heights, with the financial district cluster visible on the right. In the foreground, there's a mix of lower residential buildings and modern infrastructure, including a highway interchange. The overall scene is a clear, sunny day.

Redis Functions and Data Structures- Today and Beyond

Boston Redis Meetup – June 26, 2018

Agenda

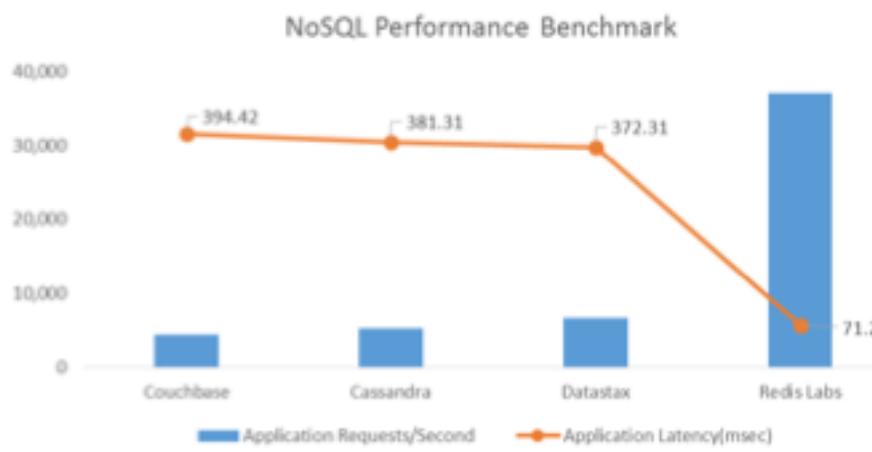
- Redis Key/Value Data Structures
- Interesting Redis Use-cases: Versatility
- More Resources

Also in this deck:

- Why Redis (*not covered in meetup*)
- Who is Redis Labs (*not covered in meetup*)

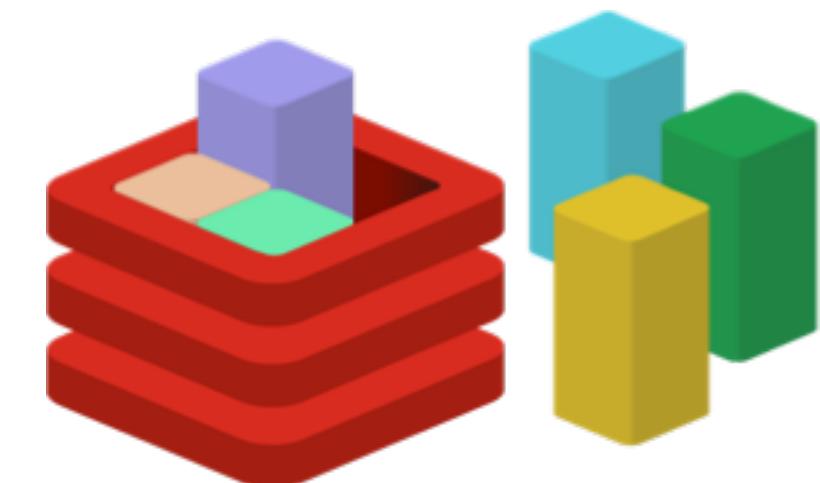
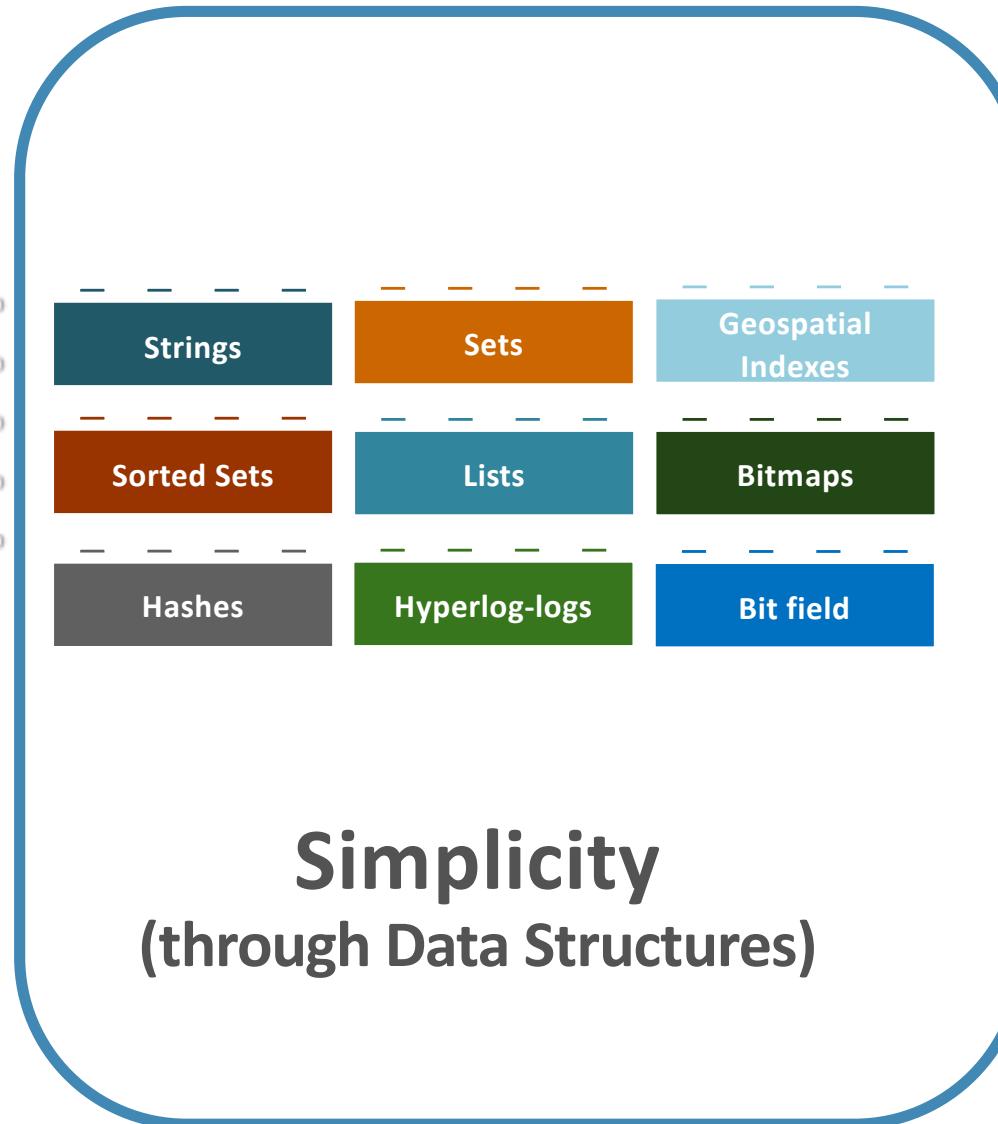
Redis Key/Value Data Structures

Redis is a Game Changer



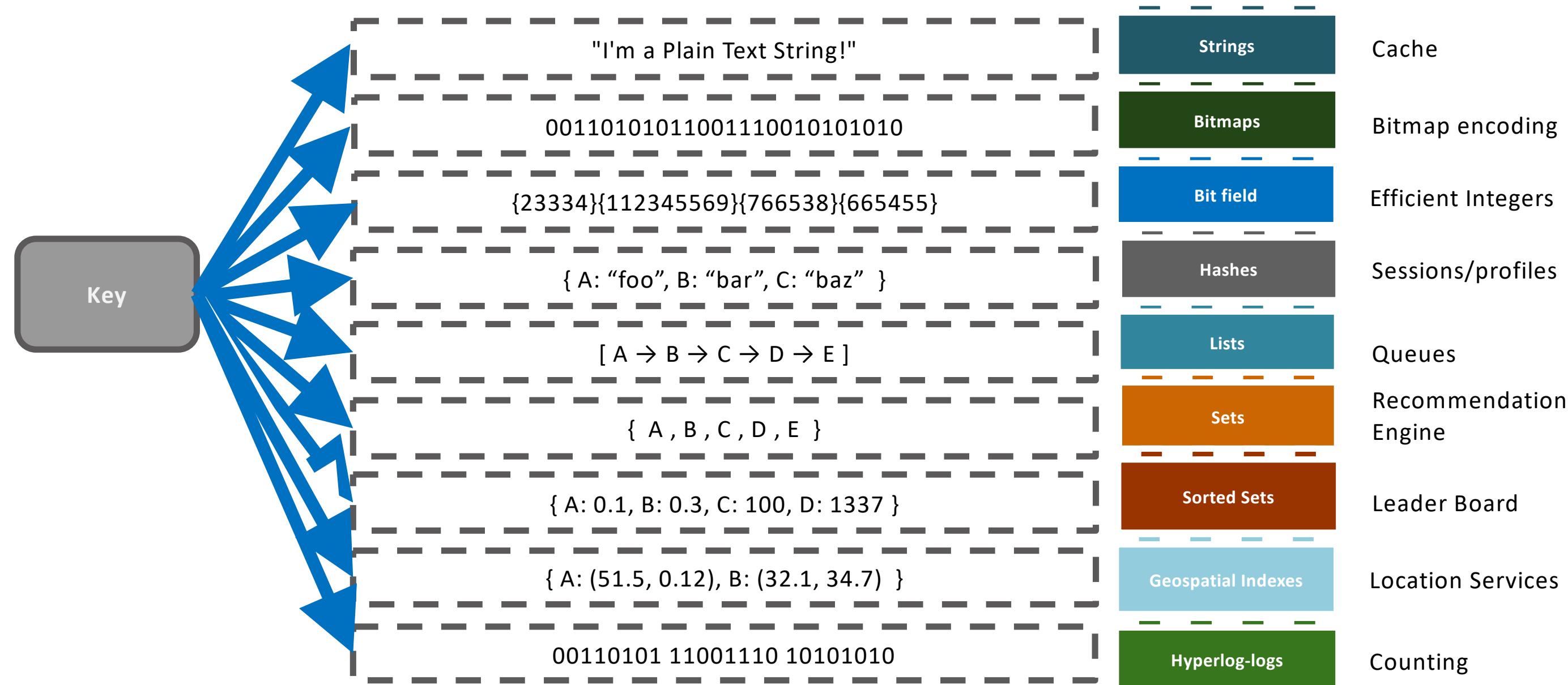
Performance

Simplicity
(through Data Structures)



Extensibility
(through Redis Modules)

Data Structure Store: Lego For Your App



What Can You Do With Redis?

Use as in-memory database, cache or message broker

Common Uses

- Cache
- Message Brokers/Queues
- User Sessions
- Real-time Recommendation Engine
- Leaderboards
- ...More.....much, MUCH more!!!

Simple Cache

The Problem

- Multiple database calls create impossibly slow web page response times

Why Redis Rocks

- **Strings** are perfect for this!
- **SET** lets you save session variables as key/value pairs
- **GET** to retrieve values

Redis Strings for Simple Cache

- Strings store text, which might be made from the results of multiple database queries and HTML
- Can have expiry
- You can register to listen for changes on keys and operations
- Multiple eviction policies supported

```
$ SET userid:1 "8754"  
$ GET userid:1  
$ EXPIRE userid:1 60  
$ DEL userid:1
```

```
jedis.set("userid:1", "8754");  
jedis.get("userid:1");  
jedis.expire("userid:1", 60);  
jedis.del("userid:1");
```

User Sessions

The Problem

- Maintain session state across multiple servers
- Multiple session variables
- High speed/low latency required

Why Redis Rocks

- **Hashes** are perfect for this!
- **HMSET** lets you save session variables as key/value pairs
- **HMGET** to retrieve values
- **HINCRBY** to increment any field within the hash structure
- **HDEL** to delete one field/value

Redis Hashes for User Sessions

hash key: usersession:1

userid	8754
name	dave
ip	10:20:104:31
hits	1
lastpage	home

```
$ HMSET usersession:1 userid 8754 name dave ip 10:20:104:31 hits 1  
$ HMGET usersession:1 userid name ip hits  
$ HINCRBY usersession:1 hits 1
```

```
$ HSET usersession:1 lastpage "home"  
$ HGET usersession:1 lastpage  
$ HDEL usersession:1 lastpage
```

```
$ DEL usersession:1
```

Hashes store a mapping of keys to values – like a dictionary or associative array – but faster

Redis Hashes for User Sessions

```
Map<String, String> userSession = new HashMap<>();
userSession.put("userid", "8754");
userSession.put("name", "dave");
userSession.put("ip", "10:20:104:31");
userSession.put("hits", "1")
jedis.hmset("usersession:1", userSession);

jedis.hmget("usersession:1", "userid", "name", "ip", "hits");
jedis.hincrBy("usersession:1", "hits", 1);
jedis.hset("usersession:1", "lastpage", "home");
jedis.hget("usersession:1", "lastpage");
jedis.hdel("usersession:1", "lastpage");
```

Managing Queues of Work

The Problem

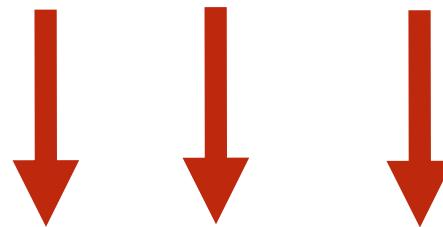
- Tasks need to be worked on asynch to reduce block/wait times
- Lots of items to be worked on
- Assign items to worker process and remove from queue at the same time
- Similar to buffering high speed data-ingestion

Why Redis Rocks

- **Lists** are perfect for this!
- **LPUSH, RPUSH** add values at beginning or end of queue
- **RPOPLPUSH** – pops an item from one queue and pushes it to another queue

Redis Lists for Managing Queues

LPUSH adds values to head of list



blue	green	orange	red
------	-------	--------	----	----	-----

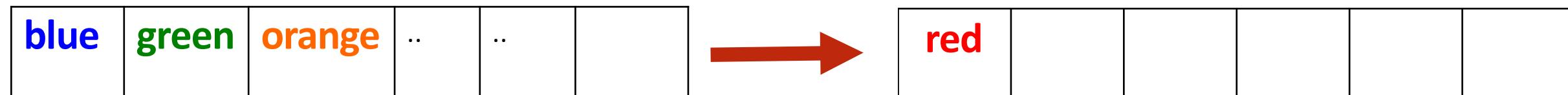


RPUSH adds value to tail of list

```
$ LPUSH queue1 orange  
$ LPUSH queue1 green  
$ LPUSH queue1 blue  
$ RPUSH queue1 red
```

Redis Lists for Managing Queues

```
$ LPUSH queue1 orange  
$ LPUSH queue1 green  
$ LPUSH queue1 blue  
$ RPUSH queue1 red
```



```
$ RPOPLPUSH queue1 queue2
```

RPOPLPUSH pops a value from one list and pushes it to another list

Redis Lists for Managing Queues

```
jedis.lpush("queue1", "orange");
jedis.lpush("queue1", "green");
jedis.lpush("queue1", "blue");
jedis.rpush("queue1", "red")

jedis.rpoplpush("queue1", "queue2");
```

Real-time Recommendation Engine

The Problem

- People who read this article also read these other articles
- Want real time not data mining

Also used for:

- Recommending Similar Purchases
- Identifying Fraud

Why Redis Rocks

- **SETS** are unique collections of strings
- **SADD** to add tags to each article
- **SISMEMBER** to check if an article has a given tag
- **SMEMBERS** to get all the tags for an article
- use **SINTER** to find similar articles tagged with the same tags

Redis Sets for Recommendations

Set: tag:1

article 1	article 3		
-----------	-----------	------	--	--

Set: tag:2

article 3	article 14	Article 22	..	
-----------	------------	------------	----	--

Set: tag:3

article 2	article 3	article 9	..	
-----------	-----------	-----------	----	--

Add values (articles) to Sets (tags)

```
$ SADD tag:1 article:3 article:1  
$ SADD tag:2 article:22 article:14 article:3  
$ SADD tag:3 article:9 article:3 article:2
```

Confirm the values have been added

```
$ SMEMBERS tag:3      (also tag:1 & tag:2)
```

- 1) "article:3"
- 2) "article:2"
- 3) "article:9"

Find values that exist in all three Sets

```
$ SINTER tag:1 tag:2 tag:3
```

- 1) "article:3"

Redis Sets for Recommendations

```
jedis.sadd("tag:1", "article:3", "article:1");
jedis.sadd("tag:2", "article:22", "article:14", "article:3");
jedis.sadd("tag:3", "article:9", "article:3", "article:2");

jedis.smembers("tag:1")
jedis.sinter("tag:1", "tag:2", "tag:3");
```

Example : Redis For Bid Management

The Application Problem

- Many users bidding on items
- Need to instantly show who's leading, in what order and by how much
- May also need to display analytics like how many users are bidding in what range
- Disk-based DBMS-es are too slow for real-time, high scale calculations

Why Redis Rocks This

- Sorted sets automatically keep list of users and scores updated and in order (ZADD)
- ZRANGE, ZREVRANGE will get your top users
- ZRANK will get any user's rank instantaneously
- ZCOUNT will return a count of users in a range
- ZRANGEBYSCORE will return all the users in a range by their bids

Redis Sorted Sets

Item: 1

id:3		44000
------	--	-------

id:4		35000
------	--	-------

--	--	--

id:1		21000
------	--	-------

id:2		10000
------	--	-------

--	--	--

ZADD item:1 10000 id:2 21000 id: 1

ZADD item:1 34000 id:3 35000 id 4

ZINCRBY item:1 10000 id:3

ZREVRANGE item:1 0 0

id:3

```
jedis.zadd("item:1", 10000, "id:2");
jedis.zadd("item:1", 21000, "id:1");
jedis.zadd("item:1", 34000, "id:3");
jedis.zadd("item:1", 35000, "id:4");
jedis.zincrby("item:1", 10000, "id:3");
```

```
jedis.zrevrange("item:1", 0,0);
```

Sorted Sets for Leaderboards

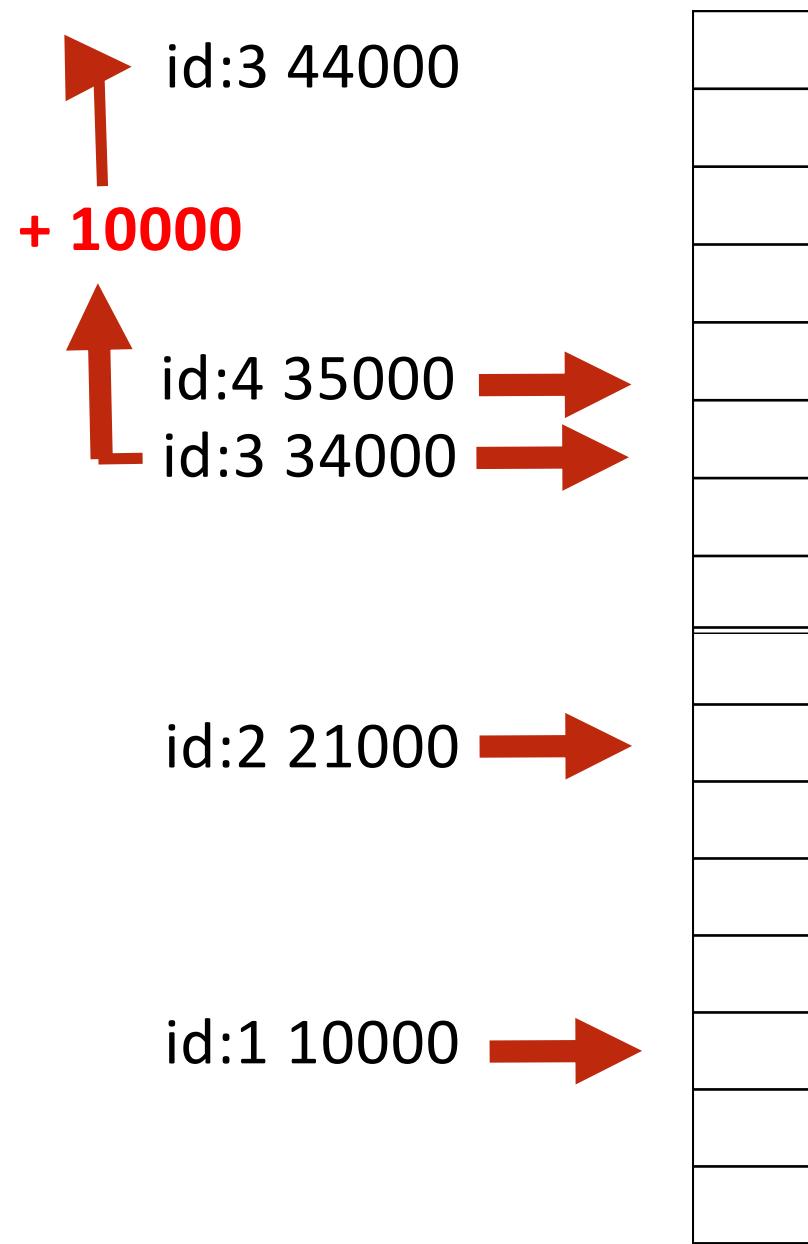
The Problem

- MANY users playing a game or collecting points
- Display real-time leaderboard.
- Who is your nearest competition
- Disk-based DB is too slow

Why Redis Rocks

- **Sorted Sets** are perfect!
- Automatically keeps list of users sorted by score
- **ZADD** to add/update
- **ZRANGE**, **ZREVRANGE** to get user
- **ZRANK** will get any users rank instantaneously

Redis Sorted Sets



```
$ ZADD game:1 10000 id:1  
$ ZADD game:1 21000 id:2  
$ ZADD game:1 34000 id:3  
$ ZADD game:1 35000 id:4  
$ ZINCRBY game:1 10000 id:3
```

Get the Leader Board

```
$ ZREVRANGE game:1 0 0  
$ ZREVRANGE game:1 0 1 WITHSCORES
```

Redis Sorted Sets

```
jedis.zadd("game:1", 10000, "id:1");
jedis.zadd("game:1", 21000, "id:2");
jedis.zadd("game:1", 34000, "id:3");
jedis.zadd("game:1", 35000, "id:4");

jedis.zincrby("game:1", 10000,"id:3");

jedis.zrevrange("game:1", 0,0);
jedis.zrevrangeWithScores("game:1", 0,1);
```

Search by Location

The Problem

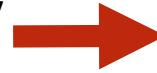
- Give me all the pharmacies in 2 km radius
- How far am I from the hospital

Why Redis Rocks

- **GeoSet** is perfect!
- Stores location as Geohash
- **GEOADD** to add a location
- **GEODIST** to get distance
- **GEORADIUS** to get locations in radius

Search By Location

St Margerets Pharmacy
2163543909330618



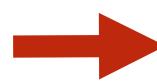
GEOADD pharmacies -0.310392 51.456454 "Charles Harry Pharmacy"
GEOADD pharmacies -0.296402 51.462069 "Richmond Pharmacy"
GEOADD pharmacies -0.318604 51.455338 "St Margerets Pharmacy"

Charles Harry Pharmacy
2163543917444056



GEORADIUS pharmacies -0.30566239999996014 51.452921 600 m
WITHDIST WITHCOORD ASC
1) 1) "Charles Harry Pharmacy"
2) "511.6979"
3) 1) "-0.31039327383041382"
2) "51.45645288459863309"

Richmond Pharmacy
2163544020748440



Search By Location

```
jedis.geoadd("pharmacies", -0.310392, 51.456454, "Charles Harry  
Pharmacy");  
jedis.geoadd("pharmacies", -0.296402, 51.462069, "Richmond  
Pharmacy");  
jedis.geoadd("pharmacies", -0.318604, 51.455338, "St Margerets  
Pharmacy");  
  
jedis.georadius("pharmacies", -0.30566239999996014, 51.452921,  
600 , GeoUnit.M ,  
GeoRadiusParam.geoRadiusParam().withCoord().withCoord().withDist  
( ));
```

Count Unique Visitors

The Problem

- Count unique daily visitors to the site
- How many unique users have clicked on an ad

Why Redis Rocks

- **HyperLogLog** is perfect!
- Keeps Count of each unique element
- **PFADD** to add an element
- **PFCOUNT** to get count

HyperLogLog to Count Unique Visitors

"HYLL\x01\x00\x00\x00\x04\x00\x00\x00\x00\x00\x00\x00\x00E\xa0\x80S1\x84E2\x88\\|\x17\x80E\xdd"

PFADD visitors:20160921 86.163.34.208
PFADD visitors:20160921 52.203.210.236
PFADD visitors:20160921 54.87.203.132
PFADD visitors:20160921 54.87.201.121
PFADD visitors:20160921 52.203.210.236

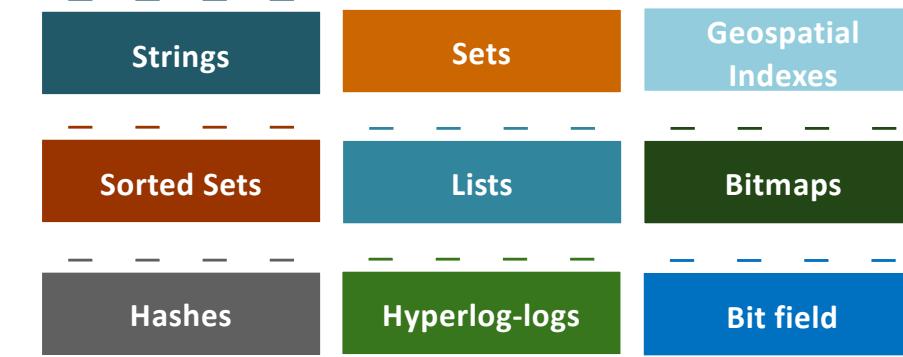
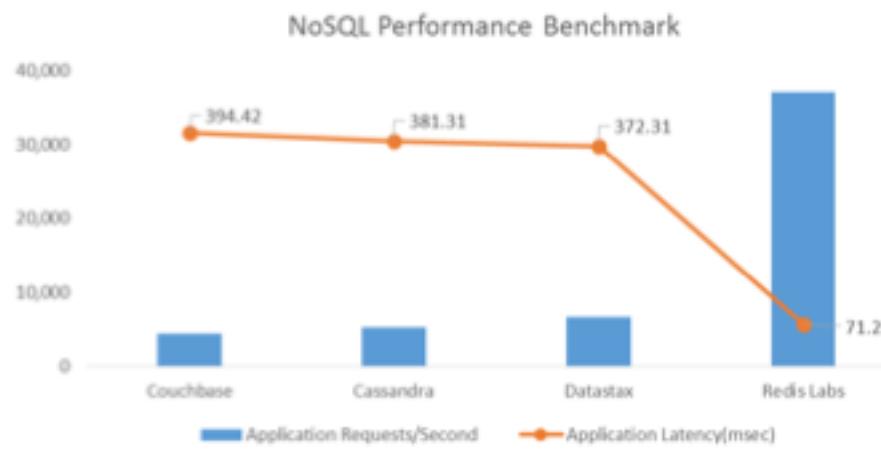
PFCOUNT visitors:20160921
(integer) 4

HyperLogLog to Count Unique Visitors

```
jedis.pfadd("visitors:20160921", "86.163.34.208");
jedis.pfadd("visitors:20160921", "52.203.210.236");
jedis.pfadd("visitors:20160921", "54.87.203.132");
jedis.pfadd("visitors:20160921", "54.87.201.121");
jedis.pfadd("visitors:20160921", "52.203.210.236");

jedis.pfadd("visitors:20160921");
```

Redis is a Game Changer



Performance

Simplicity
(through Data Structures)

Extensibility
(through Redis Modules)

Secondary Index?

SQL?

Full Text Search?

Machine Learning?

But Can Redis Do X?

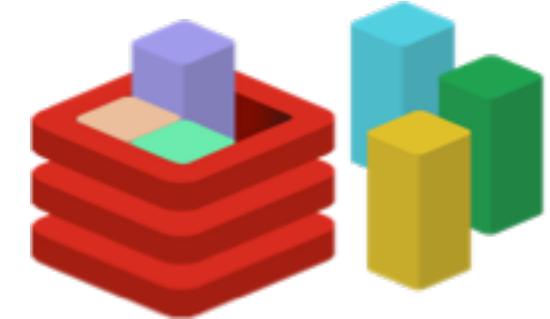
AutoComplete?

Graph?

Time Series?

Redis Modules

- C extensions to Redis
- Work at native speed
- Add commands
- Can implement their own native data types
- Can extend redis or just use it as a “Server Framework”
 - Isolated API
 - ABI Backwards Compatibility



Adapt your database to your data

Neural Redis

Simple Neural Network Native
to Redis

Redis-ML

Machine Learning Model
Serving

RediSearch

Full Text Search Engine in Redis

ReJSON

JSON Engine on Redis.
Pre-released

Time Series

Time series values aggregation
in Redis

Graph

Graph database on Redis based
on Cypher language

Rate Limiter

Based on Generic Cell Rate
Algorithm (GCRA)

Crypto Engine Wrpper

Secure way to store data in
Redis via encrypt/decrypt with
various [Themis](#) primitives

Secondary Index/RQL

Indexing + SQL -like syntax for
querying indexes.
Pre-released

Redis Use-cases: Redis Versatility

Redis Use-cases: Redis Versatility

- Hydra Microservices framework
- Polyglot / Multi-model Persistence Architecture
- Where to learn more: Redis University

Hydra Microservices Framework

FlyWheel Sports



(Light)
Case Study!

Hydra

Lightweight NodeJS Microservices Framework

Hydra provides service discovery, distributed messaging, load balancing, queuing, routing, presence, logging, configuration management and health monitoring – all leveraging Redis under the covers

Target Environments: from IoT to laptops to scalable container-based cloud infrastructure

Example: Application Routes

```
127.0.0.1:6379[15]> keys *routes*
1) "hydra:service:imageproc-svcs:service:routes"
2) "hydra:service:payment-svcs:service:routes"
3) "hydra:service:hydra-router:service:routes"
4) "hydra:service:user-svcs:service:routes"
5) "hydra:service:notification-svcs:service:routes"
```

More at <https://speakersdeck.com/cjus/building-light-weight-microservices-using-redis>

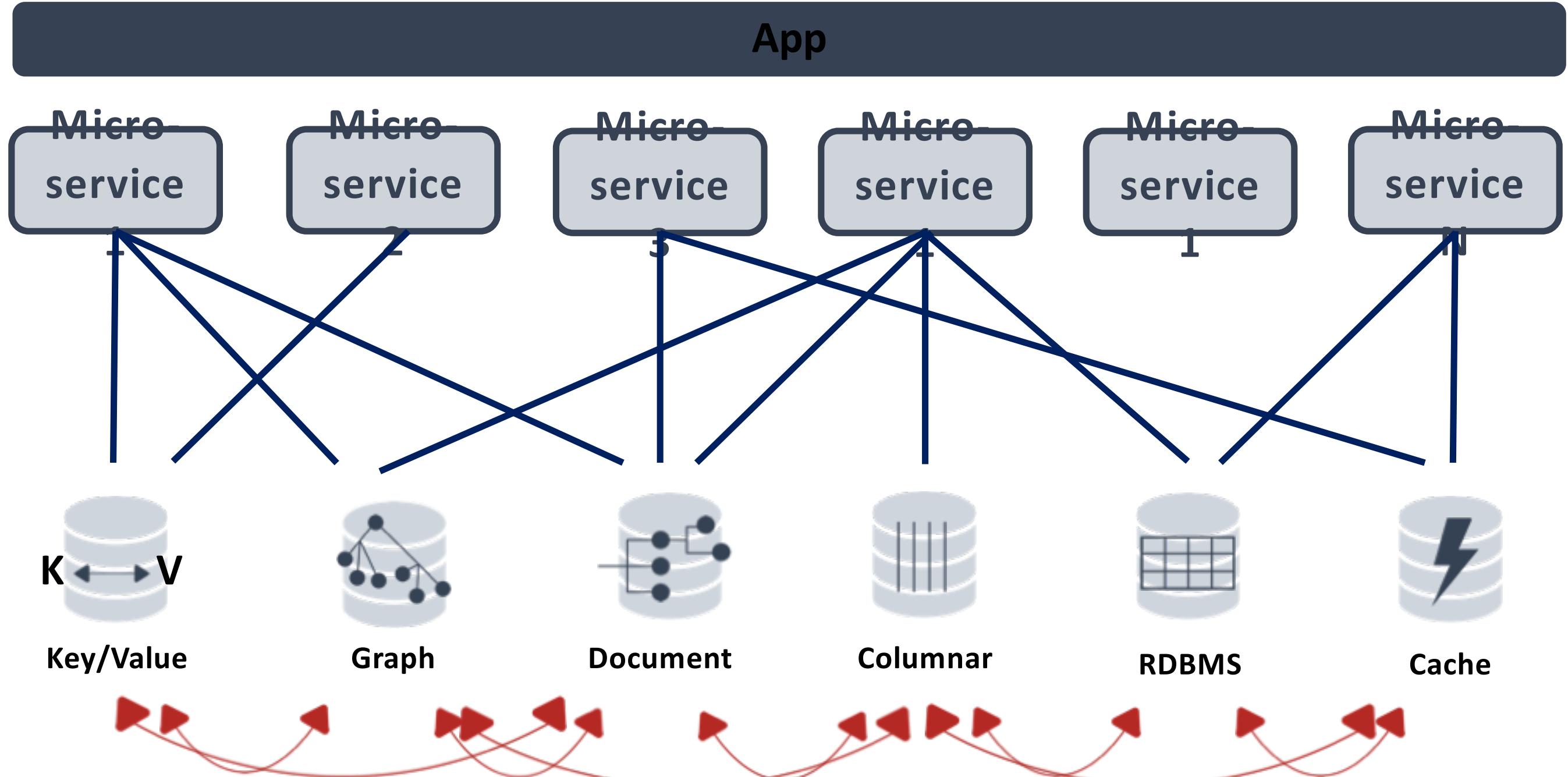
Result: Highly available,
portable, light-weight
microservices framework

```
127.0.0.1:6379[15]> type hydra:service:asset-svcs:service:routes
set
127.0.0.1:6379[15]> smembers hydra:service:asset-svcs:service:routes
1) "[post]/v1/asset/create-thumbnails"
2) "[put]/v1/asset/:aid"
3) "[get]/v1/asset/assets"
4) "[get]/v1/asset/health"
5) "[get]/asset-svcs/"
```

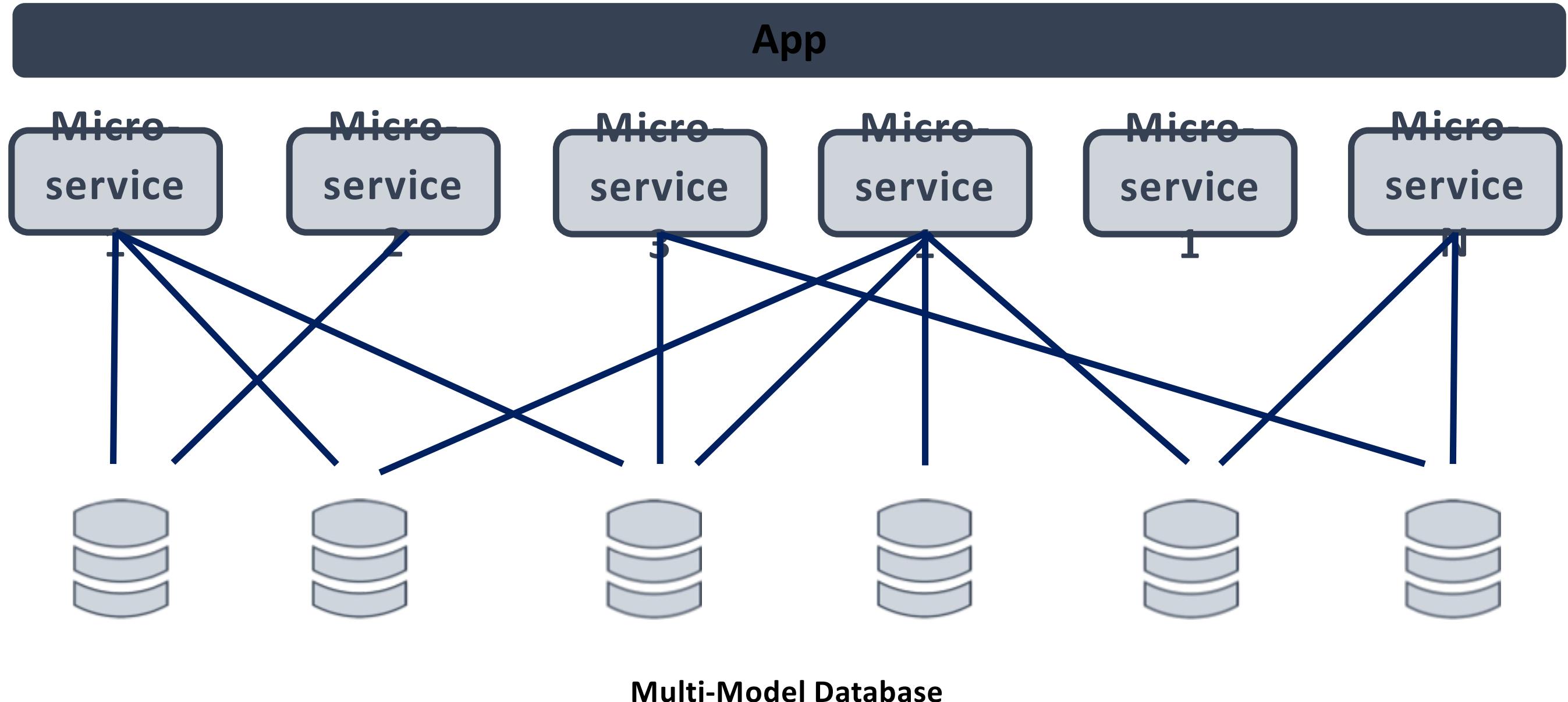
App Arch Simplification Concept and Example

Twitter Firehose app

Polygot Persistence Architecture



Multi-Model Database Architecture



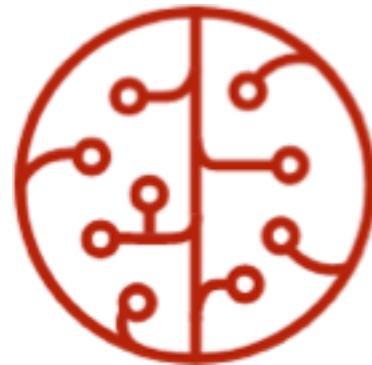
Redis' Unique Multi-Model Implementation



Rate Limiter



Rebloom



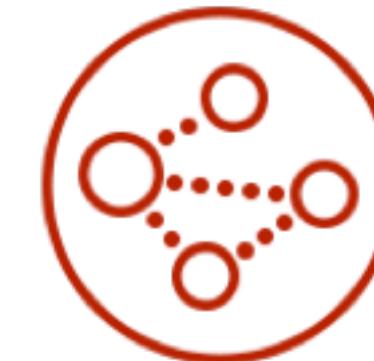
Redis-ML



Redisearch



Database, Cache,
Message Broker, Streams



RedisGraph



ReJSON

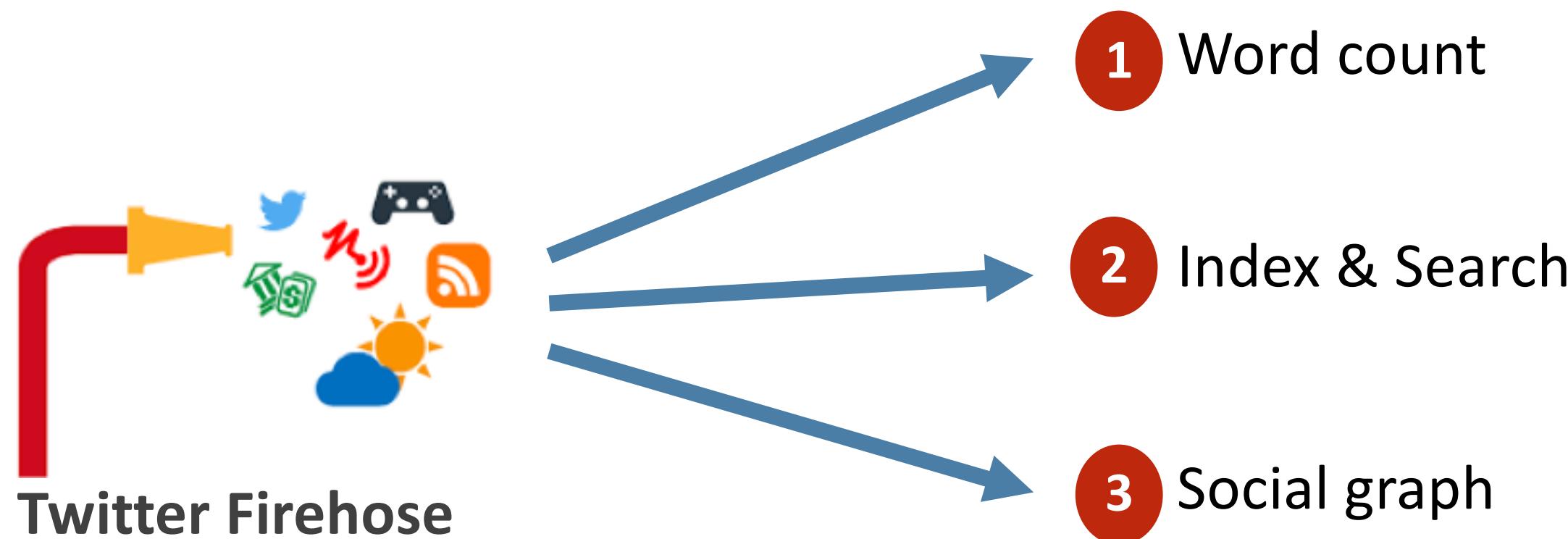


Custom

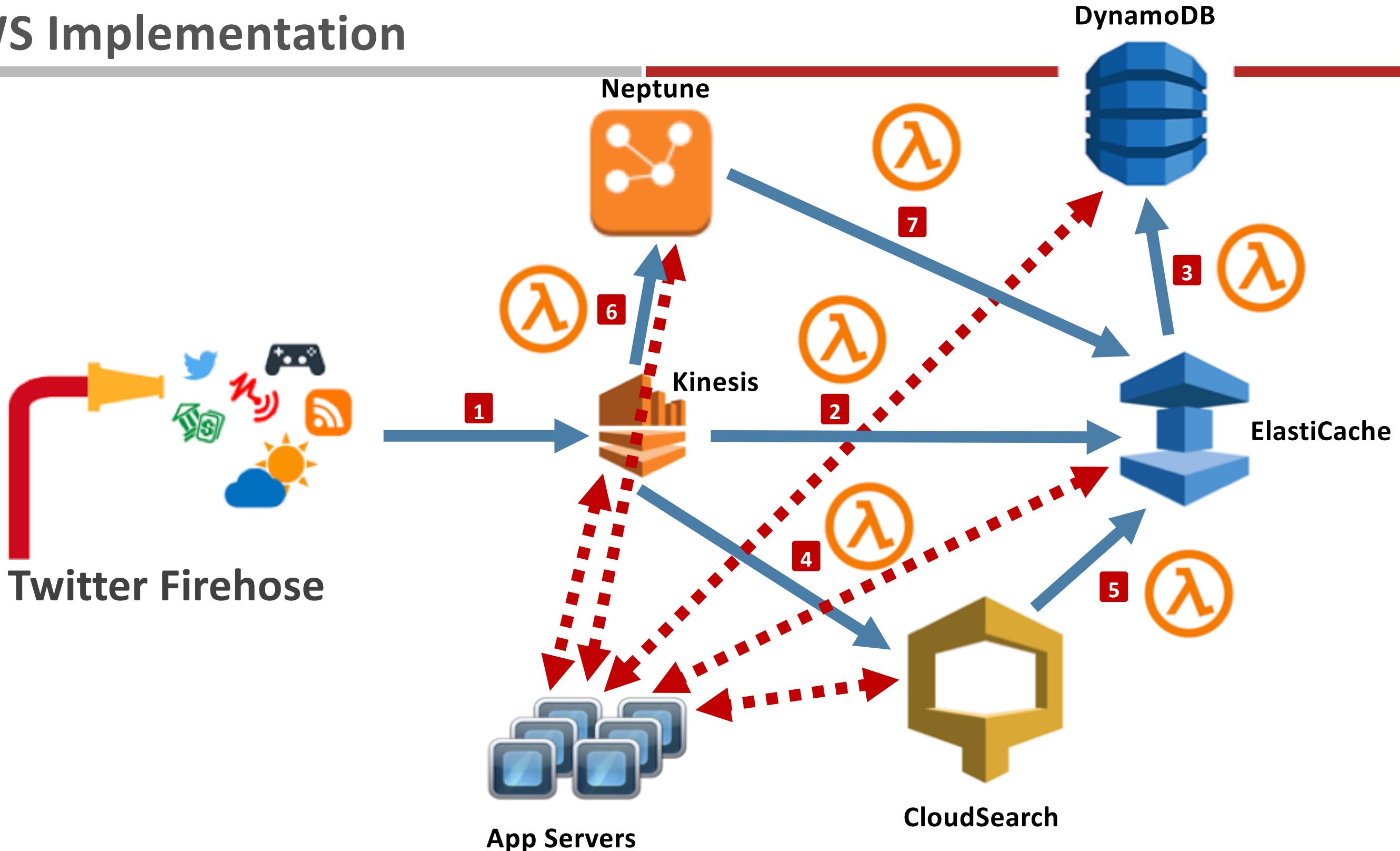


Redis-Timeseries

Build an App With the Following Functionalities:

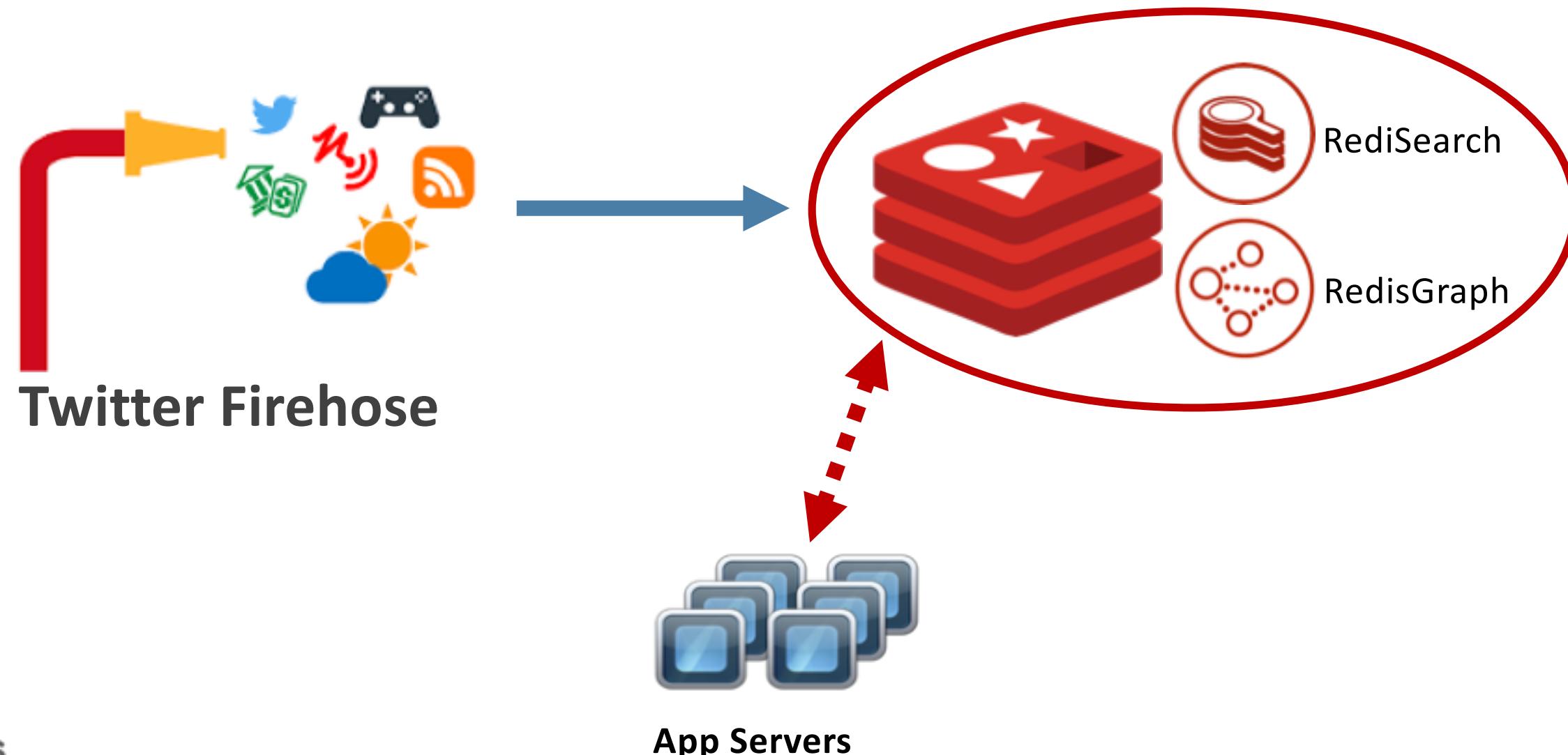


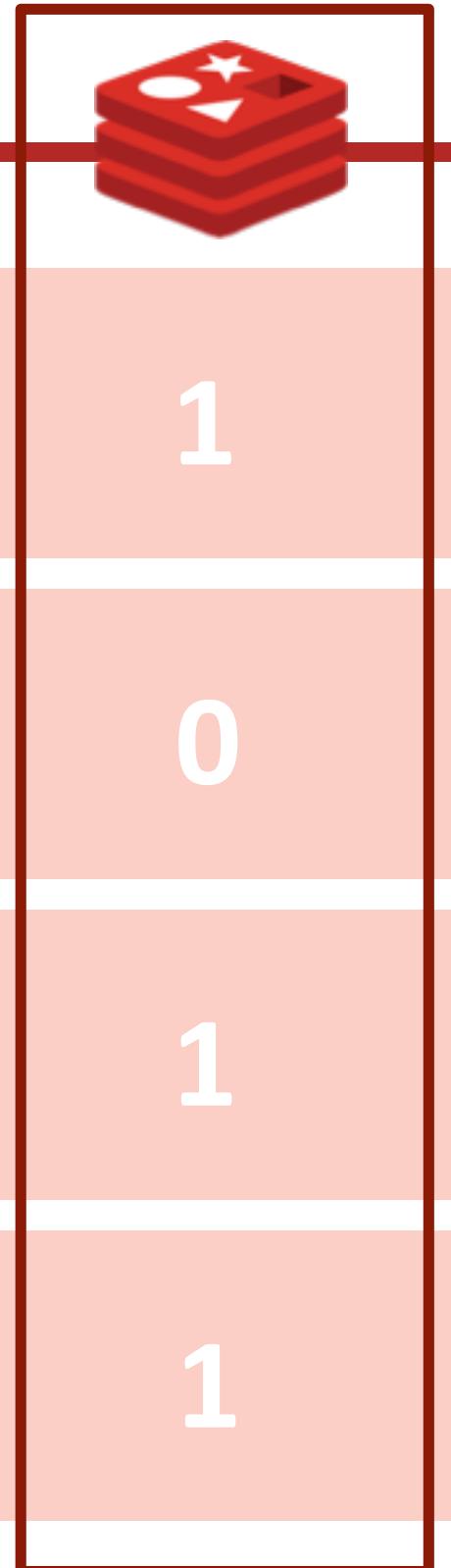
AWS Implementation





Redis Implementation





Data services

5

1

Lambda functions

6

0

Application connectors

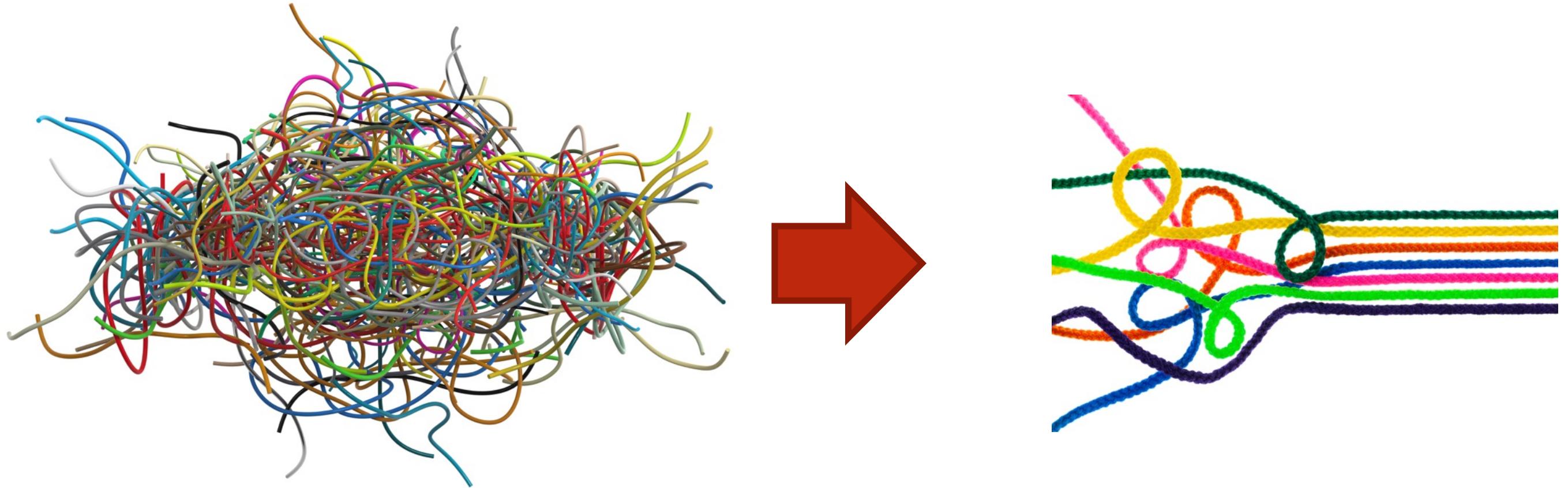
5

1

Network hops

7

1



Redis untangles architecture complexity

Ok, where can I see it in action?

<https://www.youtube.com/watch?v=SGEG5q4LT9s>

<http://bit.ly/2Km22Gx>

<https://github.com/stockholmux/redisconf2018-keynote-multi-model-demo>

How Can I Learn More?

RedisU, RedisConf, Redis In Action

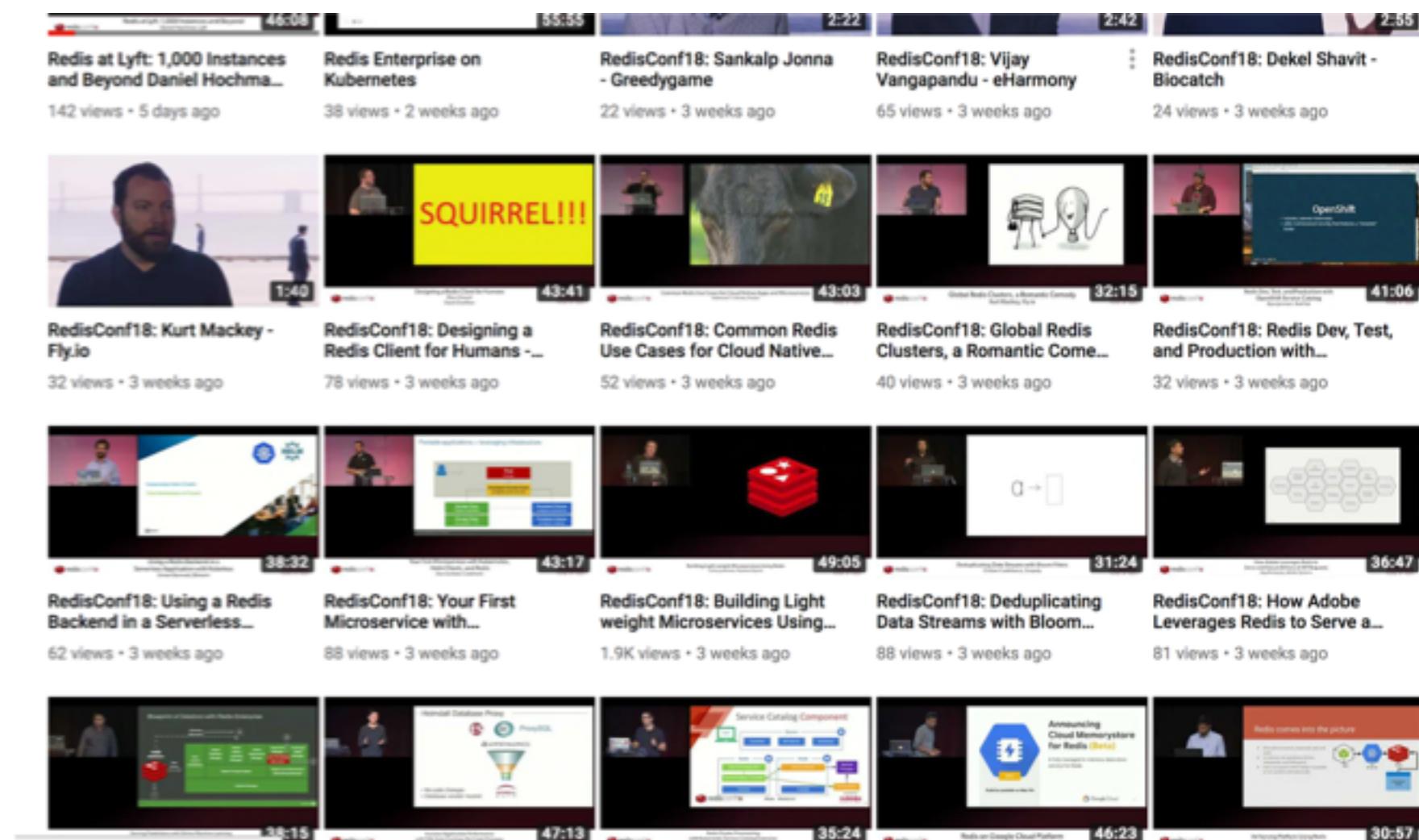
Launching Redis University



- Free, Online, classroom style education for global Redis community
- Learn from the best - Taught by Redis engineers and practitioners
- Initial Course: RU101 - An Introduction to Redis Data Structures
 - Enrollment opens at RedisConf 18, starts June 2018
 - Duration of 6 weeks
 - Certification of completion if achieve 65% or more
- Further course being developed and released this year
- Enroll at university.redislabs.com

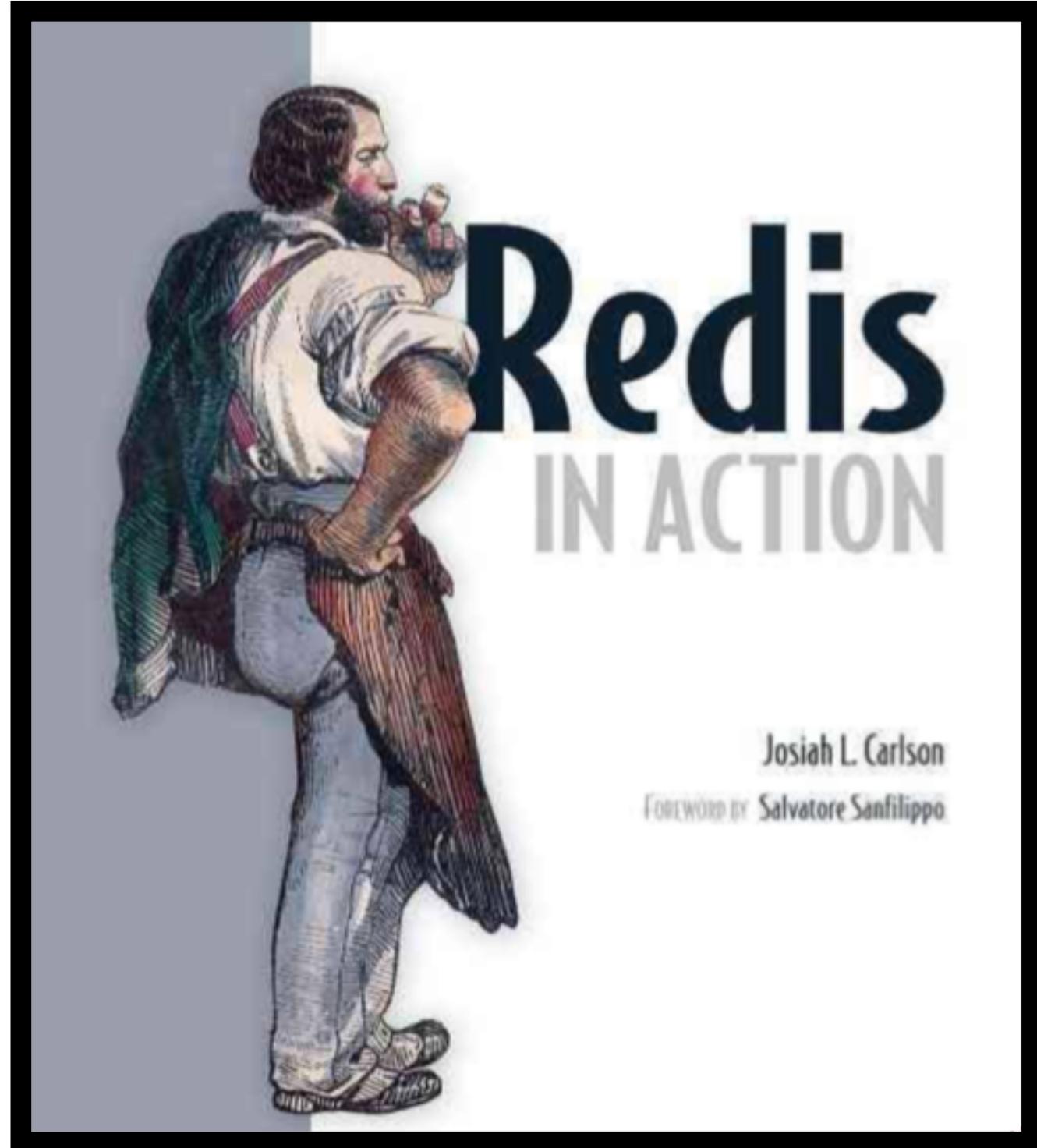
RedisConf18

- You don't *have* to go to in person...
- <http://bit.ly/2txUrOZ>



Redis in Action

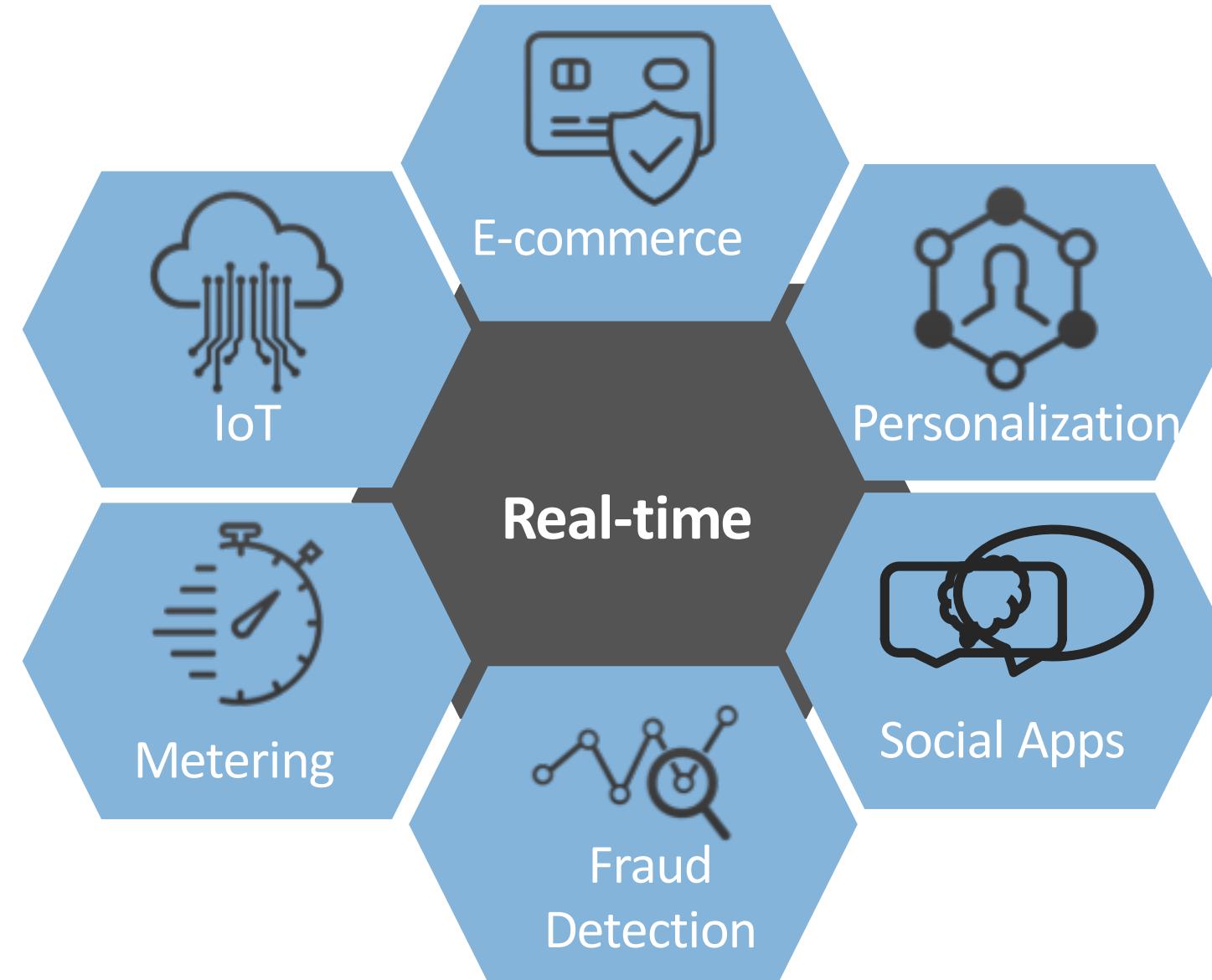
- Comprehensive book describing use-cases and implementing them!
- <http://bit.ly/2ItstlT>



**Remaining content not covered in
meetup**

Why Redis

Redis Powers a Range of Solutions



...AND MANY MORE

Redis Uniquely Suited To Modern Apps

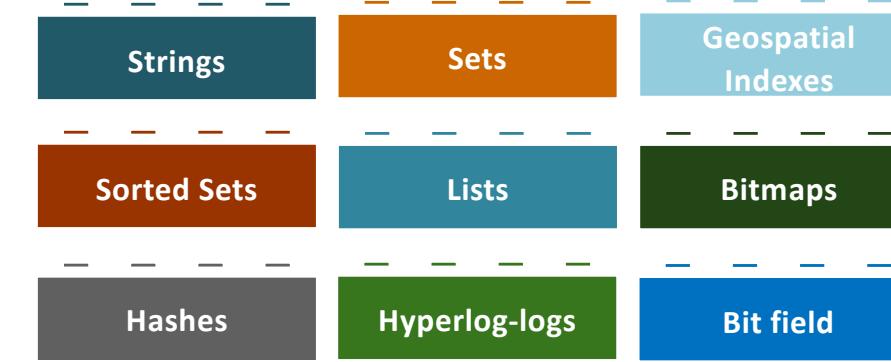
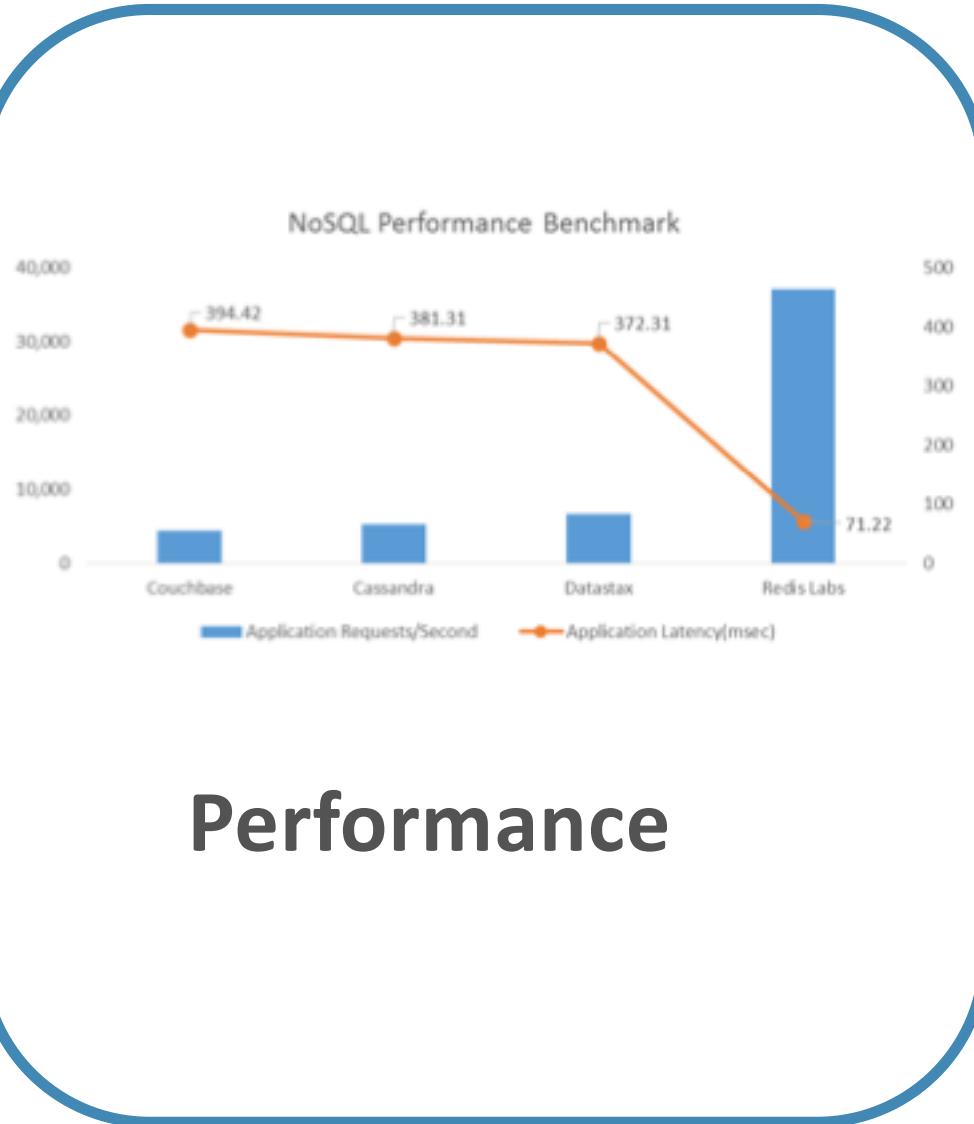
A full range of capabilities that simplify and accelerate next generation applications



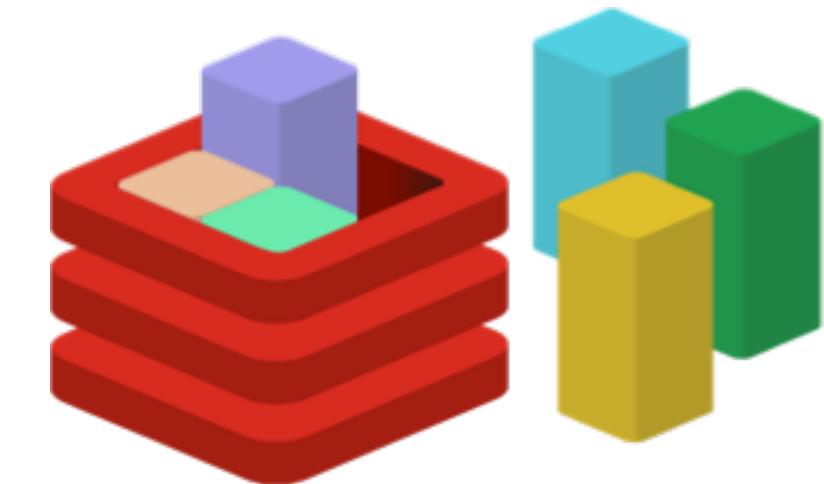
Redis Uses Span Many Verticals

 Telco Billing (CDRs, SDRs)	 Finance High speed delivery of prices / transactions	 Business Services CRM, ERP
 Retail / E-Commerce Items Viewed, Similar Purchases, Top trends	 Technology High-Speed Operations	 Advertising Real-time ad placements, personalization
 Travel Recommendations, Ordering	 Media Notifications, Recommendations, Caching	 Education Subjects, Classes Classification
 Social Timeline, social graph, top followers, following	 Gaming Real time analytics for leaderboards, dashboards, messaging	

Redis is a Game Changer



Simplicity
(through Data Structures)



Extensibility
(through Redis Modules)

Redis is Effective in OLTP and OLAP

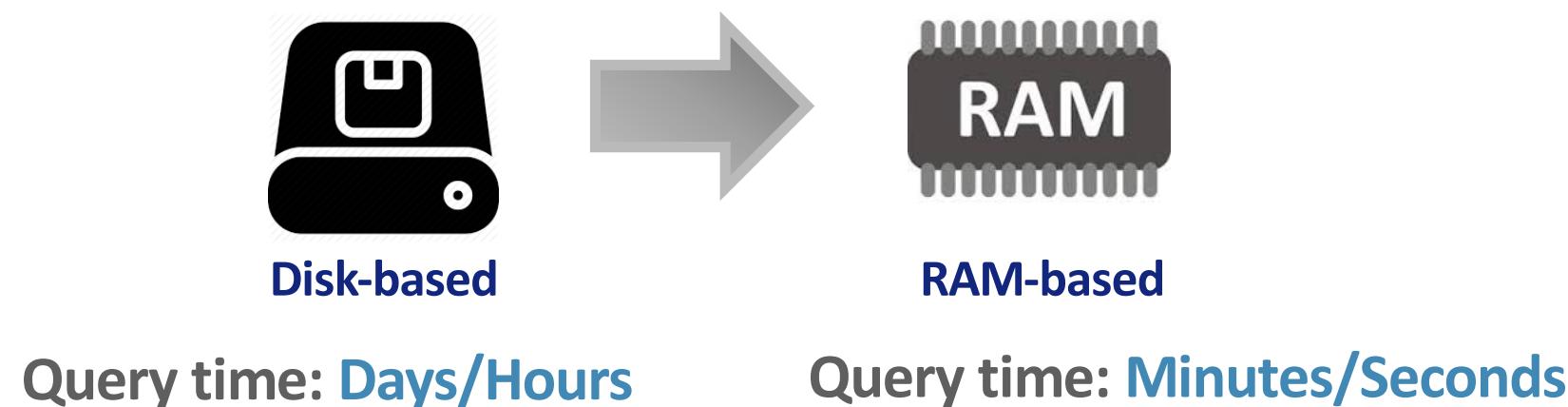
OLTP



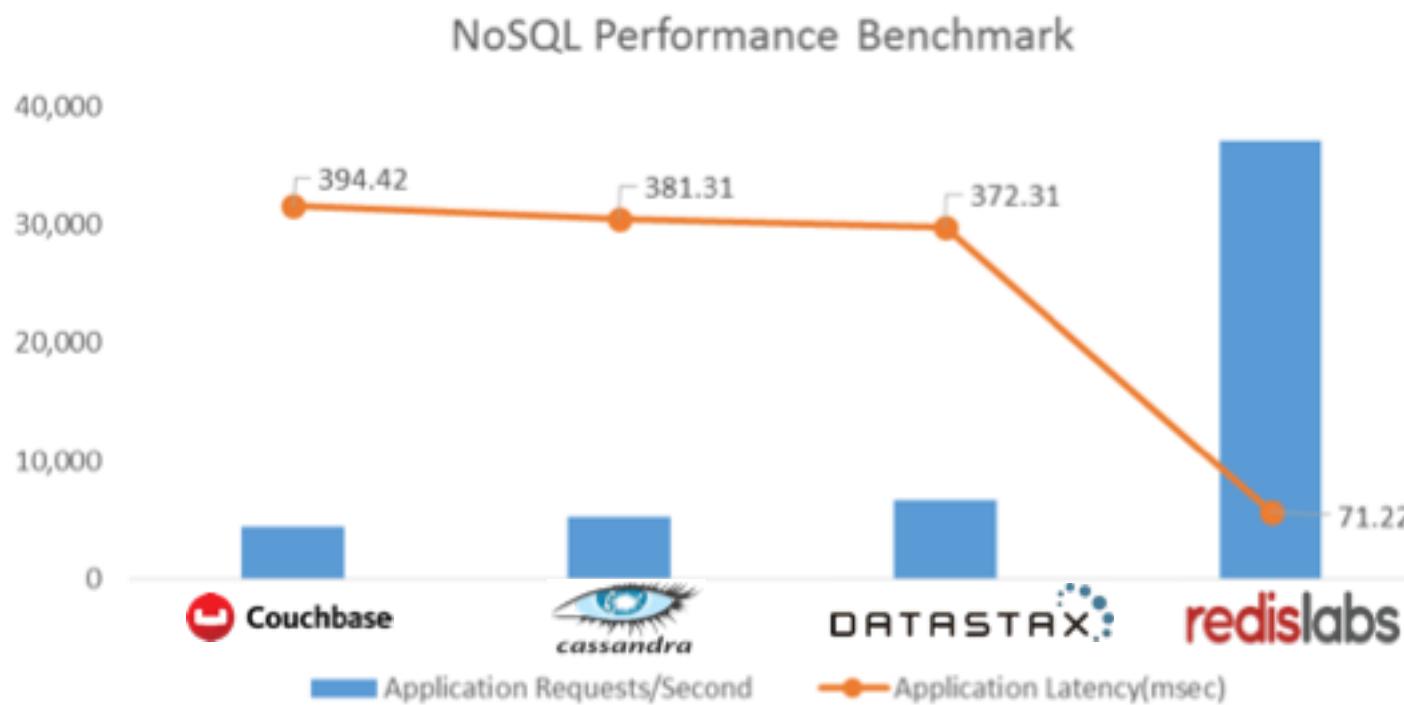
The 100 msec de-facto standard for end-to-end app response time
requires <1msec DB response time.

Redis is the only database that can support this under heavy load.

OLAP

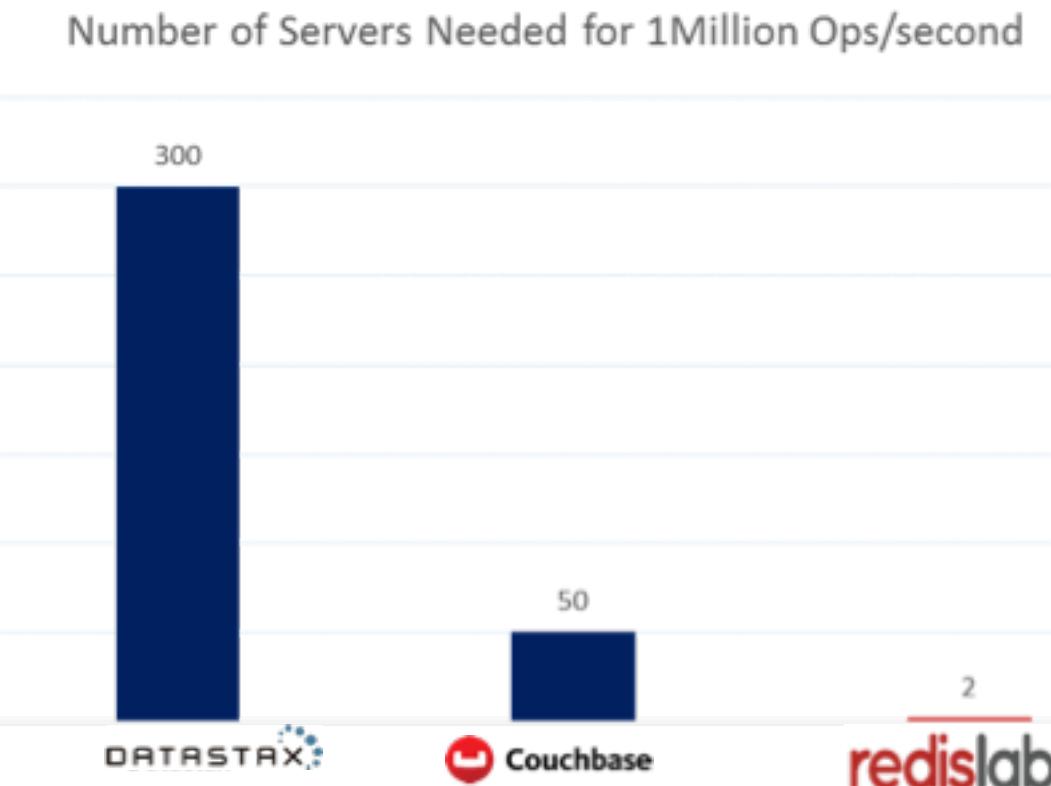


Performance: The Most Powerful Database



Highest Throughput at Lowest Latency
in High Volume of Writes Scenario

Benchmarks performed by Avalon Consulting Group



Lowest number of servers needed to
deliver 1 Million writes/second

Benchmarks published in the Google blog

Redis – Purpose Built For Performance

Optimized Architecture

Written in C

Served entirely from memory

Single threaded lock free

Advanced Processing

Most commands are executed
with O(1) complexity

Access to discrete elements
within objects

Reduced bandwidth/overhead
requirements

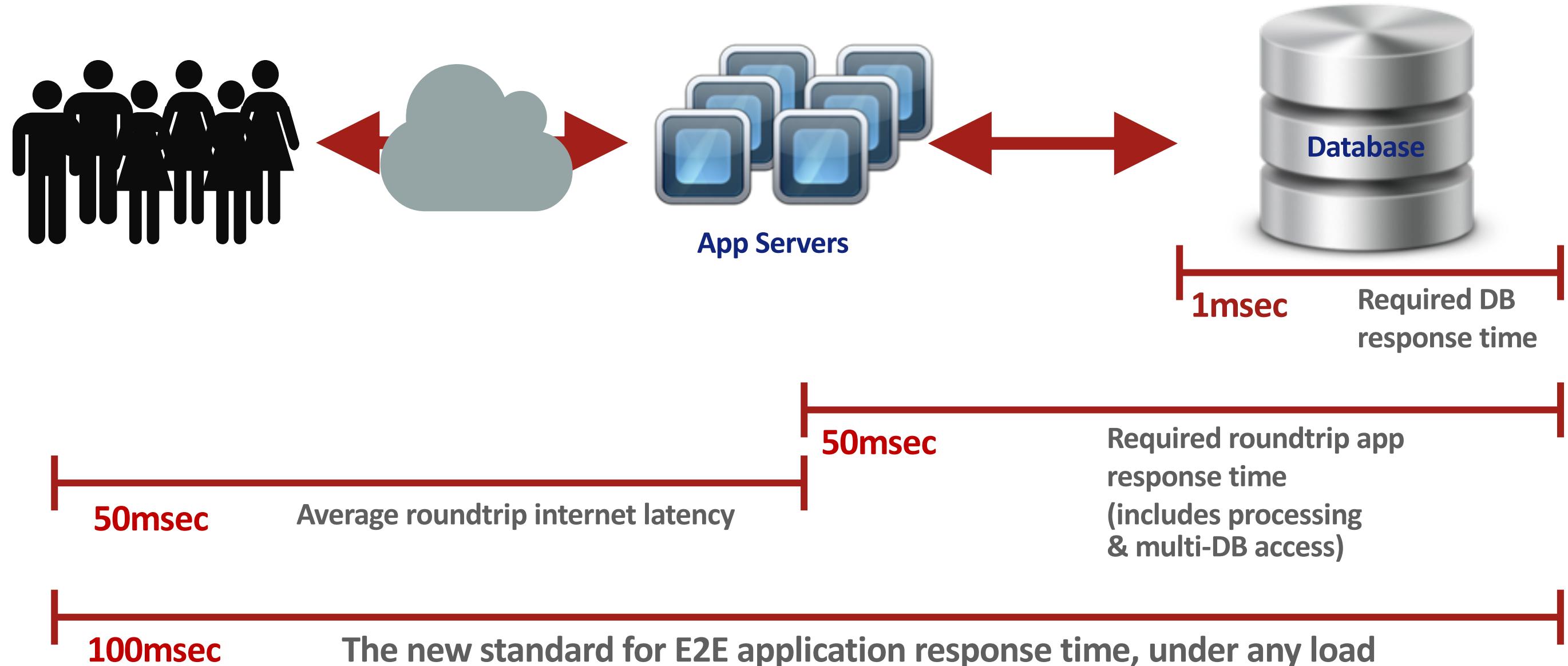
Efficient Operation

Easy to parse networking
protocol

Pipelining for reduced network
overhead

Connection pooling

Why Use Redis as an Operational DBMS ?



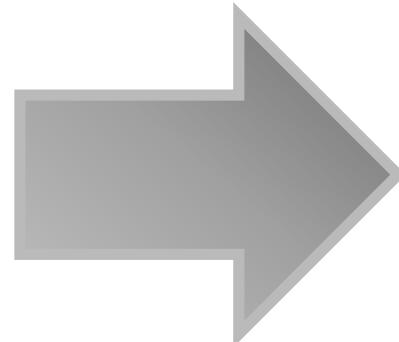
Why Use Redis as an Analytics DBMS ?

Query time:

Days/Hours

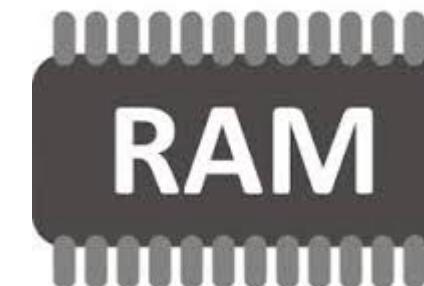


Disk-based



Query time:

Minutes/Seconds



RAM-based

[More on Redis in Analytics](#)

Redis Labs Introduction

Who We Are



Open source. The leading **in-memory database platform**, supporting any high performance operational, analytics or hybrid use case.

The open source home and commercial provider of **Redis Enterprise (Redis^e)** technology, platform, products & services.

What is Redis?

- Redis – (REmote DIctionary Server)
- Open source.
- The leading **in-memory database platform**
- Created in 2009 by Salvatore Sanfilippo (a.k.a @antirez)
- Source: <https://github.com/antirez/redis>
- Website: <http://redis.io>

REmote DIctionary Server

Strings

Hashes

Lists

Sets

Bitmaps

Hyperlog-
logs

Sorted
Sets

Geospatial

Bitfield

Redis Tops Database Popularity Rankings



.....#1 database technology on AWS



.....#1 database used by Node.js developers



.....#1 NoSQL in User Satisfaction



.....#1 NoSQL among Top 10 Data Stores



.....#1 database on Docker



#1 NoSQL database deployed in containers



.....#1 in growth among top 3 NoSQL databases



.....#1 database in skill demand



.....# 1 database in Top Paying Technologies



The Redis Community

162

CLIENTS IN 48 LANGUAGES

100+

HIGHER LEVEL LIBRARIES AND TOOLS

219

CONTRIBUTORS

6K+

GITHUB COMMITS

21.5K

REDIS GITHUB STARS (#75)

39K+

STACK OVERFLOW QUESTIONS

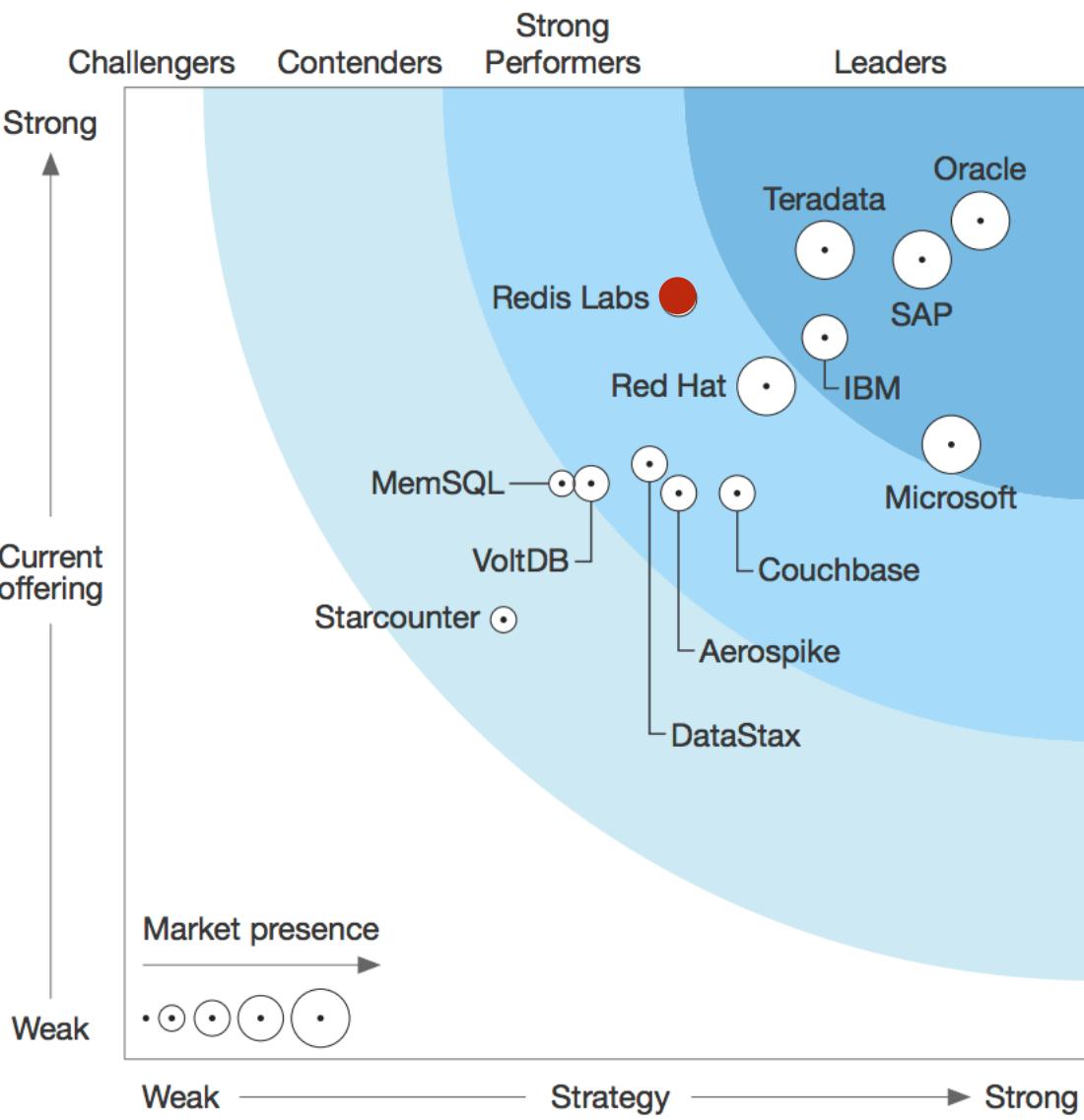
Industry Recognition

Gartner®

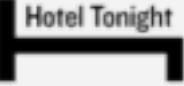
Figure 1. Magic Quadrant for Operational Database Management Systems



FORRESTER®



Customers From All Verticals

Banks	Financial Services	Retail/E-commerce	Social	Media	Advertising
  	  	  	  	  	  
Technology	Communications	Business Services	Travel	Gaming	Education
  	  	  	  	  	  

Redis is Extensively & Diversely Used

Company	Use Case	Scope
Twitter	Timeline, follower, following	0.5-1PB, 30 MM ops/sec
Weibo (Chinese Twitter)	Entire database	± 100 TB, 10MM ops/sec
Samsung US	Fast data store for mobile apps	50MM users, <100 msec (E2E)
HTC	Fast data store for mobile apps	40TB
Pinterest	Graph database	10+TB
Stack overflow	Local/site/global caching	

Redis is Extensively & Diversely Used (2)

Company	Use Case	Scope
Booking.com	Online bookings/fast transactions	10-20 TB
Github	Repository router	10+ TB
Tinder	Geo search, user profiles	10-20TB
Snapchat	All messages	40TB
Pinterest	Graph database	500 MM messages/day, >100TB

Redis Labs Products



Redis^e Cloud

Fully managed Redis^e service on hosted servers within AWS, MS Azure, GCP, IBM Softlayer, Heroku, CF & OpenShift



Redis^e Cloud Private

Fully managed Redis^e service in VPCs within AWS, MS Azure, GCP & IBM Softlayer



or



SOFTWARE



Redis^e Pack

Downloadable Redis^e software for any enterprise datacenter or cloud environment



or



Redis^e Pack Managed

Fully managed Redis^e Pack in private data centers



or



Redis^e Cloud - Offered Over IaaS and PaaS

4 Clouds, 45 data centers across the world

