Alternatives to Redis

Bill Sendewicz
PronunciationProfessor.com
EmergencyContact.info

ThaiPy
November 2025

PronunciationProfessor.com

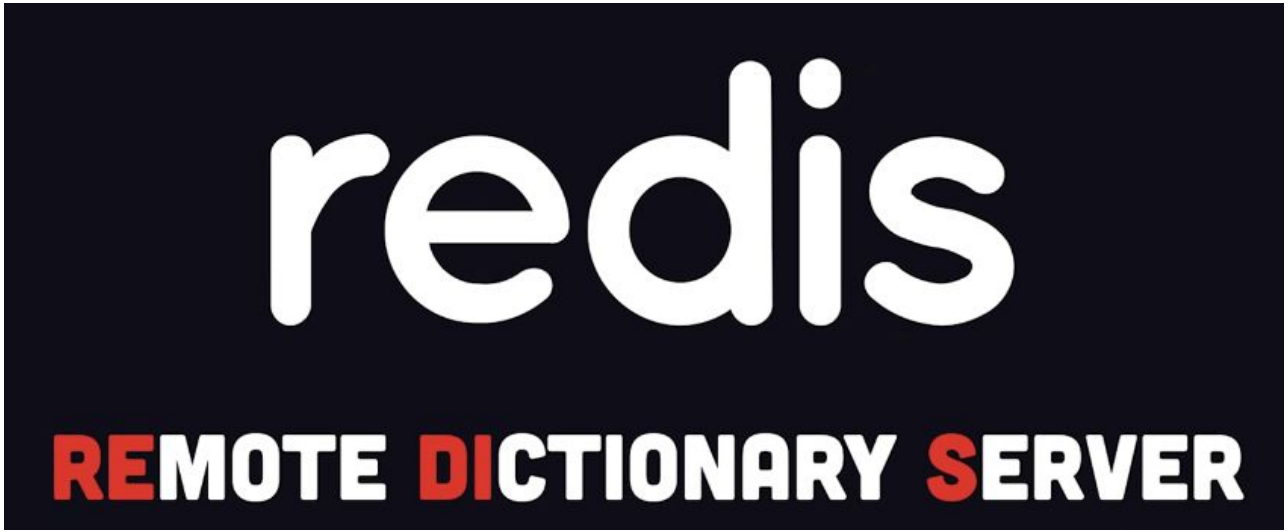
Table of Contents

Feedback Form

Feedback form: https://forms.gle/X7xTmjL8VTkbBWzu9

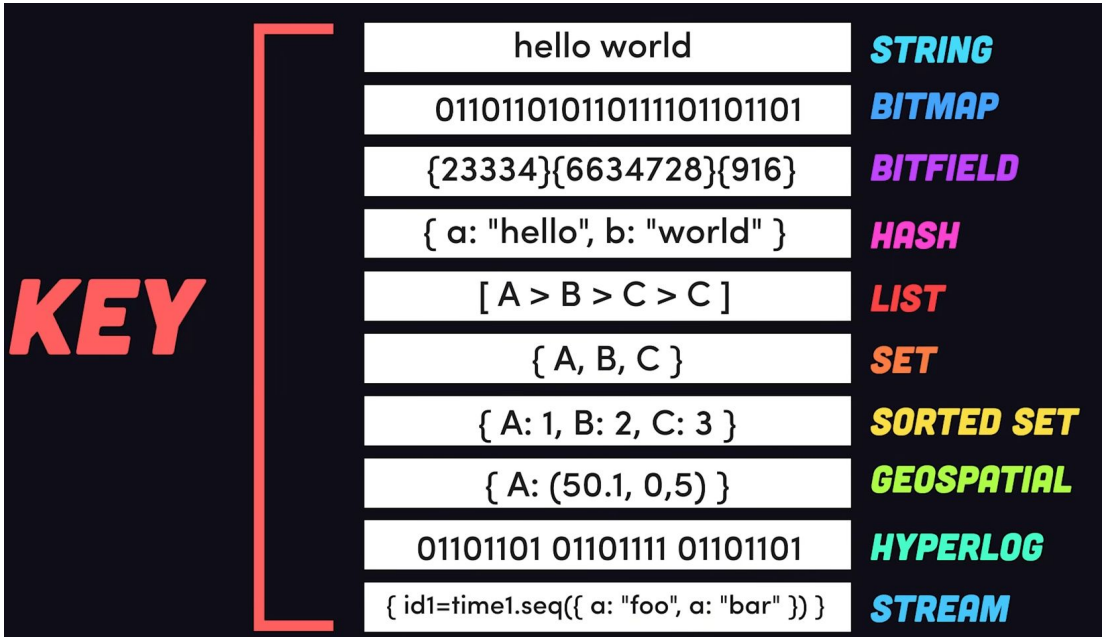
Chapter 1: What is Redis?

Chapter 1: What is Redis?

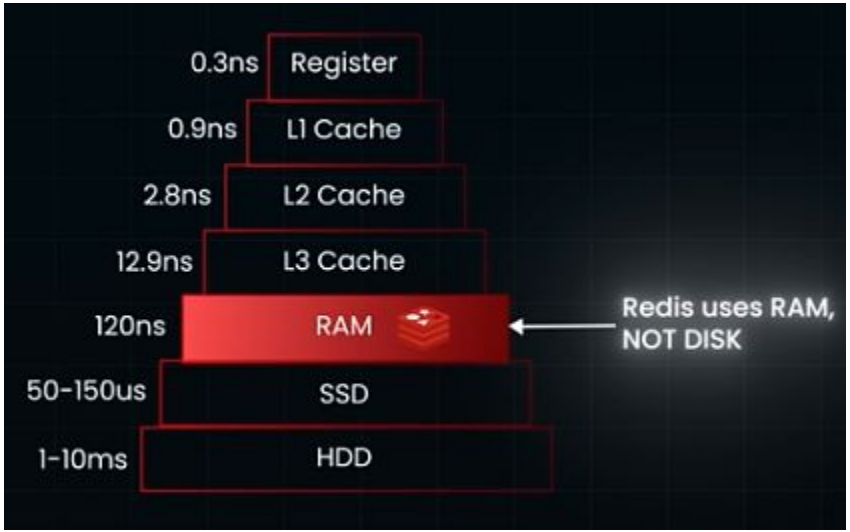


Chapter 1: What is Redis?

Chapter 1: What is Redis?

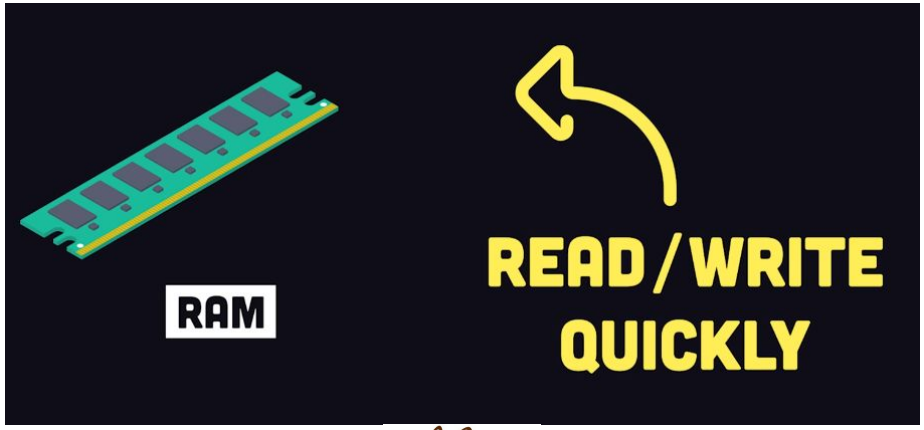


Chapter 1: What is Redis?



over 400x faster

Chapter 1: What is Redis?

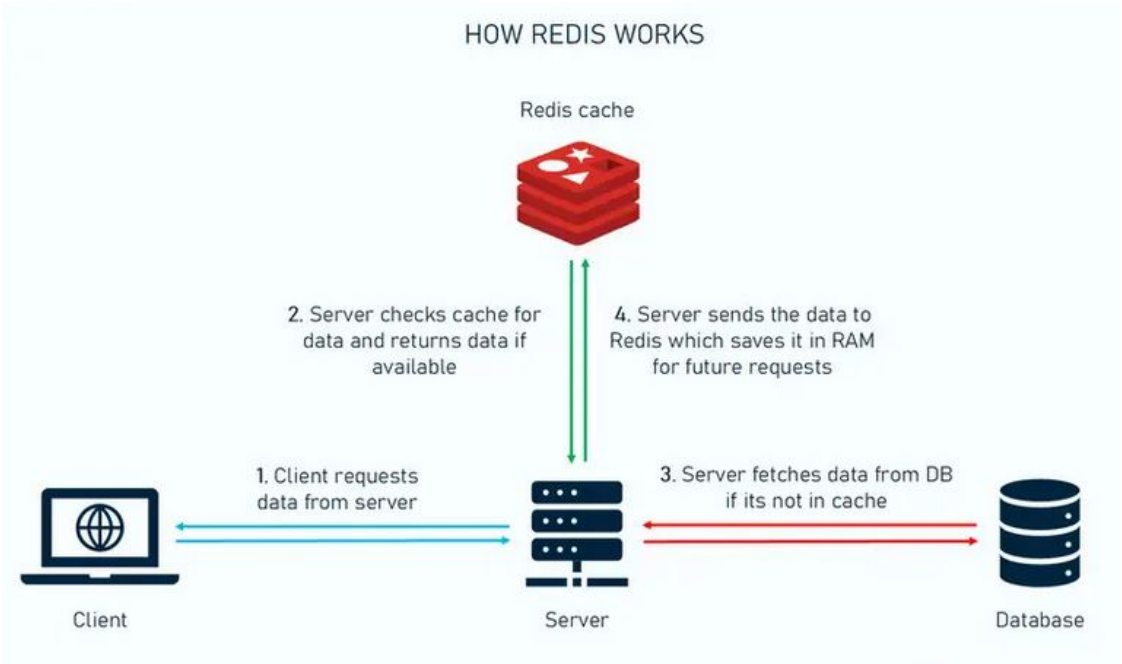


Fast

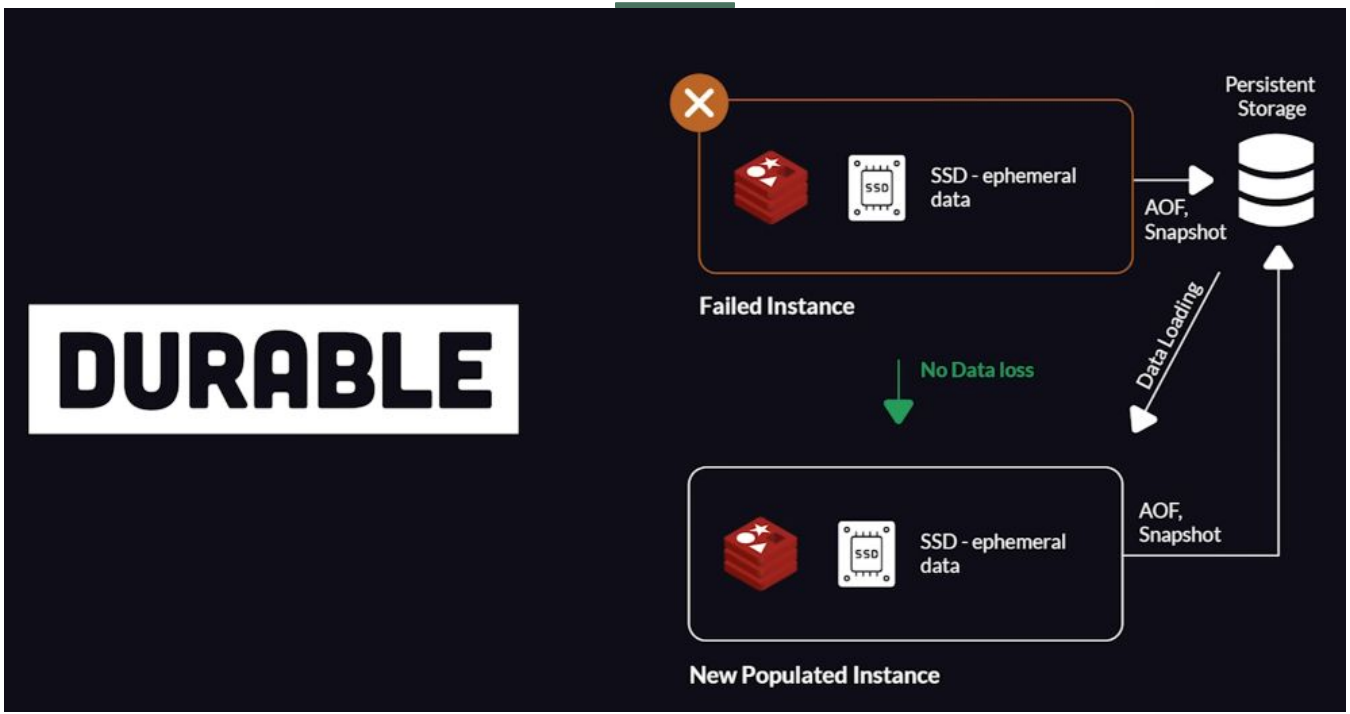
Slow

Chapter 2: Why Use Redis?

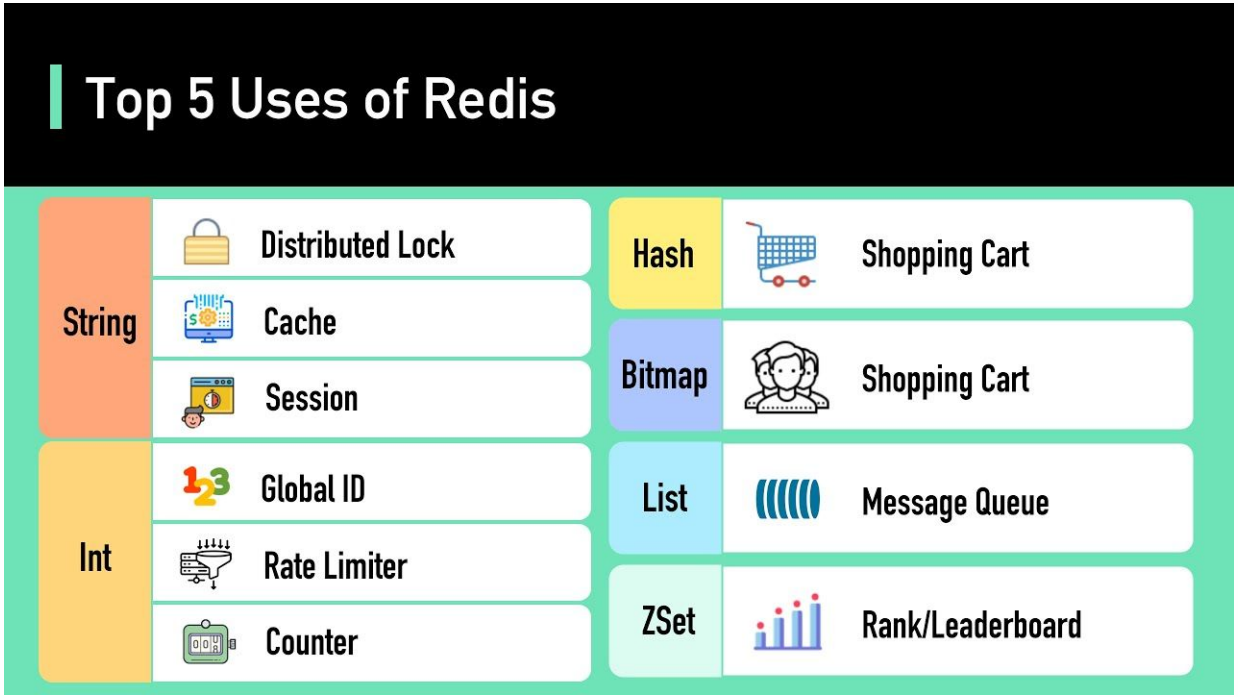
Chapter 2: Why Use Redis?



Chapter 2: Why Use Redis?



Chapter 2: Why Use Redis?

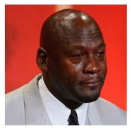


Chapter 3: Redis Shortcomings

Chapter 3: Redis Shortcomings

Chapter 3: Redis Shortcomings

Indie Hacker on Twitter Starter Pack



learning platform risk
(the hard way)
(again)

Location is an MRR scale

Location 22 / 30
$4k MRR

<insert angry comment about VCs>

"freedom"

Brightly coloured profile picture

Bio is full of projects

Indie hacker.

SomeTwitterApp.so
ProductivityTool.app
IndieHackingCourse.com (Gumroad even though they said they swore to leave)

"building an audience"

another tweet about why marketing is important (but they're tweeting about it instead of actually doing it)

12 Following 1,234 Followers

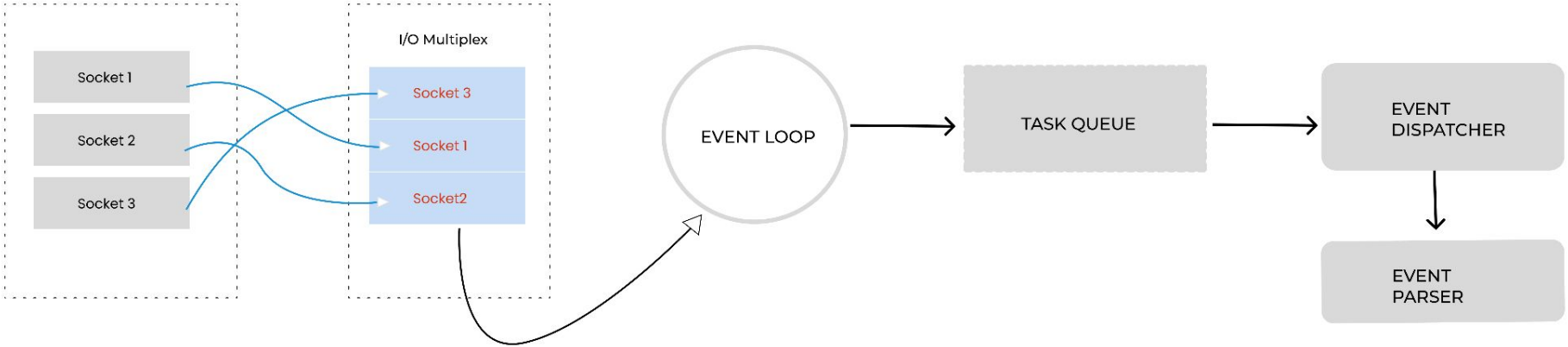
Uses lists to follow people

"I quite my full time job to..."

Chapter 3: Redis Shortcomings

Chapter 3: Redis Shortcomings

- Single-threaded



I/O MULTIPLEXING AND SINGLE THREADED READ /WRITE

Chapter 3: Redis Shortcomings

- Single-threaded
  - Event loop runs continuously to check for events

Chapter 3: Redis Shortcomings

- Single-threaded
  - Event loop runs continuously to check for events
  - When an event is triggered (e.g., client sends a command)

Chapter 3: Redis Shortcomings

- Single-threaded
  - Event loop runs continuously to check for events
  - When an event is triggered (e.g., client sends a command)
    - event loop processes request synchronously

Chapter 3: Redis Shortcomings

- Single-threaded
  - Event loop runs continuously to check for events
  - When an event is triggered (e.g., client sends a command)
    - event loop processes request synchronously
    - executes command

Chapter 3: Redis Shortcomings

- Single-threaded
  - Event loop runs continuously to check for events
  - When an event is triggered (e.g., client sends a command)
    - event loop processes request synchronously
    - executes command
    - returns response

Chapter 3: Redis Shortcomings

- Multiplexing

Chapter 3: Redis Shortcomings

- Multiplexing: one single thread watches many socket connections at once
  - Non-blocking sockets

Chapter 3: Redis Shortcomings

- Multiplexing: one single thread watches many socket connections at once
  - Non-blocking sockets
    - asynchronous read/write operations on a socket without having to wait for operations to complete

Chapter 3: Redis Shortcomings

- Multiplexing: one single thread watches many socket connections at once
  - Non-blocking sockets
    - asynchronous read/write operations on a socket without having to wait for operations to complete
    - If the operation cannot be performed immediately (e.g., data not available), the call returns immediately, letting the application continue executing other code

Chapter 3: Redis Shortcomings

- Multiplexing: one single thread watches many socket connections at once
  - Non-blocking sockets → high throughput

Chapter 3: Redis Shortcomings

- Multiplexing: one single thread watches many socket connections at once
  - Non-blocking sockets → high throughput
  - Single thread monitors multiple input/output channels

Chapter 3: Redis Shortcomings

- Multiplexing: one single thread watches many socket connections at once
  - Non-blocking sockets → high throughput
  - Single thread monitors multiple input/output channels
  - Can handle multiple socket connections to clients; no need to create new thread for each connection

Chapter 3: Redis Shortcomings

- Multiplexing: one single thread watches many socket connections at once
  - Non-blocking sockets → high throughput
  - Single thread monitors multiple input/output channels
  - Can handle multiple socket connections to clients; no need to create new thread for each connection
  - Hundreds or thousands of client connections simultaneously with just a single thread

Chapter 3: Redis Shortcomings

- CPU-intensive tasks

Chapter 3: Redis Shortcomings

- CPU-intensive tasks
  - Redis processes commands sequentially

Chapter 3: Redis Shortcomings

- CPU-intensive tasks
  - Redis processes commands sequentially
  - High latency when other requests wait for intensive tasks to finish

Chapter 3: Redis Shortcomings

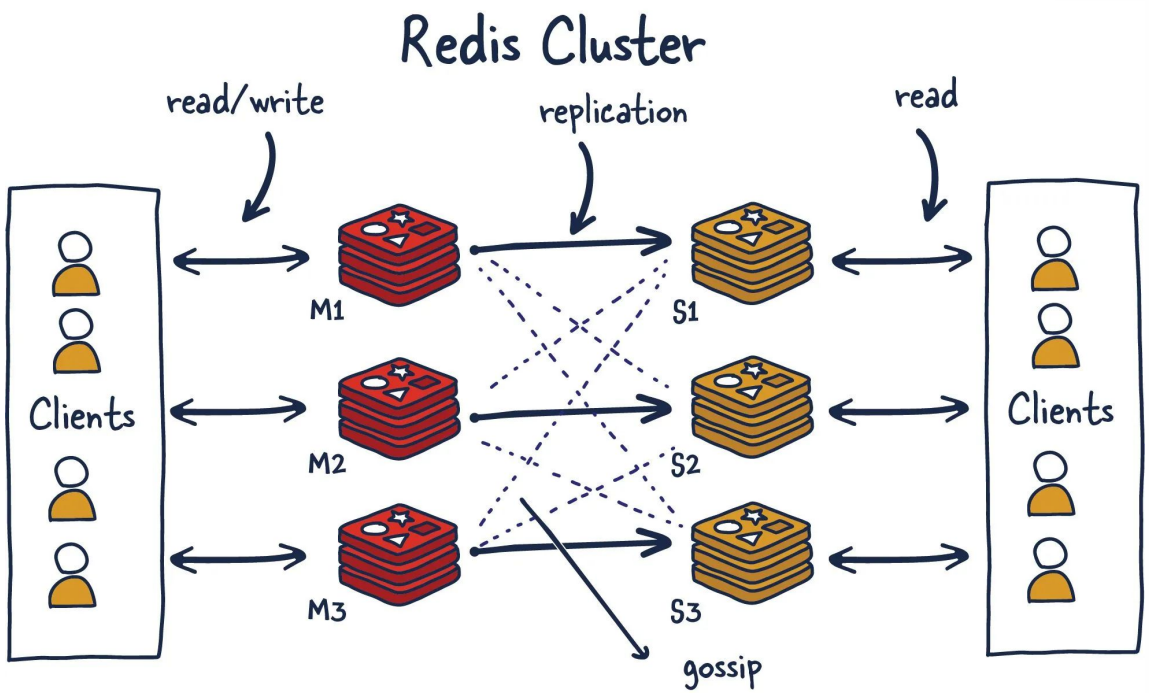
- CPU-intensive tasks
  - Redis processes commands sequentially
  - High latency when other requests wait for intensive tasks to finish
  - **Cannot use multiple cores for heavy computations because Redis is single-threaded**

Chapter 3: Redis Shortcomings

Solution?

- Redis cluster

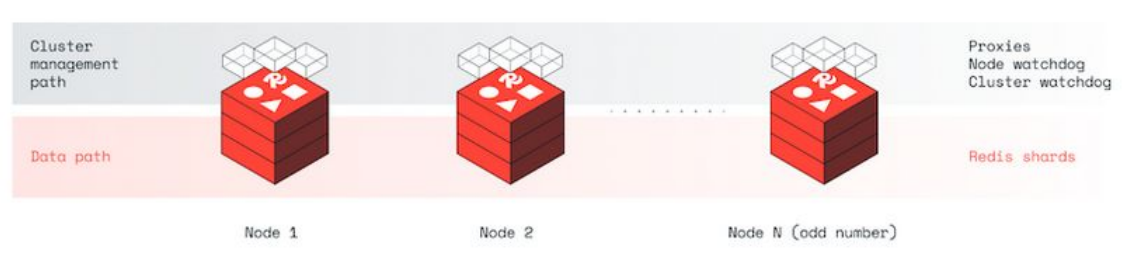
Chapter 3: Redis Shortcomings



Chapter 3: Redis Shortcomings

Solution?

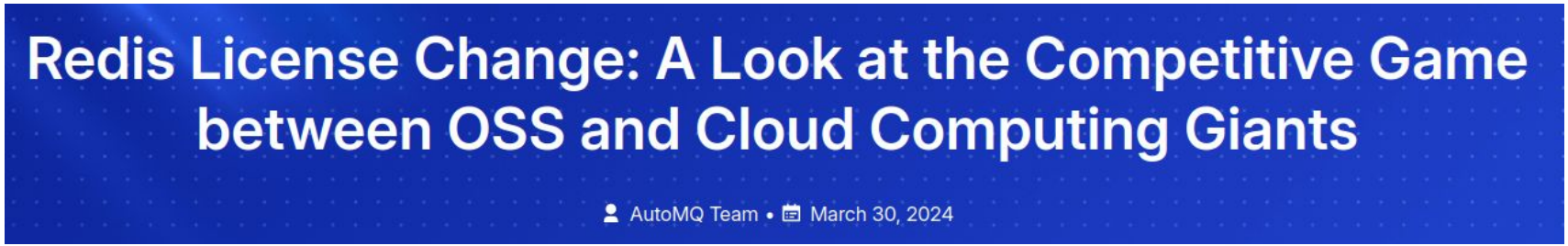
- Redis cluster
- Dataset sharding



Chapter 3: Redis Shortcomings

- Single-threaded
- License change drama

Chapter 3: Redis Shortcomings



Chapter 3: Redis Shortcomings

License change drama

- Inception 2009: BSD 3-Clause License

Chapter 3: Redis Shortcomings

License change drama

- Inception 2009: BSD 3-Clause License
  - permissive open source license allowing almost unrestricted use, modification, and distribution of software, including commercial use

Chapter 3: Redis Shortcomings

License change drama

- Inception 2009: BSD 3-Clause License
- March 2024: dual license model
  - Redis Source Available License v2
  - Server Side Public License v1

Chapter 3: Redis Shortcomings

Redis Source Available License v2

- source-available but restrictive license created by Redis Labs

Chapter 3: Redis Shortcomings

Redis Source Available License v2

- source-available but restrictive license created by Redis Labs
- prevents cloud providers from offering Redis as a managed service without complying with Redis Labs' terms

Chapter 3: Redis Shortcomings

Redis Source Available License v2

- source-available but restrictive license created by Redis Labs
- prevents cloud providers from offering Redis as a managed service without complying with Redis Labs' terms
- view/modify source code but restricts certain commercial and service-related use cases to protect the business model

Chapter 3: Redis Shortcomings

Server Side Public License v1

- anyone offering the software as a service must open source the management software and infrastructure that they use to provide the service

Chapter 3: Redis Shortcomings

Server Side Public License v1

- anyone offering the software as a service must open source the management software and infrastructure that they use to provide the service
- intended to prevent use by cloud/service providers without contributing back

Chapter 3: Redis Shortcomings

License change drama

- Inception 2009: BSD 3-Clause License
- March 2024 (v7.4): dual license model
- May 2025 (v8.0): tri-license model

Chapter 3: Redis Shortcomings

License change drama

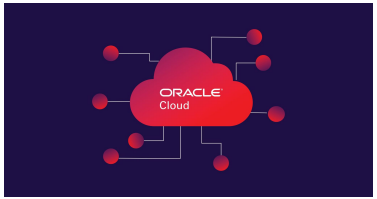
- Inception 2009: BSD 3-Clause License
- March 2024 (v7.4): dual license model
- May 2025 (v8.0): tri-license model
  - Redis Source Available License v2
  - Server Side Public License v1
  - GNU Affero General Public License v3

Chapter 3: Redis Shortcomings

GNU Affero General Public License v3

- if software is modified and run as a network service, the complete source code must be made available to users of that service

Chapter 3: Redis Shortcomings



Chapter 3: Redis Shortcomings

Chapter 3: Redis Shortcomings

- Single-threaded
- License change drama
- Personal opinion: Getting too corporate

Chapter 3: Redis Shortcomings

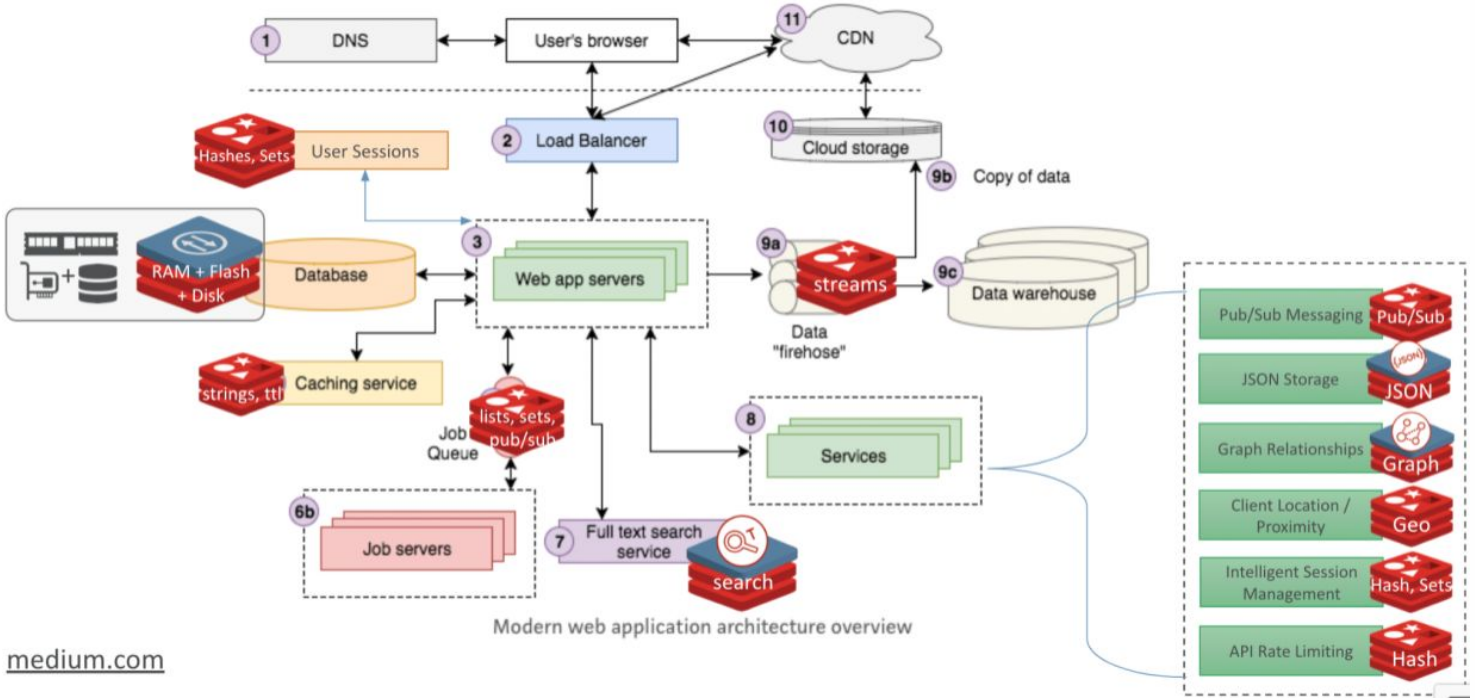
Chapter 3: Redis Shortcomings

Chapter 3: Redis Shortcomings

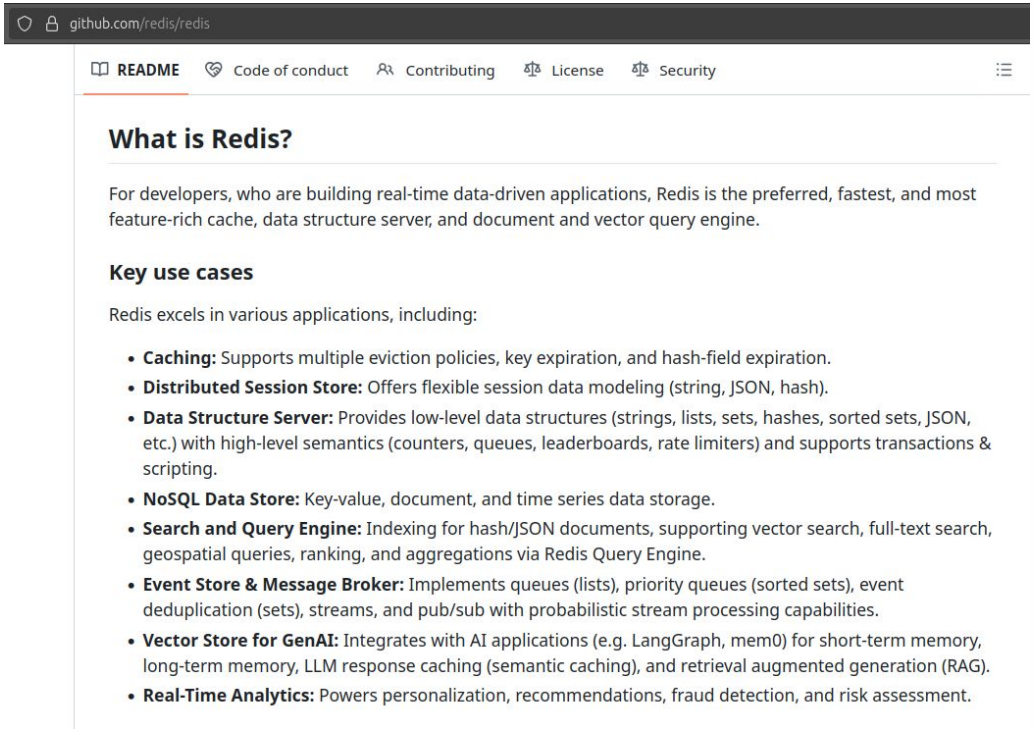
Chapter 3: Redis Shortcomings

- Single-threaded
- License change drama
- Typically run in a cluster
- Personal opinion: Getting too corporate
- Personal opinion: Trying to do too much

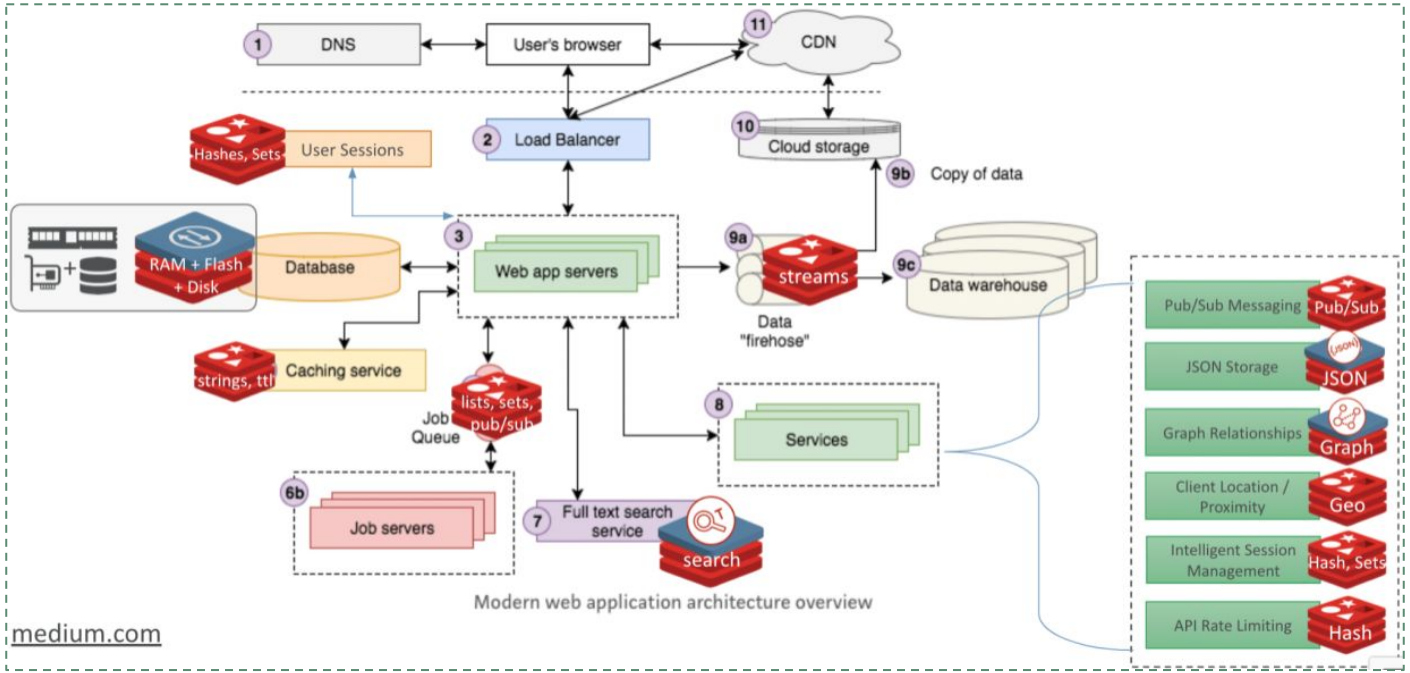
Chapter 3: Redis Shortcomings



Chapter 3: Redis Shortcomings



Chapter 3: Redis Shortcomings



Redis

Hi Bill,

Vector databases are transforming the way orgs like yours build and manage apps —with personalized user experiences, real-time inventory management, and automated subscription processes.

Watch Making it easy to build GenAI apps to learn how you can start using Redis as your vector database to build fast GenAI apps today.

During the session, Tyler Hutcherson, Sr. Applied Machine Learning Engineer at Redis, covers different use cases and explains how to make GenAI apps smarter and more cost-effective using technology you already know.

Watch the webinar

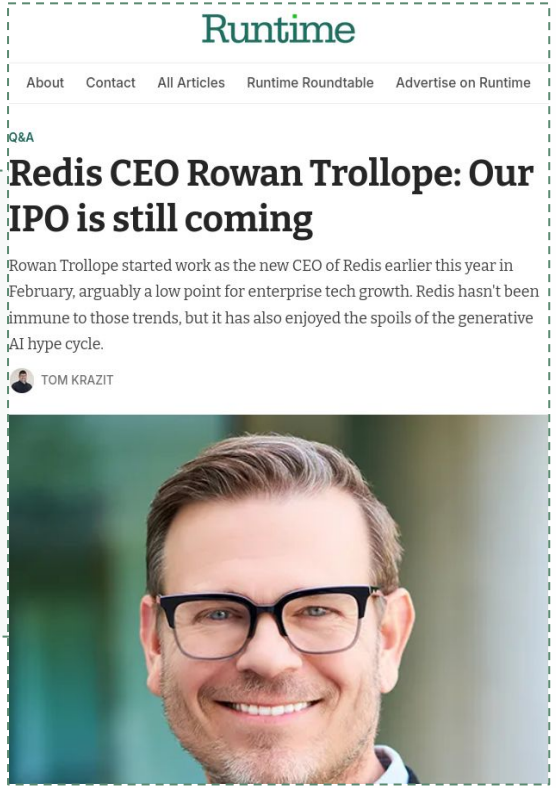
Need a vector database but don't know where to start? Talk to an expert today.

Chapter 3: Redis Shortcomings

Why?

Chapter 3: Redis Shortcomings

August 17, 2023:



Runtime

About Contact All Articles Runtime Roundtable Advertise on Runtime

Q&A

Redis CEO Rowan Trollope: Our IPO is still coming

Rowan Trollope started work as the new CEO of Redis earlier this year in February, arguably a low point for enterprise tech growth. Redis hasn't been immune to those trends, but it has also enjoyed the spoils of the generative AI hype cycle.

TOM KRAZIT

Chapter 3: Redis Shortcomings

- Single-threaded
- License change drama
- Typically run in a cluster
- Personal opinion: Getting too corporate
- Personal opinion: Trying to do too much
    - arguably a dumb opinion

Chapter 4: Alternatives to Redis

Chapter 4: Alternatives to Redis

Ignore managed options; focus on self-hosting instead

Chapter 4: Alternatives to Redis

- Aerospike
- Dragonfly
- Garnet
- KeyDB
- Memcached
- Valkey
- ...many more depending on how you plan to use

Chapter 4: Alternatives to Redis

- Aerospike
- Dragonfly
  - built ground-up
- Garnet
- KeyDB
- Memcached
- Valkey
  - Redis fork

Chapter 4: Alternatives to Redis

Chapter 4: Alternatives to Redis

```
redis:
  image: redis:7-alpine
```

- Aerospike
- Dragonfly
- Garnet
- KeyDB
- Memcached
- Valkey
- ...many more depending on how you plan to use

```
dragonfly:
  image: docker.dragonflydb.io/dragonflydb/dragonfly
```

```
keydb:
  image: eqalpha/keydb:latest
```

```
valkey:
  image: valkeydb/valkey:latest
```

Chapter 4: Alternatives to Redis

What is a drop-in database?

Chapter 4: Alternatives to Redis

What is a drop-in database?

- Drop-in Redis replacement supports the exact same commands, protocols and APIs

Chapter 4: Alternatives to Redis

What is a drop-in database?

- Drop-in Redis replacement supports the exact same commands, protocols and APIs
- Can switch datastore without needing to change application code or client libraries

Chapter 4: Alternatives to Redis

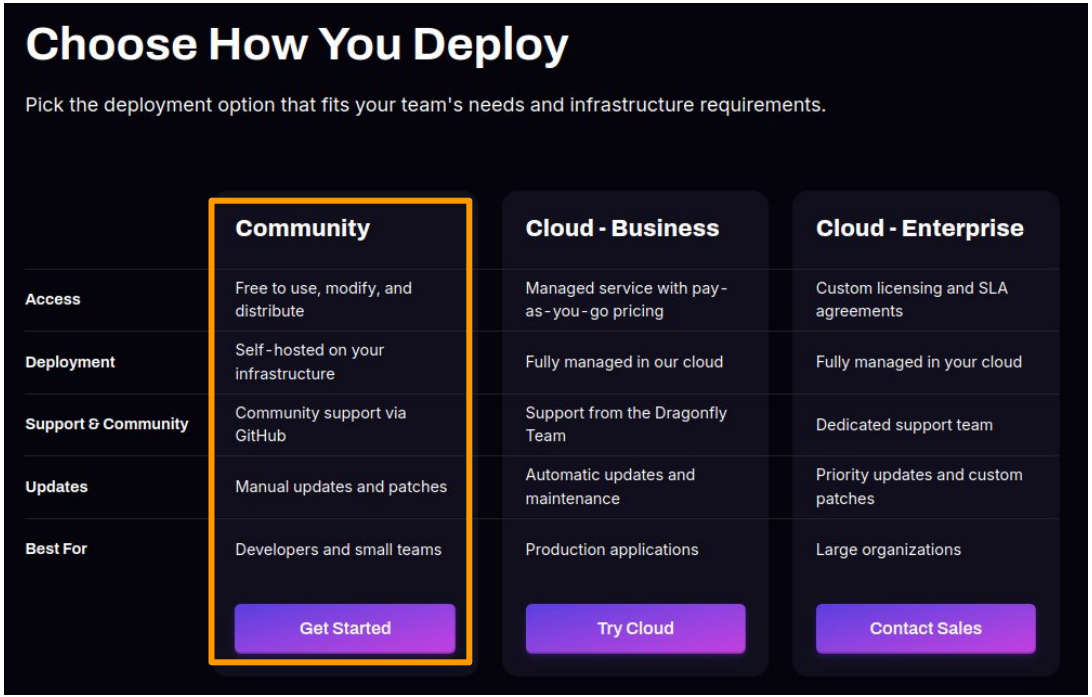
What is a drop-in database?

- Drop-in Redis replacement supports the exact same commands, protocols and APIs
- Can switch datastore without needing to change application code or client libraries
- Seamless migration or experimentation with alternatives

Chapter 5: What I Chose for Pronunciation Professor and Why

Chapter 5: What I Chose for Pronunciation Professor and Why

Chapter 5: What I Chose for Pronunciation Professor and Why



Chapter 5: What I Chose for Pronunciation Professor and Why

Why Dragonfly?

- Drop-in replacement

Chapter 5: What I Chose for Pronunciation Professor and Why

Why Dragonfly?

- Drop-in replacement
- Multi-threaded

Chapter 5: What I Chose for Pronunciation Professor and Why

Why Dragonfly?

- Drop-in replacement
- Multi-threaded
- Lightweight

Chapter 5: What I Chose for Pronunciation Professor and Why

Why Dragonfly?

- Drop-in replacement
- Multi-threaded
- Lightweight
- Reasonable license

Chapter 5: What I Chose for Pronunciation Professor and Why

Why Dragonfly?

- Drop-in replacement
- Multi-threaded
- Lightweight
- Reasonable license: Business Source License (BSL) 1.1

Chapter 5: What I Chose for Pronunciation Professor and Why

Business Source License (BSL) 1.1

- code is free to use and modify for community and production purposes
- restrictions primarily aimed at preventing users from offering DragonflyDB as a competing managed service

Chapter 5: What I Chose for Pronunciation Professor and Why

Why Dragonfly?

- Drop-in replacement
- Multi-threaded
- Lightweight
- Reasonable license
- Try something new

Chapter 5: What I Chose for Pronunciation Professor and Why

Benchmarking

- Don’t believe the hype
- Apples and oranges

Chapter 5: What I Chose for Pronunciation Professor and Why

Benchmarking

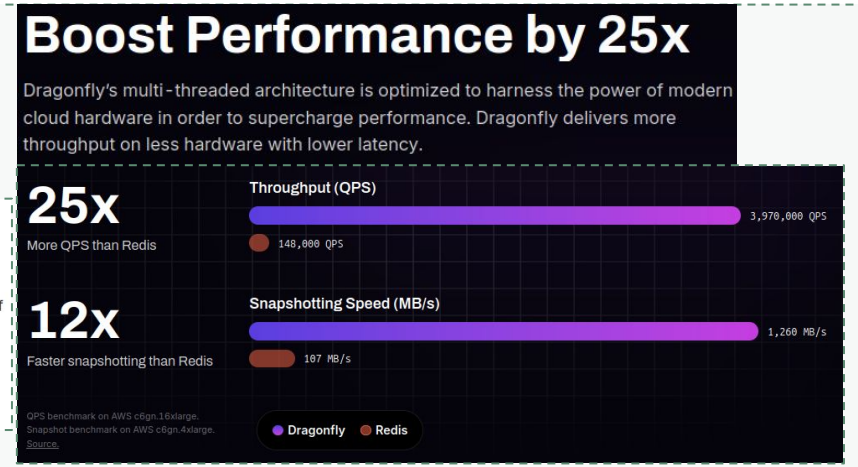
AWS › EC2 › c6gn.4xlarge

c6gn.4xlarge

The c6gn.4xlarge instance is in the Compute optimized family with 16 vCPUs, 32 GiB of memory and 25 Gibps of bandwidth starting at $0.6912 per hour.

$ Pricing

$0.691	$0.27	$0.436	$0.297
On Demand	Spot	1-Year Reserved	3-Year Reserved



Chapter 5: What I Chose for Pronunciation Professor and Why

Benchmarking

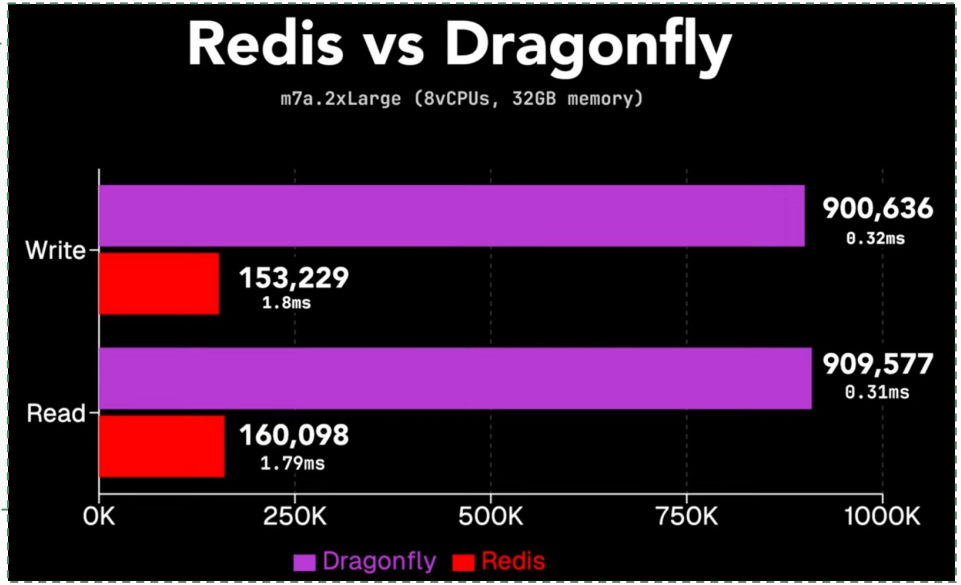
AWS > EC2 > m7a.2xlarge

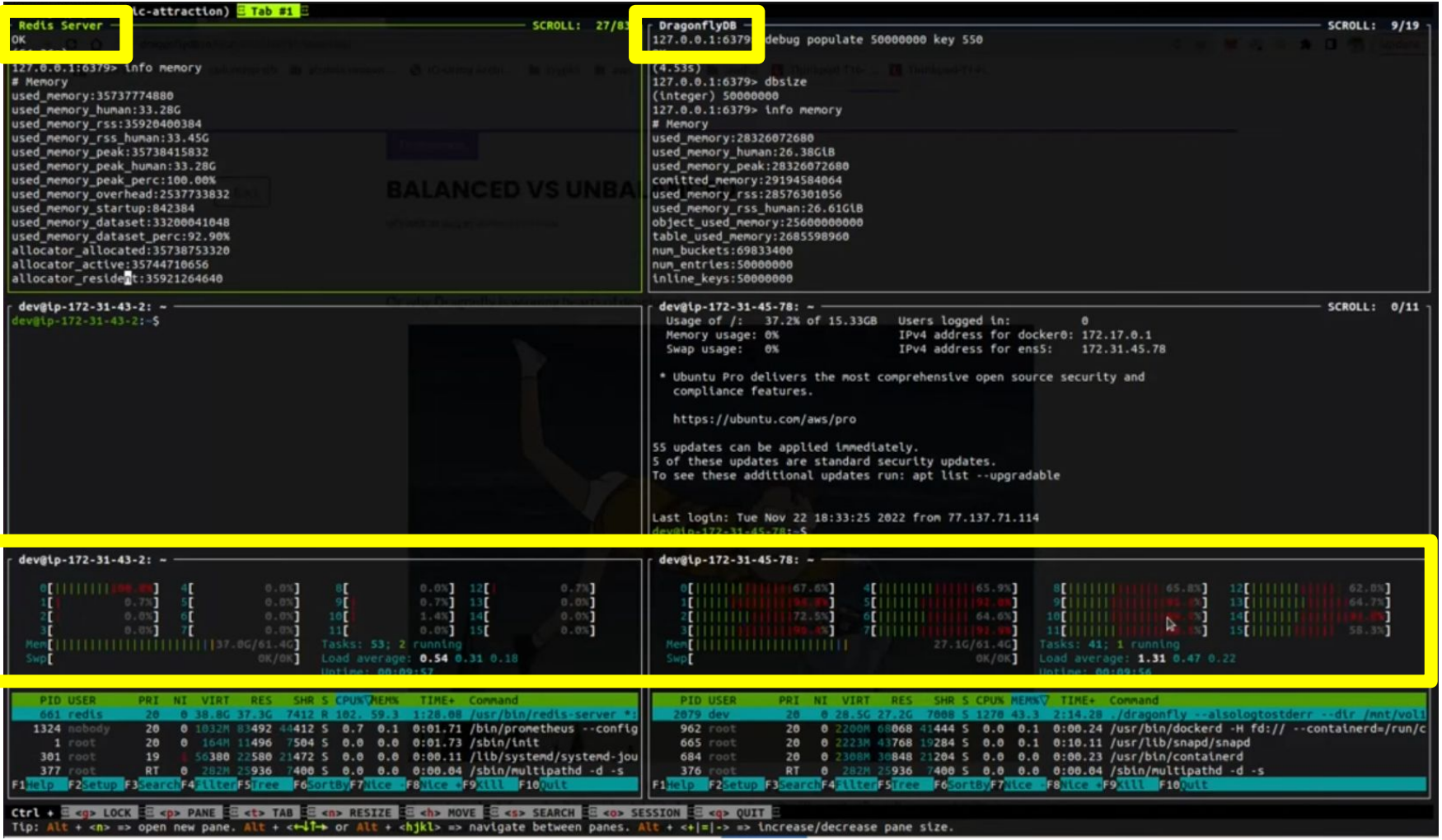
m7a.2xlarge

The m7a.2xlarge instance is in the General purpose family with 8 vCPUs, 32 GiB of memory and up to 12.5 Gibps of bandwidth starting at $0.46368 per hour.

$ Pricing

$0.464	$0.194	$0.307	$0.21
On Demand	Spot	1-Year Reserved	3-Year Reserved




Redis Server
OK
127.0.0.1:6379> info memory
Memory
used_memory:35737774880
used_memory_human:33.28G
used_memory_rss:35920400384
used_memory_rss_human:33.45G
used_memory_peak:35738415832
used_memory_peak_human:33.28G
used_memory_peak_perc:100.00%
used_memory_overhead:2537733832
used_memory_startup:842384
used_memory_dataset:33200041048
used_memory_dataset_perc:92.90%
allocator_allocated:35738753320
allocator_active:35744710656
allocator_resident:35921264640
DragonflyDB
127.0.0.1:6379> debug populate 50000000 key 550
(4.535)
127.0.0.1:6379> dbsize
(integer) 50000000
127.0.0.1:6379> info memory
Memory
used_memory:28326072680
used_memory_human:26.38GiB
used_memory_peak:28326072680
comitted_memory:29194584064
used_memory_rss:28576301056
used_memory_rss_human:26.61GiB
object_used_memory:25600000000
table_used_memory:2685598960
num_buckets:69833400
num_entries:50000000
inline_keys:50000000
dev@ip-172-31-43-2: ~
dev@ip-172-31-43-2:~$
dev@ip-172-31-45-78: ~
Usage of /: 37.2% of 15.33GB
Memory usage: 0%
Swap usage: 0%
Users logged in: 0
IPv4 address for docker0: 172.17.0.1
IPv4 address for ens5: 172.31.45.78
* Ubuntu Pro delivers the most comprehensive open source security and compliance features.
https://ubuntu.com/aws/pro
55 updates can be applied immediately.
5 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable
Last login: Tue Nov 22 18:33:25 2022 from 77.137.71.114
dev@ip-172-31-43-2: ~
dev@ip-172-31-45-78: ~
Tasks: 53; 2 running
Load average: 0.54 0.31 0.18
Tasks: 41; 1 running
Load average: 1.31 0.47 0.22
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
661 redis 20 0 38.8G 37.3G 7412 R 102. 59.3 1:28.08 /usr/bin/redis-server *:
1324 nobody 20 0 1032M 83492 44412 S 0.7 0.1 0:01.71 /bin/prometheus --config
1 root 20 0 164M 11496 7504 S 0.0 0.0 0:01.73 /sbin/init
301 root 19 -1 56380 22580 21472 S 0.0 0.0 0:00.11 /lib/systemd/systemd-jou
377 root RT 0 282M 25936 7400 S 0.0 0.0 0:00.04 /sbin/multipathd -d -s
2079 dev 20 0 28.5G 27.2G 7008 S 1270 43.3 2:14:28 ./dragonfly --alsologtostderr --dir /mnt/voli
962 root 20 0 2200M 68068 41444 S 0.0 0.1 0:00.24 /usr/bin/dockerd -H fd:// --containerd=/run/c
665 root 20 0 2223M 43768 19284 S 0.0 0.1 0:10.11 /usr/lib/snapd/snapd
684 root 20 0 2308M 30848 21204 S 0.0 0.0 0:00.23 /usr/bin/containerd
376 root RT 0 282M 25936 7400 S 0.0 0.0 0:00.04 /sbin/multipathd -d -s
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice F8Nice F9Kill F10Quit
Ctrl + <g> LOCK <p> PANE <t> TAB <n> RESIZE <h> MOVE <s> SEARCH <o> SESSION <q> QUIT
Tip: Alt + <n> => open new pane. Alt + <←↓↑→> or Alt + <hjkl> => navigate between panes. Alt + <+|=|-> => increase/decrease pane size.

Chapter 5: What I Chose for Pronunciation Professor and Why

Benchmarking

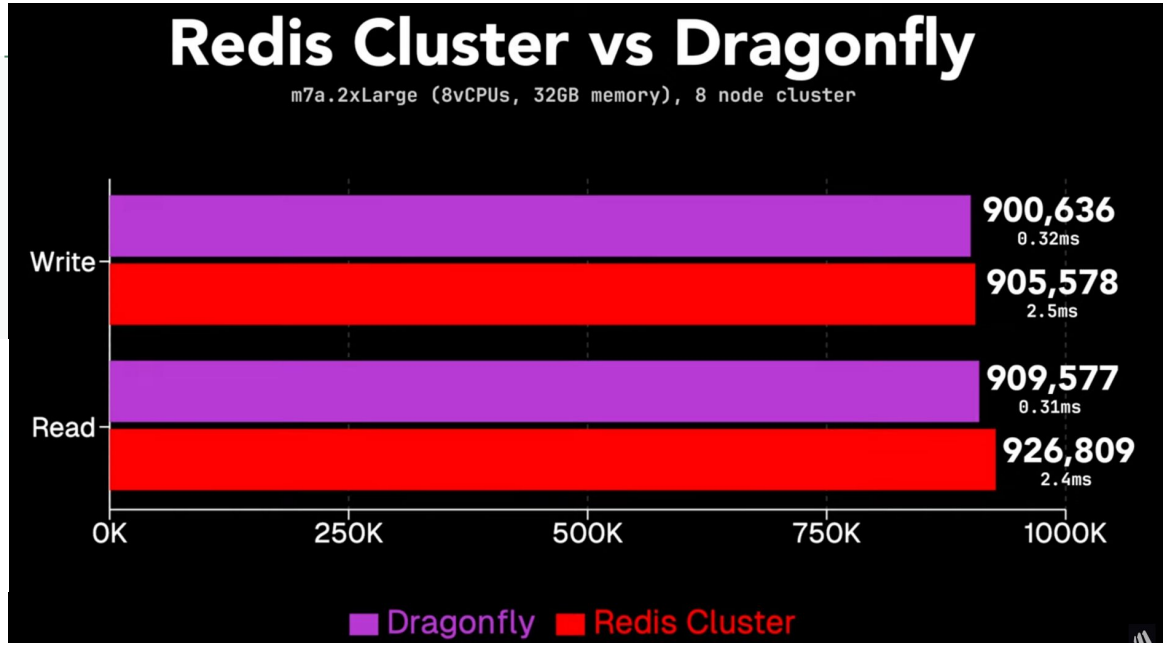
AWS › EC2 › m7a.2xlarge

m7a.2xlarge

The m7a.2xlarge instance is in the General purpose family with 8 vCPUs, 32 GiB of memory and up to 12.5 Gibps of bandwidth starting at $0.46368 per hour.

$ Pricing

$0.464	$0.194	$0.307	$0.21
On Demand	Spot	1-Year Reserved	3-Year Reserved



Chapter 5: What I Chose for Pronunciation Professor and Why

TL;DR

- Single instance *can* handle larger workloads more quickly
- Drop-in substitute (just change a Docker image and the rest is basically identical)
- Smaller, hungrier company
- More generous licensing, closer to FOSS than Redis (arguable)

Chapter 6: Demo

Pronunciation Professor uses DragonflyDB for:

- Leaderboard
- Rate limiting
- Caching

Chapter 6: Demo

Redis Commands:

- GET
- INCR
- EXPIRE
- SETEX
- DELETE
- ZINCRBY
- HSET
- ZREVRANGE (WITHSCORES)
- HGET
- SCAN
- ZADD
- KEYS

Thank You!

- https://LinkedIn.com/in/Bill-Sendewicz
- PronunciationProfessor.com
- Feedback form: https://forms.gle/X7xTmjL8VTkbBWzu9