

Alternatives to Redis

Bill Sendewicz
PronunciationProfessor.com
EmergencyContact.info

ThaiPy
November 2025

Table of Contents

1

What is Redis?



2

Why Use Redis?



3

Redis Shortcomings



4

Alternatives to Redis



5

**What I Chose for Pronunciation Professor
and Why**



6

Quick Demo & Conclusion



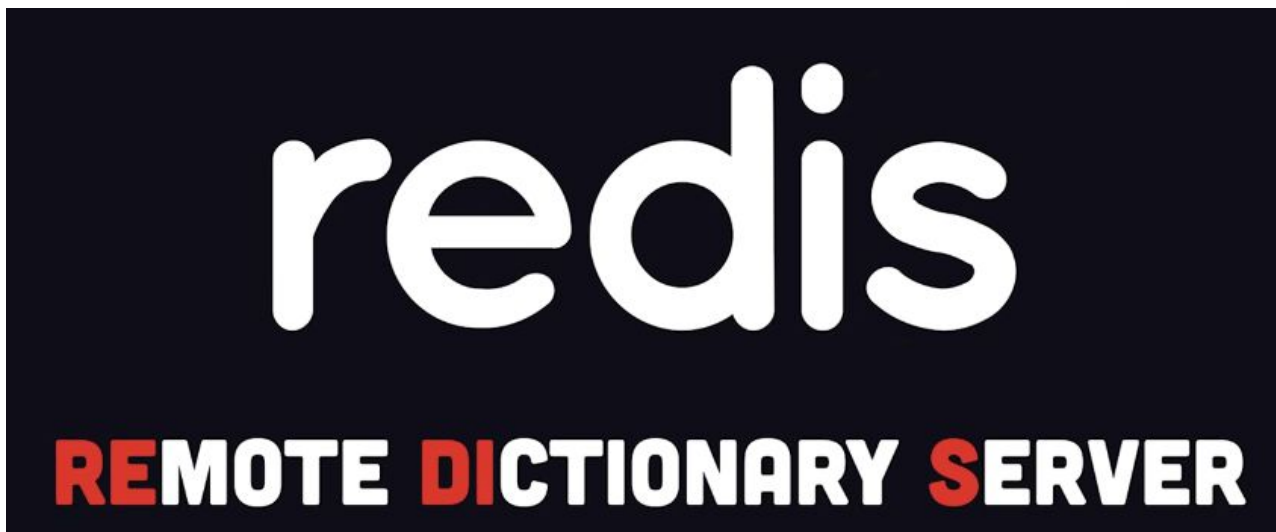
Feedback Form

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

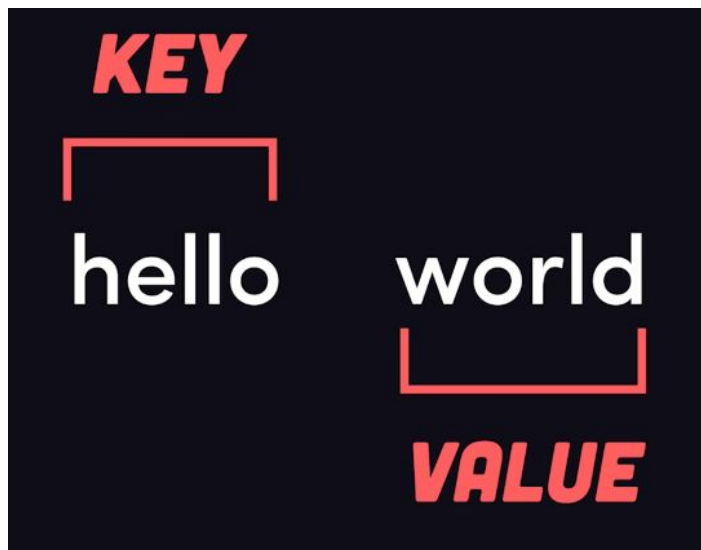


Chapter 1: What is Redis?

Chapter 1: What is Redis?



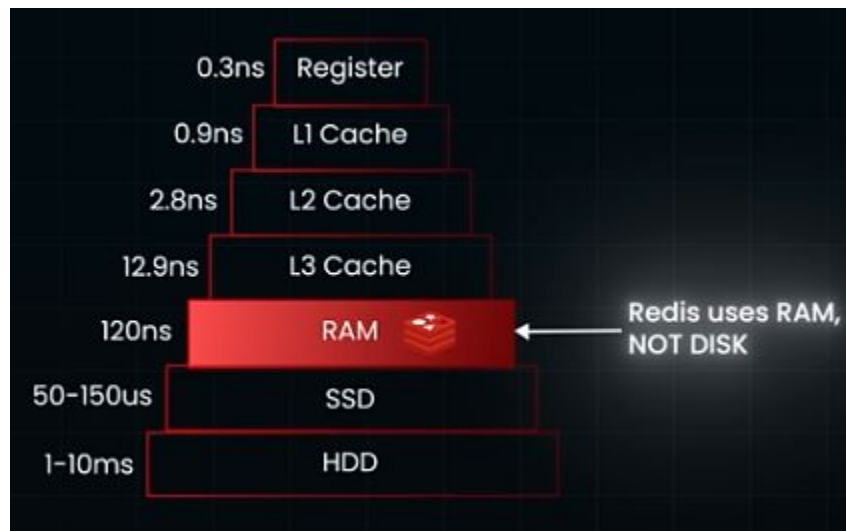
Chapter 1: What is Redis?



Chapter 1: What is Redis?

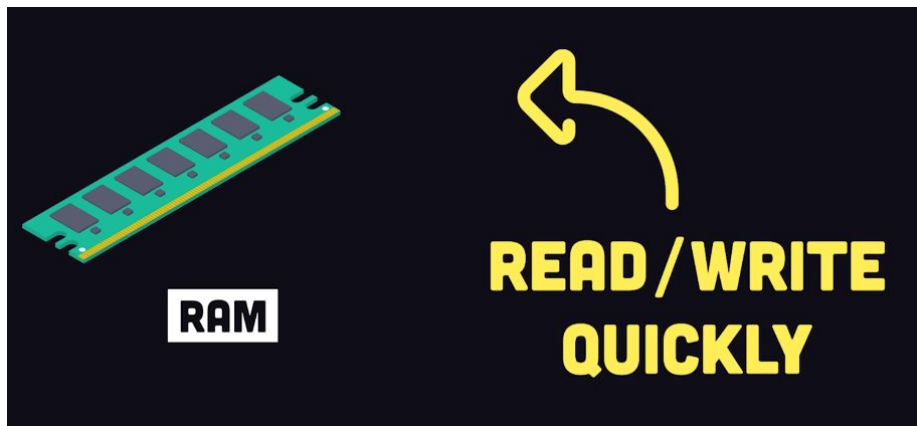
KEY	hello world	STRING
	011011010110111101101101	BITMAP
	{23334}{6634728}{916}	BITFIELD
	{ a: "hello", b: "world" }	HASH
	[A > B > C > C]	LIST
	{ A, B, C }	SET
	{ A: 1, B: 2, C: 3 }	SORTED SET
	{ A: (50.1, 0,5) }	GEOSPATIAL
	01101101 01101111 01101101	HYPERLOG
	{ id1=time1.seq({ a: "foo", a: "bar" }) }	STREAM

Chapter 1: What is Redis?



over 400x faster

Chapter 1: What is Redis?



Fast

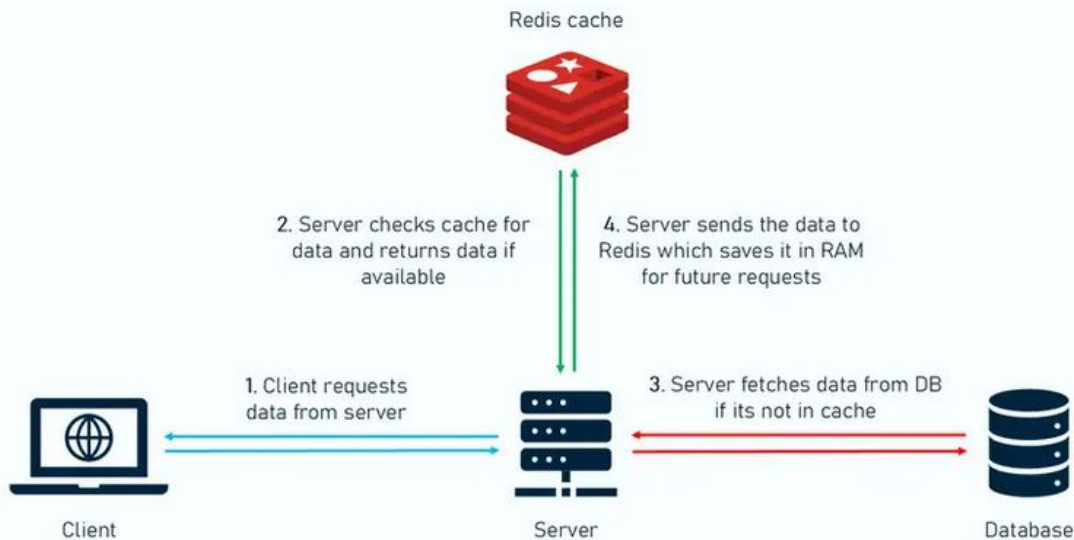


Slow

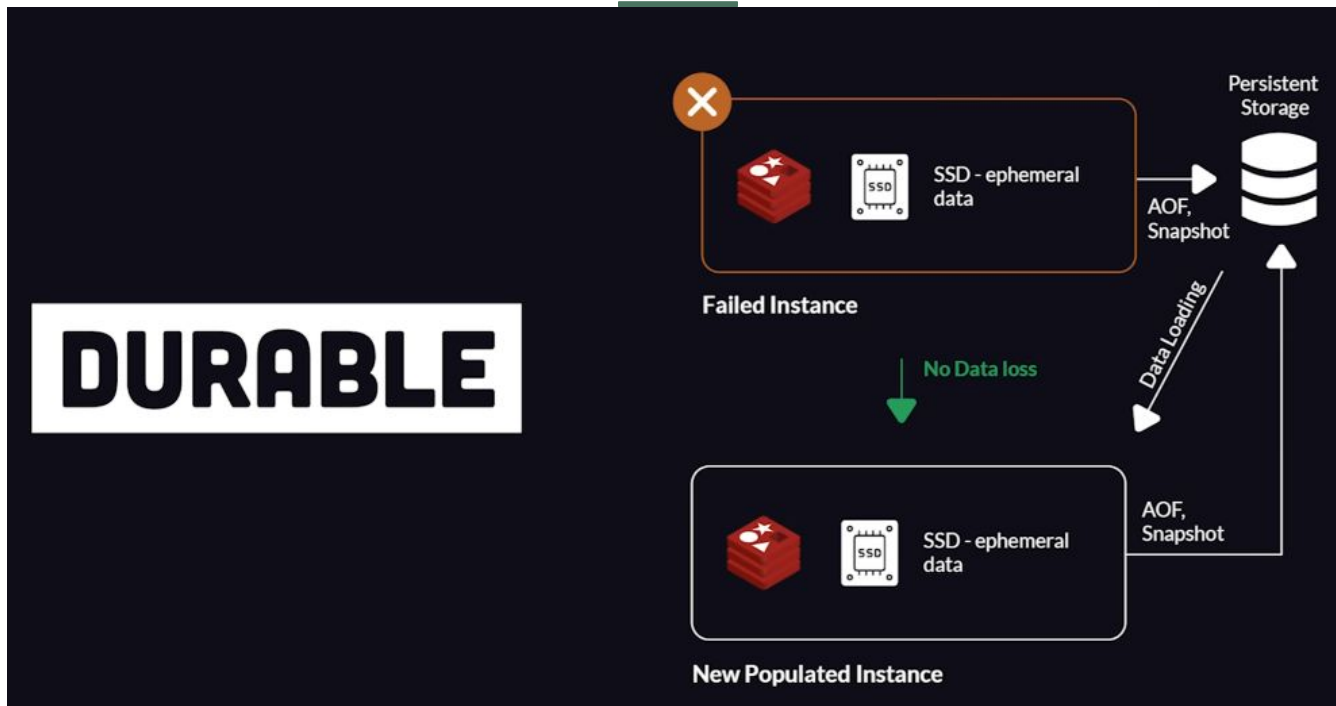
Chapter 2: Why Use Redis?

Chapter 2: Why Use Redis?

HOW REDIS WORKS



Chapter 2: Why Use Redis?



Chapter 2: Why Use Redis?

Top 5 Uses of Redis

String	 Distributed Lock	Hash	 Shopping Cart
	 Cache		
	 Session	Bitmap	 Shopping Cart
Int	 Global ID		
	 Rate Limiter	List	 Message Queue
	 Counter	ZSet	 Rank/Leaderboard

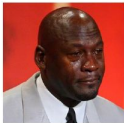
Chapter 3: Redis Shortcomings

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



<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

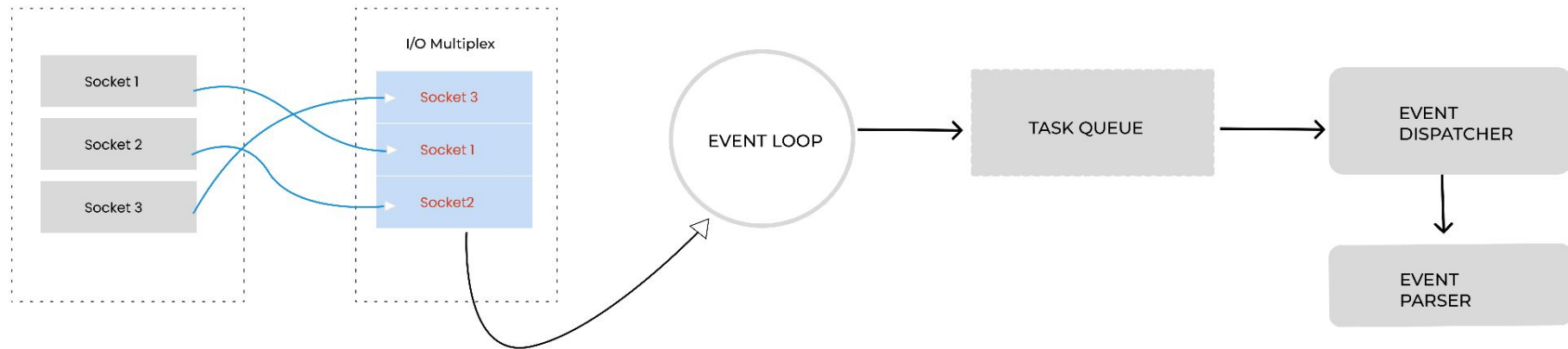
“I quite my full time job to...”

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

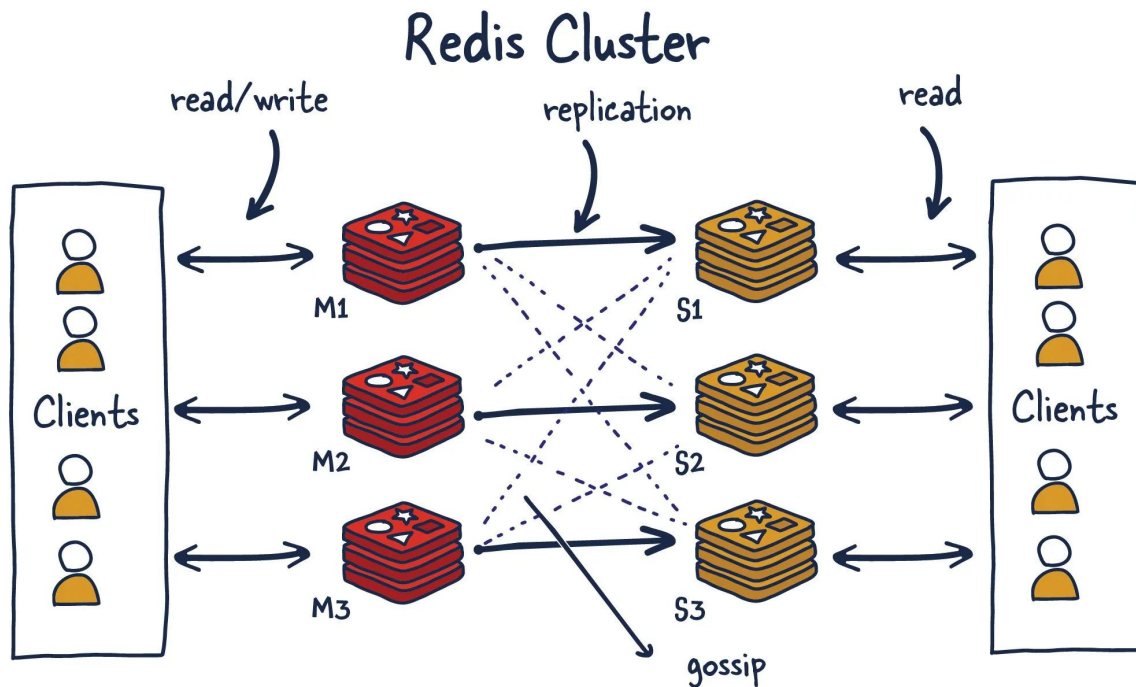
- CPU-intensive tasks
 - Redis processes commands sequentially
 - High latency when other requests wait for intensive tasks to finish
 - **Cannot use multiple cores for heavy computations because Redis is single-threaded**

Chapter 3: Redis Shortcomings

Solution?

- Redis cluster

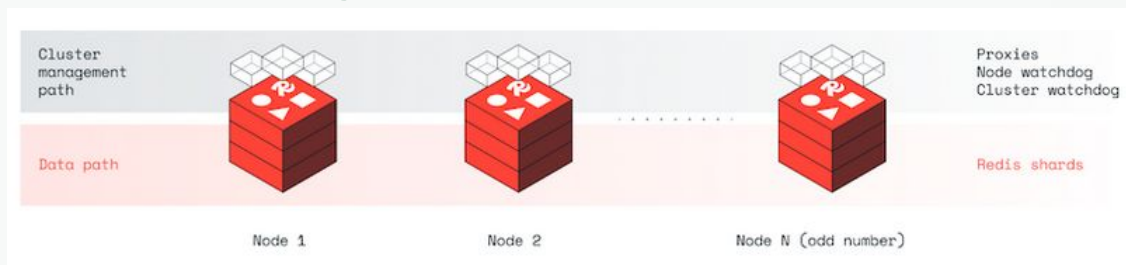
Chapter 3: Redis Shortcomings



Chapter 3: Redis Shortcomings

Solution?

- Redis cluster
- Dataset sharding



Chapter 3: Redis Shortcomings

- Single-threaded
- License change drama

Chapter 3: Redis Shortcomings

Redis License Change: A Look at the Competitive Game between OSS and Cloud Computing Giants

👤 AutoMQ Team • 📅 March 30, 2024

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

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

Chapter 3: Redis Shortcomings

A photograph of the Grand Hyatt Erawan Bangkok building at night. The building is a large, modern structure with a prominent entrance featuring columns. The name "GRAND HYATT" is visible on the upper part of the building. The image is dark, with the building's lights providing the main illumination. The text is overlaid on the image.

Why GenAI Success Requires Real-Time Data

Networking Dinner at Grand Hyatt Erawan Bangkok

Successfully held on 11th July 2024

Chapter 3: Redis Shortcomings

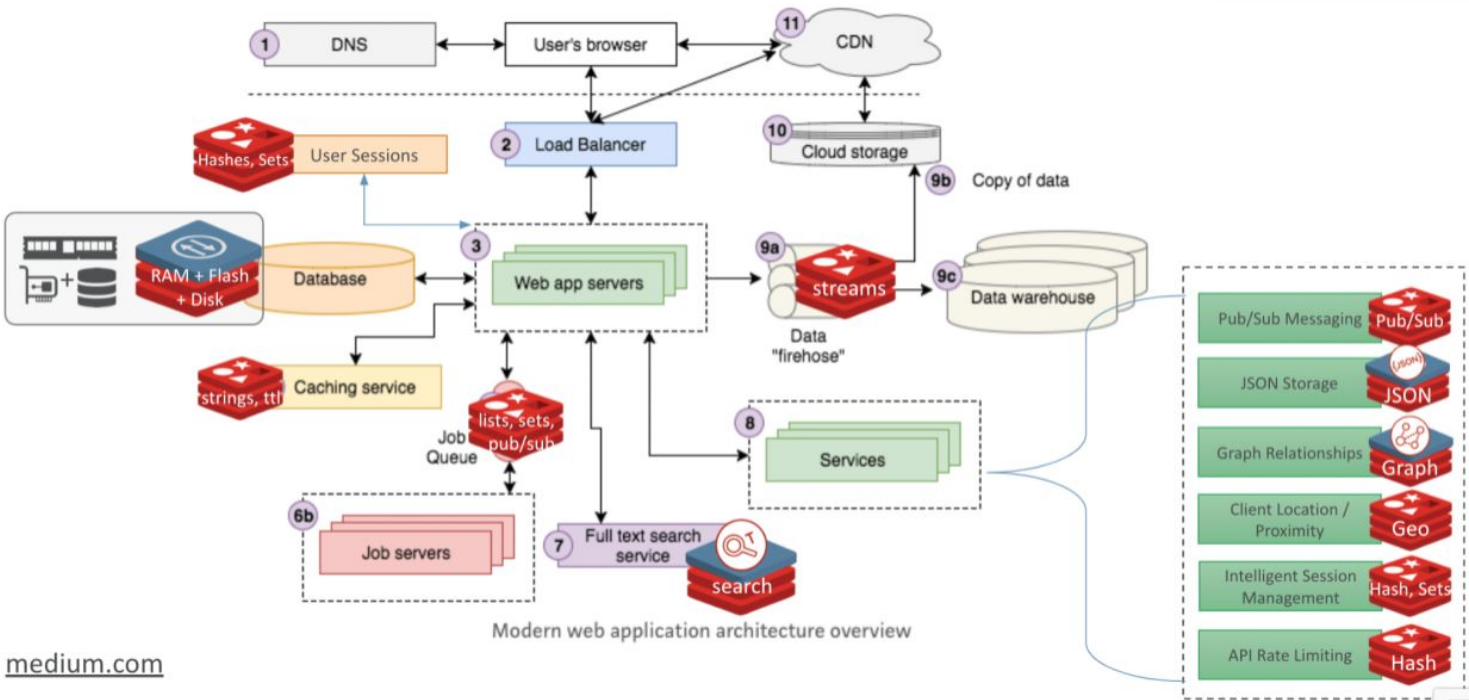


Chapter 3: Redis Shortcomings

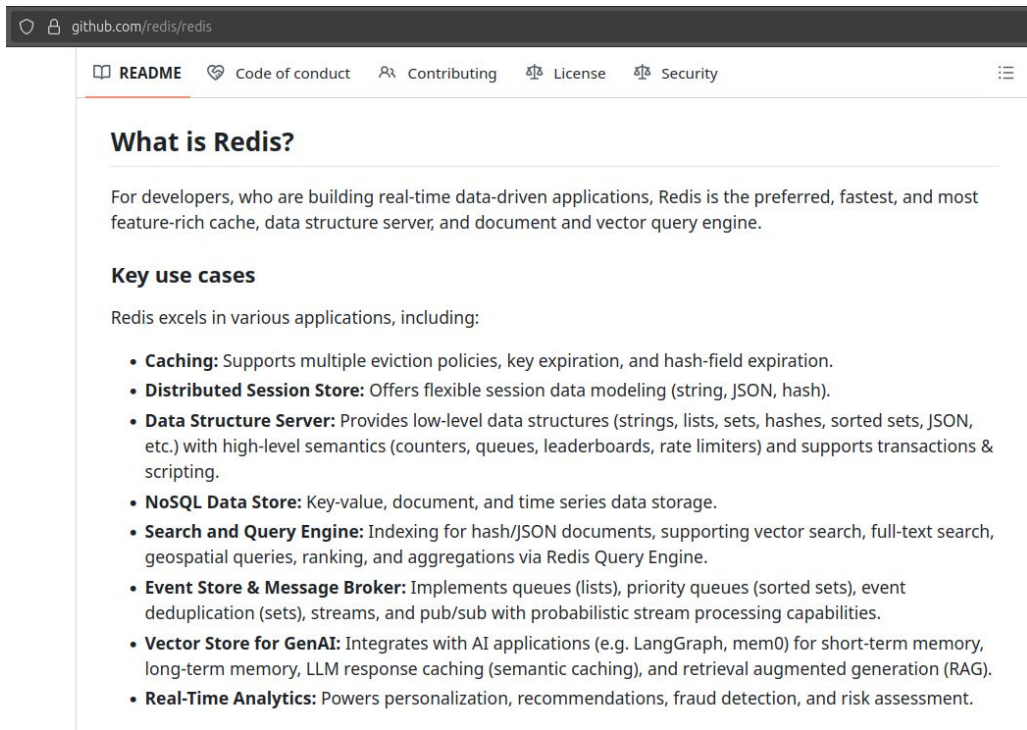
Chapter 3: Redis Shortcomings

- 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



The screenshot shows the GitHub repository page for Redis. The header includes the repository name 'github.com/redis/redis' and navigation links for README, Code of conduct, Contributing, License, and Security. The main content area is titled 'What is Redis?' and describes Redis as a preferred, fast, and feature-rich cache, data structure server, and document and vector query engine. It then lists 'Key use cases' for Redis, including caching, distributed session store, data structure server, NoSQL data store, search and query engine, event store & message broker, vector store for GenAI, and real-time analytics.

github.com/redis/redis

[README](#) [Code of conduct](#) [Contributing](#) [License](#) [Security](#)

What is Redis?

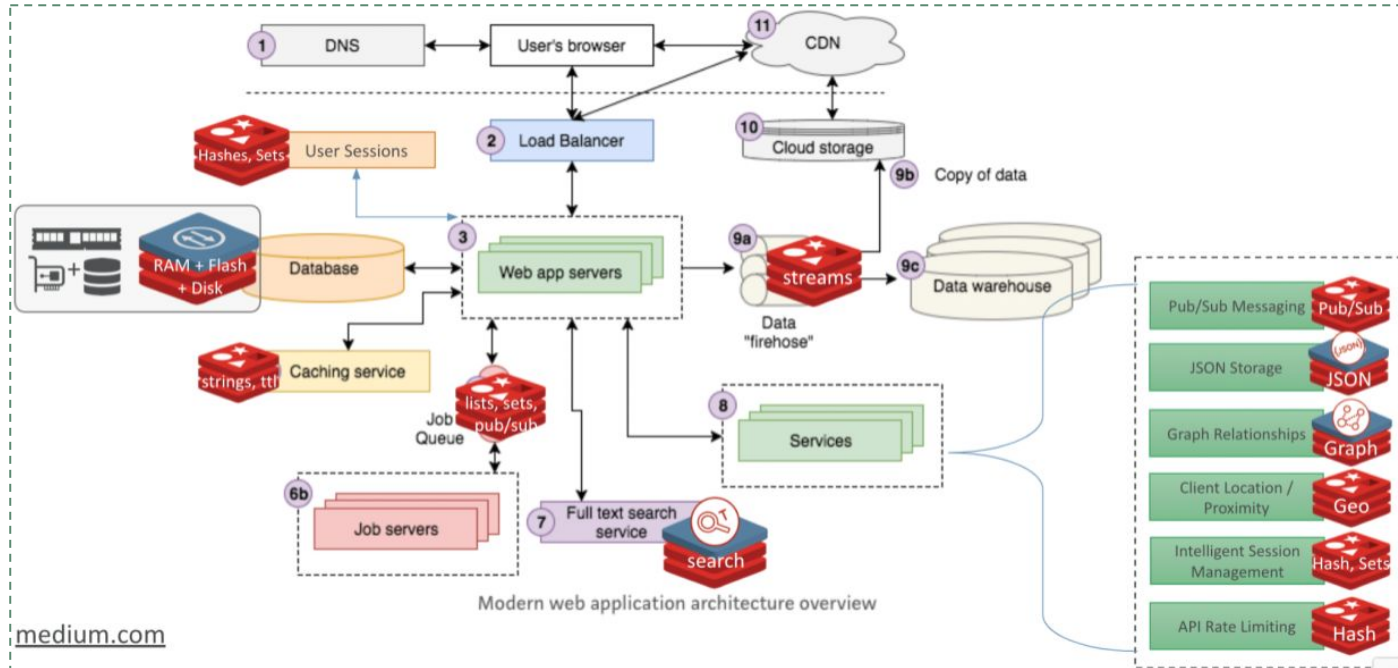
For developers, who are building real-time data-driven applications, Redis is the preferred, fastest, and most feature-rich cache, data structure server, and document and vector query engine.

Key use cases

Redis excels in various applications, including:

- **Caching:** Supports multiple eviction policies, key expiration, and hash-field expiration.
- **Distributed Session Store:** Offers flexible session data modeling (string, JSON, hash).
- **Data Structure Server:** Provides low-level data structures (strings, lists, sets, hashes, sorted sets, JSON, etc.) with high-level semantics (counters, queues, leaderboards, rate limiters) and supports transactions & scripting.
- **NoSQL Data Store:** Key-value, document, and time series data storage.
- **Search and Query Engine:** Indexing for hash/JSON documents, supporting vector search, full-text search, geospatial queries, ranking, and aggregations via Redis Query Engine.
- **Event Store & Message Broker:** Implements queues (lists), priority queues (sorted sets), event deduplication (sets), streams, and pub/sub with probabilistic stream processing capabilities.
- **Vector Store for GenAI:** Integrates with AI applications (e.g. LangGraph, mem0) for short-term memory, long-term memory, LLM response caching (semantic caching), and retrieval augmented generation (RAG).
- **Real-Time Analytics:** Powers personalization, recommendations, fraud detection, and risk assessment.

Chapter 3: Redis Shortcomings





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

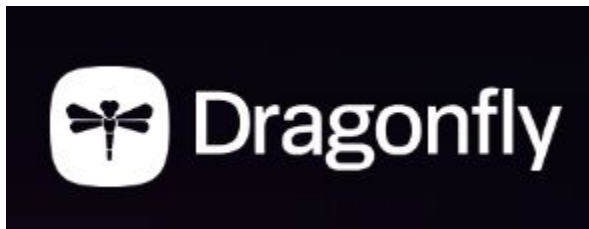
Chapter 4: **Alternatives to Redis**

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

Choose How You Deploy

Pick the deployment option that fits your team's needs and infrastructure requirements.

	Community	Cloud - Business	Cloud - Enterprise
Access	Free to use, modify, and distribute	Managed service with pay-as-you-go pricing	Custom licensing and SLA agreements
Deployment	Self-hosted on your infrastructure	Fully managed in our cloud	Fully managed in your cloud
Support & Community	Community support via GitHub	Support from the Dragonfly Team	Dedicated support team
Updates	Manual updates and patches	Automatic updates and maintenance	Priority updates and custom patches
Best For	Developers and small teams	Production applications	Large organizations
	Get Started	Try Cloud	Contact Sales

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 Gbps 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

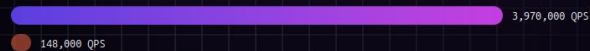
Boost Performance by 25x

Dragonfly's multi-threaded architecture is optimized to harness the power of modern cloud hardware in order to supercharge performance. Dragonfly delivers more throughput on less hardware with lower latency.

25x

More QPS than Redis

Throughput (QPS)



12x

Faster snapshotting than Redis

Snapshotting Speed (MB/s)



QPS benchmark on AWS c6gn.16xlarge.
Snapshot benchmark on AWS c6gn.4xlarge.
[Source](#).

Dragonfly Redis

Chapter 5: What I Chose for Pronunciation Professor and Why

Benchmarking

AWS > EC2 > m7a.2xlarge

m7a.2xlarge

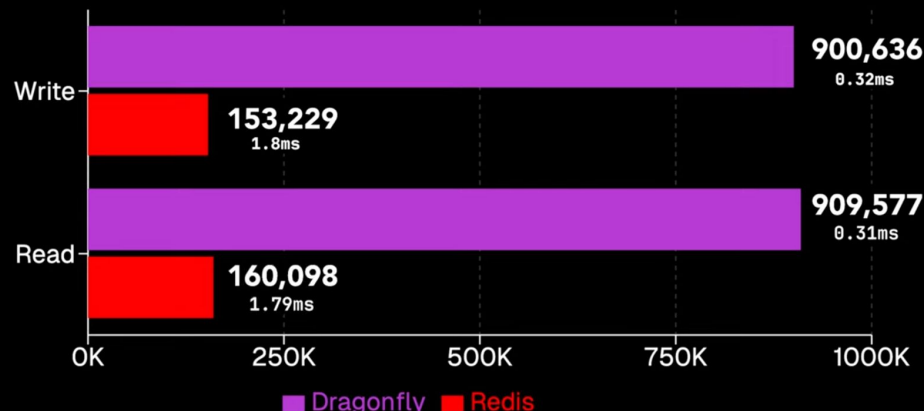
The m7a.2xlarge instance is in the General purpose family with 8 vCPUs, 32 GiB of memory and up to 12.5 Gbps of bandwidth starting at \$0.46368 per hour.

\$ Pricing

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

Redis vs Dragonfly

m7a.2xLarge (8vCPUs, 32GB memory)



ic-attraction) Tab #1

Redis Server

127.0.0.1:6379> info memory

used_memory:3573774880

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

dev@ip-172-31-43-2: -

dev@ip-172-31-43-2:~\$

DragonflyDB

127.0.0.1:6379> debug populate 50000000 key 550

(4.53s)

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

committed_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-172-31-45-78: -

Usage of /: 37.2% of 15.33GB Users logged in: 0

Memory usage: 0% IPv4 address for docker0: 172.17.0.1

Swap usage: 0% 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-172-31-45-78:~\$

dev@ip-172-31-43-2: -

Mem[|||||100.0%] 4[|||||0.0%] 0[|||||0.0%] 12[|||||0.7%]

1[|||||0.7%] 5[|||||0.0%] 9[|||||0.7%] 13[|||||0.0%]

2[|||||0.0%] 6[|||||0.0%] 10[|||||1.4%] 14[|||||0.0%]

3[|||||0.0%] 7[|||||0.0%] 11[|||||0.0%] 15[|||||0.0%]

Mem[|||||137.0G/61.4G] Tasks: 53; 2 running

Swap[|||||OK/OK] Load average: 0.54 0.31 0.18

Uptime: 00:00:57

dev@ip-172-31-45-78: -

Mem[|||||67.6%] 4[|||||65.9%] 8[|||||65.8%] 12[|||||62.8%]

1[|||||81.9%] 5[|||||81.9%] 9[|||||81.9%] 13[|||||64.7%]

2[|||||72.5%] 6[|||||64.6%] 10[|||||81.9%] 14[|||||81.9%]

3[|||||81.9%] 7[|||||81.9%] 11[|||||81.9%] 15[|||||58.3%]

Mem[|||||27.1G/61.4G] Tasks: 41; 1 running

Swap[|||||OK/OK] Load average: 1.31 0.47 0.22

Uptime: 00:00:56

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	8	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

F1Help F2Setup F3Search F4Filter F5Free F6SortBy F7Nice F8I/O F9Kill F10Quit

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
2079	dev	20	0	28.5G	27.2G	7808	S	1278	43.3	2:14.28	/dragonfly --alsologtostderr --dir /mnt/vol1
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 F5Free F6SortBy F7Nice F8I/O F9Kill F10Quit

Ctrl + LOCK

Tab

Resize

Move

Search

Session

Quit

Tip: Alt + <n> => open new pane. Alt + <+/-> => Alt + <h> => navigate between panes. Alt + <+/-> => increase/decrease pane size.

Chapter 5: What I Chose for Pronunciation Professor and Why

Benchmarking

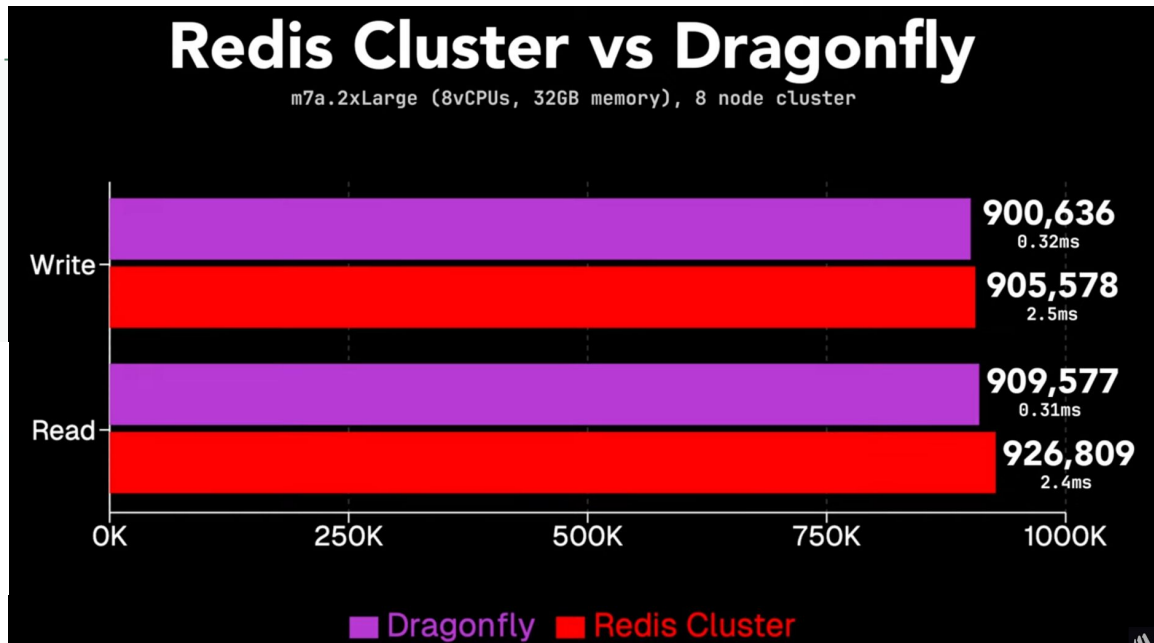
AWS > EC2 > m7a.2xlarge

m7a.2xlarge

The m7a.2xlarge instance is in the General purpose family with 8 vCPUs, 32 GiB of memory and up to 12.5 Gbps 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>

