

> simple ways to jumpstart a **performance** culture



> why jumpstart <

> why jumpstart < security?

> why jumpstart < compliance?

> why jumpstart < performance?

> **non functional** <

> \$1.56
trillion/year *
why <

> unlocks opportunities ^[1] <

[1] <https://www-file.huawei.com/-/media/CORPORATE/PDF/mbb/5g-unlocks-a-world-of-opportunities-v4.pdf?la=en>

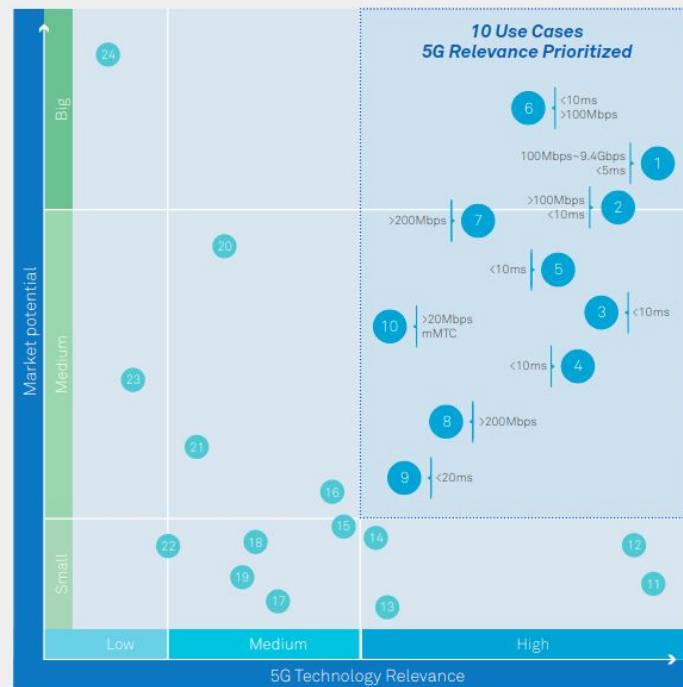
> unlocks opportunities

[1]

Index Definitions

- 1. Cloud Virtual & Augmented Reality** – Real-time Computer Rendering Gaming/Modeling
- 2. Connected Automotive** – ToD, Platooning, Autonomous Driving
- 3. Smart Manufacturing** – Cloud Based Wireless Robot Control
- 4. Connected Energy** – Feeder Automation
- 5. Wireless eHealth** – Remote Diagnosis With Force-Feedback
- 6. Wireless Home Entertainment** – UHD 8K Video & Cloud Gaming
- 7. Connected Drones** – Professional Inspection & Security
- 8. Social Networks** – UHD/Panoramic Live Broadcasting
- 9. Personal AI Assistant** – AI Assisted Smart Helmet
- 10. Smart City** – AI-enabled Video Surveillance

TOP TEN 5G USE CASES



[1] <https://www-file.huawei.com/-/media/CORPORATE/PDF/mbb/5g-unlocks-a-world-of-opportunities-v4.pdf?la=en>

“we ship fast”...

...“unnecessary”

> why now? <

> why not PROD? <



> cost of bug

DEV vs PROD

100X



> foster a culture <

> dev lifecycle <

> step 0 <

> **measure it** <

> if you can't measure <
you can't improve it

> describe state <

> step 1 <

> set clear goals <

> dev lifecycle <

1) setting perf goals

- At 3.3 seconds **page load time**, **conversion rate** was 1.5%
- At 4.2 seconds **page load time**, **conversion rate** was less than 1%
- At 5.7+ seconds **page load time**, **conversion rate** was 0.6%

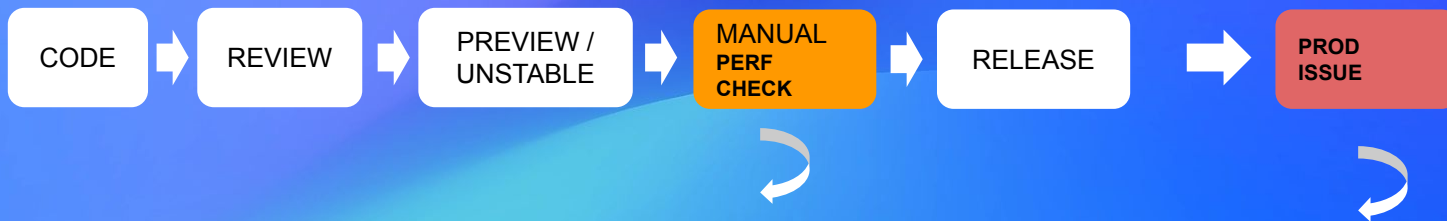
> 75% users <

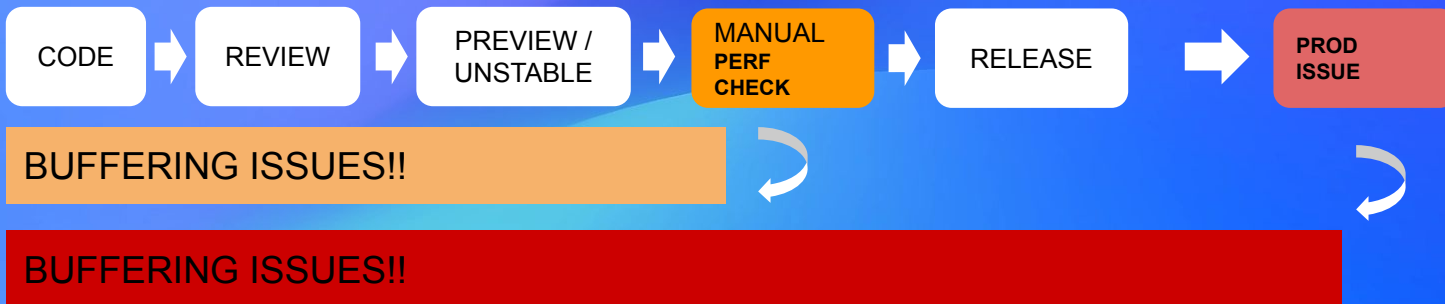
PLT < 3.5 secs

> step 2 <

> ongoing feedback <

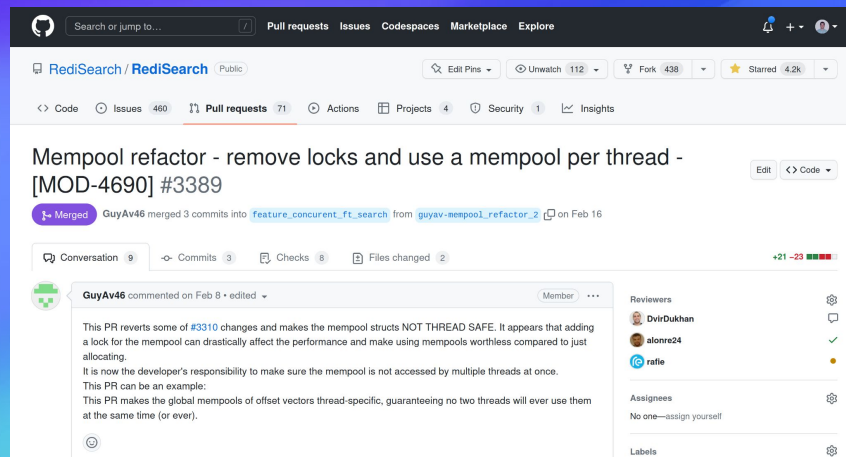
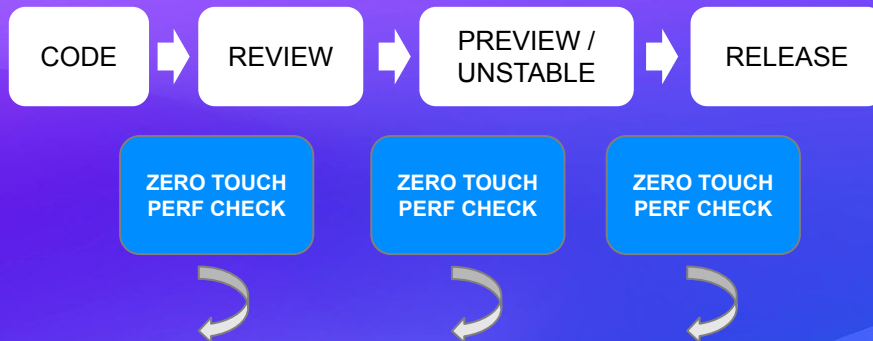






> **make it simple** <

> **as soon as possible** <






GuyAv46 added the `action:run-benchmark` label on Feb 8



GuyAv46 added the `action:run-benchmark` label on Feb 8

 **codecov** [bot] commented on Feb 8 • edited


Codecov Report


Base: **77.56%** // Head: **82.32%** // Increases project coverage by **+4.75%** 🎉


Coverage data is based on head ([933427d](#)) compared to base ([198afe0](#)).
Patch coverage: 18.84% of modified lines in pull request are covered.

! Current head [933427d](#) differs from pull request most recent head [2d8ddd9](#). Consider uploading reports for the commit [2d8ddd9](#) to get more accurate results

► Additional details and impacted files

 [View full report at Codecov.](#)

 Do you have feedback about the report comment? [Let us know in this issue.](#)





GuyAv46 added the **action:run-benchmark** label on Feb 8

codecov bot commented on Feb 8 • edited

Codecov Report

Base: 77.56% // Head: 82.32% // Increases project coverage by +4.75%

Coverage data is based on head (933427d) compared to base (198af0e).
Patch coverage: 18.84% of modified lines in pull request are covered.

Current head 933427d differs from pull request most recent head 2d8ddd9. Consider uploading reports for the commit 2d8ddd9 to get more accurate results

Additional details and impacted files

[View full report at Codecov.](#)

Do you have feedback about the report comment? [Let us know in this issue.](#)

filpecosta90 commented on Feb 8 • edited

Automated performance analysis summary

This comment was automatically generated given there is performance data available.

In summary:

- Detected a total of 11 stable tests between versions.
- Detected a total of 6 highly unstable benchmarks.
- Detected a total of 2 improvements above the improvement water line.

You can check a comparison in detail via the [grafana link](#)

Comparison between master and guyav-mempool_refactor_2.

Time Period from 30 days ago. (environment used: oss-standalone)

Test Case	Baseline master (median obs. +- std.dev)	Comparison guyav- mempool_refactor_2 (median obs. +- std.dev)	% change (higher- better)	Note
ftsb-10K-enwiki_abstract-hashes-term-prefix	8180 +- 7.8% (7 datapoints)	8056 +- nan% (1 datapoints)	-1.5%	waterline=7.8%. -- no change --
ftsb-10K-enwiki_abstract-hashes-term-suffix	2045 +- 0.9% (7 datapoints)	2187 +- nan% (1 datapoints)	6.9%	IMPROVEMENT
ftsb-10K-enwiki_abstract-hashes-term-suffix-withsuffixtrie	78063 +- 7.0% (7 datapoints)	75429 +- nan% (1 datapoints)	-3.4%	waterline=7.0%. potential REGRESSION
ftsb-10K-enwiki_abstract-hashes-term-wildcard	13479 +- 9.3% (7 datapoints)	13233 +- nan% (1 datapoints)	-1.8%	waterline=9.3%. -- no change --
ftsb-10K-multivalue-numeric-json	826 +- 2.5% (7 datapoints)	837 +- nan% (1 datapoints)	1.3%	-- no change --
ftsb-10K-singlevalue-numeric-json	368 +- 1.3% (7 datapoints)	367 +- nan% (1 datapoints)	-0.2%	-- no change --

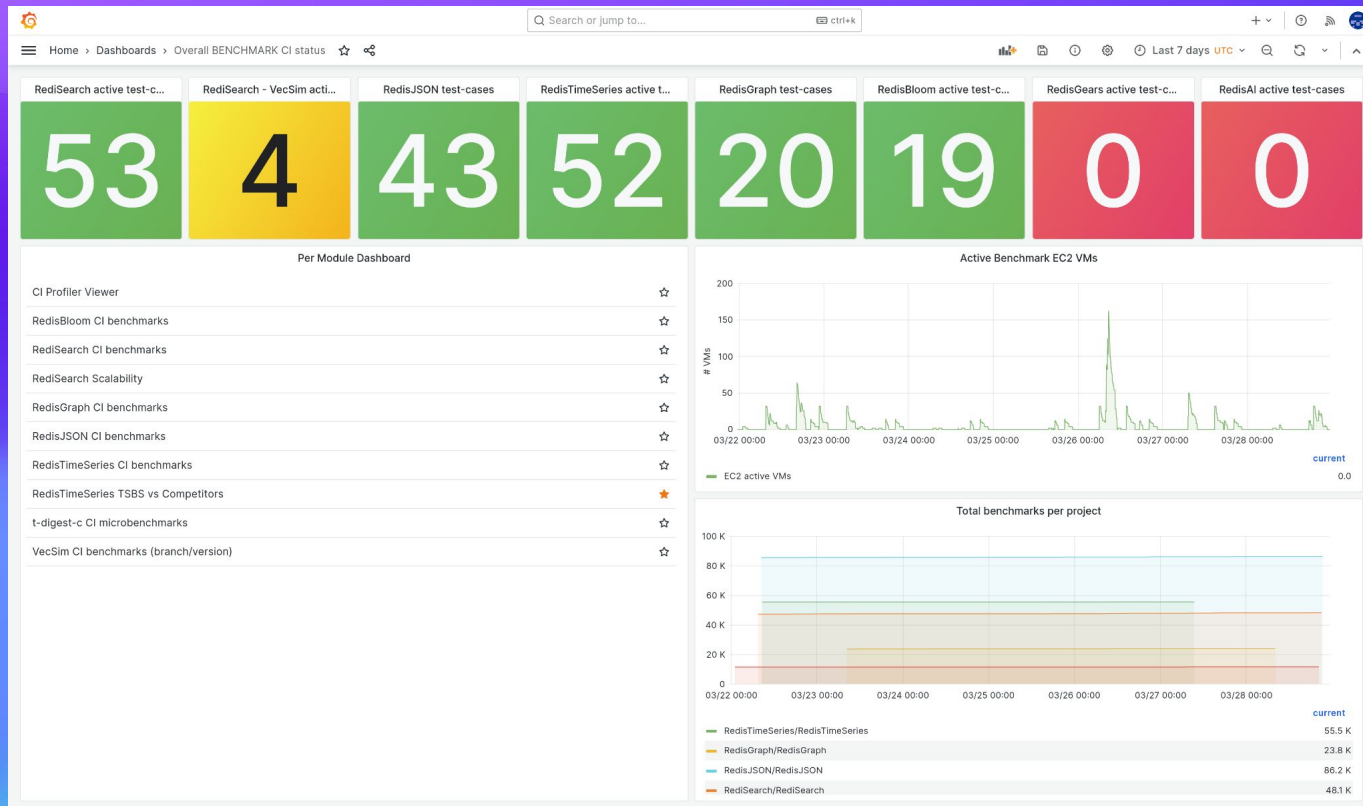
> step 3 <

continuous improvement mindset

Why?



How?

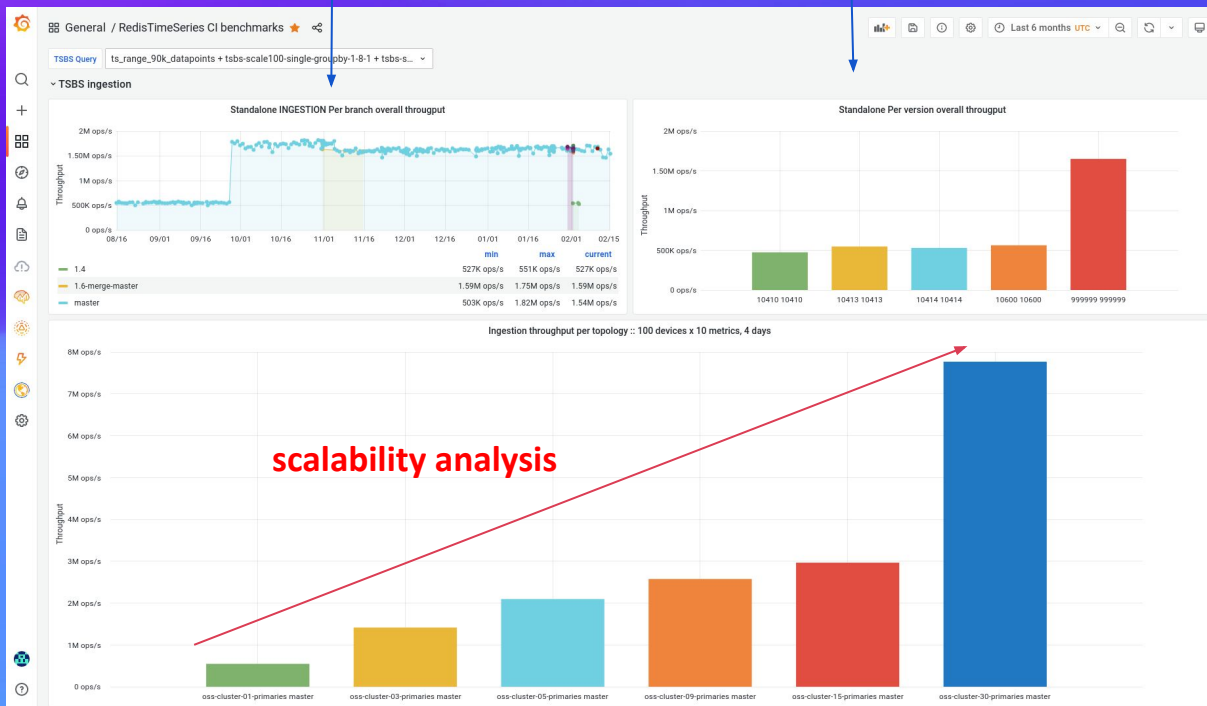


- summary dashboard for Redis Ltd Performance CI tracking -

How?

by branch

by version



How?

```
~/redislabs/RedisTimeSeries$ make benchmark
Effective log level set to INFO
2022-02-15 10:12:20,104 INFO Using: redisbench-admin 0.6.17
2022-02-15 10:12:20,118 INFO Retrieved the following local info:
2022-02-15 10:12:20,118 INFO   github_actor: filipecosta90
2022-02-15 10:12:20,118 INFO   github_org: RedisTimeSeries
2022-02-15 10:12:20,118 INFO   github_repo: RedisTimeSeries
2022-02-15 10:12:20,118 INFO   github_branch: perf_use_gnu11
2022-02-15 10:12:20,119 INFO   github_sha: 7eebb1c543c057ccaf87696ca623f451aad1ec58
2022-02-15 10:12:20,119 INFO Using the following modules ['/home/fco/redislabs
/RedisTimeSeries/bin/linux-x64-release/redistimeseries.so']
(...)
```

> step 4 <

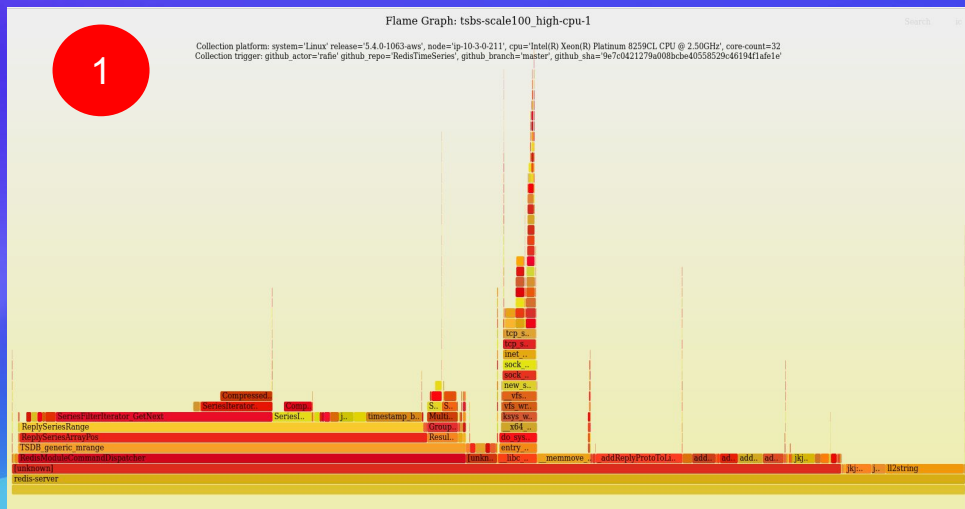
> Mentoring and Guidance <

> Avoid information silos <

> explain the why's <

Our example?

1. Full process Flame Graph + main thread Flame Graph
2. perf report per dso
3. perf report per dso,sym (w/wout callgraph)
4. perf report per dso,sym,srcline (w/wout callgraph)
5. identical stacks collapsed
6. hotpatch callgraph



Our example?

1. Full process Flame Graph + main thread Flame Graph
2. perf report per dso
3. perf report per dso,sym (w/wout callgraph)
4. perf report per dso,sym,srcline (w/wout callgraph)
5. identical stacks collapsed
6. hotpatch callgraph

2

```
49.00%  redistimeseries.so
39.62%  redis-server
 6.38%  libc-2.27.so
 4.61%  [kernel.kallsyms]
```

Our example?

1. Full process Flame Graph + main thread Flame Graph
2. perf report per dso
3. perf report per dso,sym (w/wout callgraph)
4. perf report per dso,sym,srcline (w/wout callgraph)
5. identical stacks collapsed
6. hotpatch callgraph

3

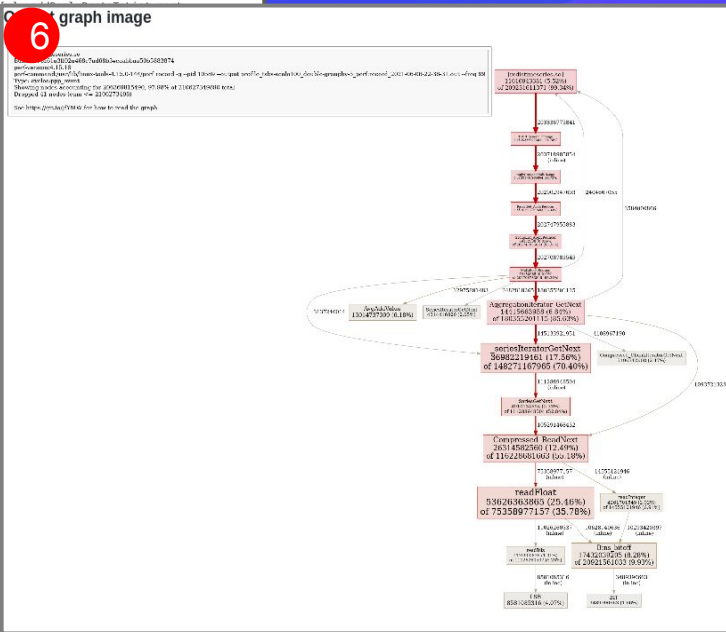
```
14.72% redistimeseries.so [.] SeriesFilterIterator_GetNext
10.39% redistimeseries.so [.] Compressed_ChunkIteratorGetNext
10.16% redis-server [.] addReplyProtoToList.part.0
9.54% redis-server [.] ll2string
5.84% redistimeseries.so [.] timestamp_binary_search
5.36% libc-2.27.so [.] __memmove_avx_unaligned_erms
4.75% redistimeseries.so [.] jkj::dragonbox::to_chars_detail::to_chars<double, jkj::dragonbox::default_float_traits<double> >
3.78% redistimeseries.so [.] SeriesIteratorGetNext
3.35% redistimeseries.so [.] jkj::dragonbox::detail::impl<double, jkj::dragonbox::default_float_traits<double> >::compute_nearby
1.56% jkj::dragonbox::detail::policy_impl::decimal_to_binary_rounding::interval_type::symmetric_boundary, jkj::dragonbox::detail::policy_imp
1.56% jkj::dragonbox::detail::policy_impl::binary_to_decimal_rounding::to_even, jkj::dragonbox::detail::policy_impl::cache::full, bool>
3.12% redis-server [.] addReplyProto
2.79% redis-server [.] addReplyProto.part.0
2.48% redis-server [.] addReplyLongLongWithPrefix
2.43% redis-server [.] addReply.part.0
1.56% redistimeseries.so [.] jkj::dragonbox::to_chars_n<double, jkj::dragonbox::default_float_traits<double> >
1.37% redistimeseries.so [.] ReplyWithSample
1.28% redistimeseries.so [.] Uncompressed_UpsertSample
1.09% [kernel.kallsyms] [k] copy_user_enhanced_fast_string
1.04% redis-server [.] RM_ReplyWithSimpleString
1.02% redis-server [.] addReply
```


Our example?

1. Full process Flame Graph + main thread Flame Graph
2. perf report per dso
3. perf report per dso,sym (w/wout callgraph)
4. perf report per dso,sym,srcline (w/wout callgraph)
5. identical stacks collapsed
6. hotpatch callgraph

4

```
12.47% redistimeseries.so [...]
SeriesFilterIterator::getNext
filter_iterator.c:16
4.75% redistimeseries.so [...] jkj::dragonbox::to_chars_detail::to_ch
>
0
3.35% redistimeseries.so [...] jkj::dragonbox::detail::impl_double,
kj::dragonbox::detail::policy_impl::decimal_to_binary_rounding::interval,
kj::dragonbox::detail::policy_impl::binary_to_decimal_rounding::to_even,
3.24% redis-server [...]
ll2string
util.c:354
2.67% redis-server [...]
ll2string
util.c:352
2.41% redistimeseries.so [...]
timestamp_binary_search
generic_chunk.c:180
2.08% redis-server
0
networking.c:326
2.01% redis-server
0
networking.c:327
1.89% redis-server
ll2string
util.c:353
1.56% redistimeseries.so
kj::dragonbox::default_float_tr
0
1.46% redistimeseries.so
Compressed ChunkIterator::getNext
gorilla.c:221
1.32% redis-server
0
networking.c:312
1.23% redistimeseries.so
Compressed ChunkIterator::getNext
gorilla.c:511
1.16% redistimeseries.so
timestamp_binary_search
generic_chunk.c:173
1.09% redis-server
0
sds.h:89
1.06% redistimeseries.so
SeriesFilterIterator::getNext
filter_iterator.c:24
1.04% [kernel.kallsyms]
copy_user_enhanced_fast_string
copy_user_enhanced_fast_string+1
```



> gains? <

improved Redis performance by up to 4x!^[1]

[1] - <https://redis.com/blog/redis-intel-performance-testing/>
<https://redis.com/blog/redis-7-geographic-commands/>

Filipe Oliveira
performance <at> redis <dot> com

> predictable <

> sustainable <

> scalable <

> stress free <

> **challenging** <



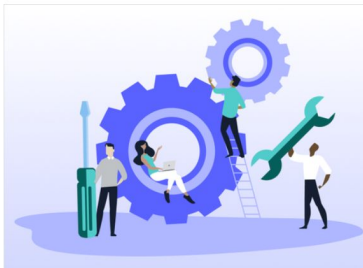
redis.com



February 21, 2023

Speeding Up Geographic Commands in Redis 7

[Learn More](#) →



February 1, 2023

Optimizing Redis' Default Compiler Flags

[Learn More](#) →



November 17, 2022

Introducing the Redis/Intel Benchmarks Specification for Performance Testing, Profiling, and Analysis

[Learn More](#) →



June 28, 2022

13 Years Later – Does Redis Need a New Architecture?

[Learn More](#) →



June 15, 2022

Making the Fast, Faster! Methodically Improving Redis Performance

[Learn More](#) →



Follow up

- our blogs
- email: performance <at> redis <dot> com
- github orgs:
 - <https://github.com/redis-performance>
 - <https://github.com/redis/redis-benchmarks-specification>
- happy to connect:
 - <https://www.linkedin.com/in/filipecosta90/>
 - https://twitter.com/fcosta_oliveira

thank you!

questions?

performance <at> redis <dot> com