

What's the Point of Multiple Redis Databases?

Asked 11 years, 8 months ago Modified 2 years ago Viewed 146k times



275



So, I've come to a place where I wanted to segment the data I store in redis into separate databases as I sometimes need to make use of the keys command on one specific kind of data, and wanted to separate it to make that faster.

If I segment into multiple databases, everything is still single threaded, and I still only get to use one core. If I just launch another instance of Redis on the same box, I get to use an extra core. On top of that, I can't name Redis databases, or give them any sort of more logical identifier. So, with all of that said, why/when would I ever want to use multiple Redis databases instead of just spinning up an extra instance of Redis for each extra database I want? And relatedly, why doesn't Redis try to utilize an extra core for each extra database I add? What's the advantage of being single threaded across databases?

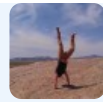
redis

Share

Improve this question

asked Apr 25, 2013 at 17:59

Follow



Eli

38.8k ● 40 ● 155 ● 212

- 1 in your Node.js app, do this ---> `module.exports = {"1":"your name for redis db one","2":"your name for redis db two","3":"your name for redis db three"}` etc, or switch the keys and values, whatever you need – [Alexander Mills](#) Sep 1, 2015 at 22:49 ✎
- 2 In Redis 2.8.0 and up it is recommended that you use SCAN instead of KEYS, because it iterates over a small number of elements at a time (thus not blocking the server for long periods of time). – [TryHarder](#) Sep 16, 2015 at 12:52

9 Answers

Sorted by:

Highest score (default)



147



You don't want to use multiple databases in a single redis instance. As you noted, multiple instances lets you take advantage of multiple cores. If you use database selection you will have to refactor when upgrading.

Monitoring and managing multiple instances is not difficult nor painful.



Indeed, you would get far better metrics on each db by segregation based on instance. Each instance would have stats reflecting that segment of data, which can allow for better tuning and more responsive and accurate monitoring. Use a recent version and separate your data by instance.

As Jonaton said, don't use the keys command. You'll find far better performance if you simply create a key index.

Whenever adding a key, add the key name to a set. The keys command is not terribly useful once you scale up since it will take significant time to return.

Let the access pattern determine how to structure your data rather than store it the way you think works and then working around how to access and mince it later. You will see far better performance and find the data consuming code often is much cleaner and simpler.

Regarding single threaded, consider that redis is designed for speed and atomicity. Sure actions modifying data in one db need not wait on another db, but what if that action is saving to the dump file, or processing transactions on slaves? At that point you start getting into the weeds of concurrency programming.

By using multiple instances you turn multi threading complexity into a simpler message passing style system.

[Share](#) [Improve this answer](#)

[Follow](#)

edited Jan 26, 2021 at 16:27



[bfontaine](#)

19.7k ● 13 ● 78 ● 120

answered Apr 26, 2013 at 19:26



[The Real Bill](#)

15.7k ● 8 ● 38 ● 40

-
- 1 If I need every key in some database, why would accessing all of those keys from a set and then iterating through be faster than the keys command? Both would be $O(n)$, but the set method would require up to double the space, and way

more back and forth and data transfer. – [Eli](#) May 7, 2013 at 19:03

71 Using multiple databases is deprecated? Can you provide a reference for that statement please. I'm aware that multiple databases aren't supported in Redis Cluster but neither are any complex multi-key commands and they're not deprecated. – [ostergaard](#) Oct 19, 2013 at 6:29

31 [Some \(strong\) evidence](#) from the 'owner' of Redis (according to Google Code) that "... databases are not going to be deprecated even if I in the past stated that they would be." – [Kenny Evitt](#) Aug 7, 2014 at 19:40

3 You won't be able to use more than one redis db on redis-cluster. Aside from that, multiple databases will still a thing. – [coredump](#) Sep 2, 2014 at 18:50

31 -1 for the deprecated statement. Multiple databases may be discouraged, and unsupported in the redis-cluster, but they are not deprecated. – [AgDude](#) Jan 21, 2015 at 14:41



In principle, Redis databases on the same instance are no different than schemas in RDBMS database instances.

140



So, with all of that said, why/when would I ever want to use multiple Redis databases instead of just spinning up an extra instance of Redis for each extra database I want?



There's one clear advantage of using redis databases in the same redis instance, and that's management. If you spin up a separate instance for each application, and let's say you've got 3 apps, that's 3 separate redis instances,

each of which will likely need a slave for HA in production, so that's 6 total instances. From a management standpoint, this gets messy real quick because you need to monitor all of them, do upgrades/patches, etc. If you don't plan on overloading redis with high I/O, a single instance with a slave is simpler and easier to manage provided it meets your SLA.

Share Improve this answer

Follow

edited Oct 6, 2022 at 10:13



jpbochi

4,396 ● 3 ● 37 ● 43

answered Apr 25, 2013 at 20:03



raffian

32k ● 26 ● 106 ● 183

37 Multiple Redis instances is always the way to go. Period. Run parallel queries for different data. If your CI/CD pipeline doesn't create cache clusters for you, fix it, rather than You get the point – Cmag Mar 31, 2016 at 3:57

7 This doesn't address the OP's points: (1) why doesn't Redis try to utilize an extra core for each extra database? (2) What's the advantage of being single threaded across databases? – ives Oct 23, 2019 at 2:11

If Redis databases are same as schemas in RDBMS, why using DBx (different from DB0) is not commonly used as usage of schemas in RDBMS ? – Ida Amit Jan 5, 2022 at 7:02

4 Another reason not to use multiple databases: redis cluster does not support multiple databases. redis.io/docs/reference/cluster-spec/#implemented-subset – shusson Mar 31, 2022 at 2:42 ✎



89



Even Salvatore Sanfilippo (creator of Redis) thinks it's a bad idea to use multiple DBs in Redis. See his comment here:

<https://groups.google.com/d/topic/redis-db/vS5wX8X4Cjg/discussion>



I understand how this can be useful, but unfortunately I consider Redis multiple database errors my worst decision in Redis design at all... without any kind of real gain, it makes the internals a lot more complex. The reality is that databases don't scale well for a number of reason, like active expire of keys and VM. If the DB selection can be performed with a string I can see this feature being used as a scalable $O(1)$ dictionary layer, that instead it is not.

With DB numbers, with a default of a few DBs, we are communication better what this feature is and how can be used I think. I hope that at some point we can drop the multiple DBs support at all, but I think it is probably too late as there is a number of people relying on this feature for their work.

Share Improve this answer

Follow

answered Apr 8, 2016 at 11:35



Nirmal

9,539 ● 11 ● 59 ● 82

12 Hold on, so using DB selection is actually less efficient than just using a prefix? Is that what this sentence here means (could someone please clarify)? "If the DB selection can be performed with a string I can see this feature being used as a scalable O(1) dictionary layer, that instead it is not."

– [davidtgq](#) Dec 14, 2018 at 6:20

2 This doesn't actually say it's a bad idea to *use* them. It says it was a bad idea for him to have included the option in the first place. – [apokryfos](#) Oct 29, 2021 at 15:27

@apokryfos He actually says that it may be considered to drop the feature, this clearly means it's a bad idea to use it

– [GregOriol](#) Jan 29, 2023 at 15:24

1 @GregOriol that is not what it says. It says : *without any kind of real gain, it makes the internals a lot more complex* so the bad idea was to implement Redis with this in it. They want to drop them to facilitate with a better implementation of Redis. They want to drop them because of internal design reasons not because it leads to bad coding practices by people who integrate Redis in their applications. Yes they argue that there is no *real gain* to be made from using them but they don't say that developers using them is a bad idea.

– [apokryfos](#) Jan 29, 2023 at 16:04 



I know this question is years old, but there's another reason multiple databases may be useful.

14



If you use a "cloud Redis" from your favourite cloud provider, you probably have a minimum memory size and will pay for what you allocate. If however your dataset is smaller than that, then you'll be wasting a bit of the allocation, and so wasting a bit of money.



Using databases you could use the same Redis cloud-instance to provide service for (say) dev, UAT and production, or multiple instances of your application, or whatever else - thus using more of the allocated memory and so being a little more cost-effective.

A use-case I'm looking at has several instances of an application which use 200-300K each, yet the minimum allocation on my cloud provider is 1M. We can consolidate 10 instances onto a single Redis without really making a dent in any limits, and so save about 90% of the Redis hosting cost. I appreciate there are limitations and issues with this approach, but thought it worth mentioning.

Share Improve this answer

answered May 22, 2020 at 10:42

Follow



Ralph Bolton

844 ● 9 ● 15

6 I think the use case of combining prod, dev, and UAT in the same Redis instance ignores the security implications of allowing access to production data to QA and Developer – [Larry Smith](#) Aug 20, 2020 at 17:53

4 Agreed - using the same redis server for non-prod and prod is probably a bad move in any "serious" environment. Mixing dev/uat may be possible in smaller shops, but your point is well made. Multiple Redis databases only allows this - it doesn't say it's a good idea ;-) – [Ralph Bolton](#) Sep 10, 2020 at 10:57



10



1. I don't really know any benefits of having multiple databases on a single instance. I guess it's useful if multiple services use the same database server(s), so you can avoid key collisions.
2. I would not recommend building around using the `KEYS` command, since it's $O(n)$ and that doesn't scale well. What are you using it for that you can accomplish in another way? Maybe redis isn't the best match for you if functionality like `KEYS` is vital.
3. I think they mention the benefits of a single threaded server in their FAQ, but the main thing is simplicity - you don't have to bother with concurrency in any real way. Every action is blocking, so no two things can alter the database at the same time. Ideally you would have one (or more) instances per core of each server, and use a consistent hashing algorithm (or a proxy) to divide the keys among them. Of course, you'll lose some functionality - piping will only work for things on the same server, sorts become harder etc.

Share Improve this answer

answered Apr 25, 2013 at 18:49


Follow



[Jonatan Hedborg](#)

4,432 ● 23 ● 31

In response to the 2: I use the keys command only when I need all the keys. I use it in the same way one would use `hgetall`. Both are $O(n)$. Keys is bad if you need to search through a huge set of keys for some regex, but it's perfectly fine if you need to do some operation on all the keys in some

db. In response to 3:I understand the benefits of single threading on one database. I don't understand it across many databases since an action on one database need never block an action on another database AFAIK. – [Eli](#) Apr 25, 2013 at 18:56 



8

Redis databases can be used in the rare cases of deploying a new version of the application, where the new version requires working with different entities.



Share Improve this answer

answered Jun 15, 2015 at 12:04

Follow



[Shlomi](#)

167 ● 1 ● 1



4

I am using redis for implementing a blacklist of email addresses , and i have different TTL values for different levels of blacklisting , so having different DBs on same instance helps me a lot .



Share Improve this answer

answered Mar 24, 2015 at 12:29

Follow



[kommrادHomer](#)

4,192 ● 5 ● 54 ● 71



1 We are facing now the same issue - we want to define different LRU policy for different parts of our data. can you please share how you implemented this? – [user2717436](#) Jul 12, 2017 at 10:07

@user2717436 i'm not sure if what i do is related to yours , but i use different databases as different sets , always setting

the TTL of the keys when i insert them . like there is blacklist A on redis.get(1) , and whenever i set a key there , i set the expire to 5000 . and there's blacklist B on redis.get(2) and whenever i set a key there , i set expire to 10000

– [kommradHomer](#) Jul 12, 2017 at 20:26



1



Our motivation has not been mentioned above. We use multiple databases because we routinely need to delete a large set of a certain type of data, and FLUSHDB makes that easy. For example, we can clear all cached web pages, using FLUSHDB on database 0, without affecting all of our other use of Redis.



There is some discussion here but I have not found definitive information about the performance of this vs scan and delete:

<https://github.com/StackExchange/StackExchange.Redis/issues/873>

Share Improve this answer

answered Dec 13, 2022 at 19:50

Follow



[Trevor Cox](#)

307 ● 1 ● 7



0



Using multiple databases in a single instance may be useful in the following scenario:

Different copies of the same database could be used for production, development or testing using real-time data. People may use replica to clone a redis instance to achieve the same purpose. However, the former



approach is easier for existing running programs to just select the right database to switch to the intended mode.

Share Improve this answer

answered Apr 10, 2019 at 12:56

Follow



[yoonghm](#)

4,615 ● 1 ● 36 ● 52

And in which way switching database is simpler than switching instance? In both cases, you need to change a number: either the database ID or the port number...

– [Roberto Lo Giacco](#) Sep 1, 2020 at 11:10

Switching database copy is simpler as they use the same login password, faster duplication of table from one copy to another, easier and faster data comparison, etc. If you switch instance, you would need in more memory for a Redis server, configuration files, longer delay in various operations.

– [yoonghm](#) Sep 1, 2020 at 13:58
