Redis is single-threaded, then how does it do concurrent I/O?

Asked 12 years, 7 months ago Modified 1 month ago

Viewed 139k times

246

Trying to grasp some basics of Redis I came across an interesting blog post .

The author states:

> Redis is single-threaded with epoll/kqueue and scale indefinitely in terms of I/O concurrency.

I surely misunderstand the whole threading thing, because I find this statement puzzling. If a program is single-threaded, how does it do anything concurrently? Why it is so great that Redis operations are atomic, if the server is single-threaded anyway?

Could anybody please shed some light on the issue?

multithreading redis concurrency epoll kqueue

Share

Improve this question

Follow

edited Nov 16 at 16:15

Guy Avraham

3,682 ● 3 ● 40 ● 53

asked May 7, 2012 at 21:19

Przemysław Pietrzkiewicz

2,747 ● 3 ● 16 ● 11

2 Starting from Redis 6, the **I/O is threaded**, see redis.com/blog/diving-into-redis-6 . Seems like full multithreading for Redis 7 is also being considered, as seen in this git issue: github.com/redis/redis/issues/8340 – Ng Sek Long Aug 26, 2021 at 4:52

@NgSekLong - agree. I am running Redis server 6.0.16 with defualt configuration and there are 5 threads in total in the Redis server process – Guy Avraham Nov 16 at 15:00

2 Answers

Sorted by: Highest score (default)

461

Well it depends on how you define concurrency.

In server-side software, concurrency and parallelism are often considered as different concepts. In a server, supporting concurrent I/Os means the server is able to serve several clients by executing several flows corresponding to those clients with only one computation unit. In this context, parallelism would mean the server is able to perform several things at the same time (with multiple computation units), which is different.

For instance a bartender is able to look after several customers while he can only prepare one beverage at a time. So he can provide concurrency without parallelism.

This question has been debated here: What is the difference between concurrency and parallelism?

See also this presentation from Rob Pike.

A single-threaded program can definitely provide concurrency at the I/O level by using an I/O (de)multiplexing mechanism and an event loop (which is what Redis does).

Parallelism has a cost: with the multiple sockets/multiple cores you can find on modern hardware, synchronization between threads is extremely expensive. On the other hand, the bottleneck of an efficient storage engine like Redis is very often the network, well before the CPU. Isolated event loops (which require no synchronization) are therefore seen as a good design to build efficient, scalable, servers.

The fact that Redis operations are atomic is simply a consequence of the single-threaded event loop. The interesting point is atomicity is provided at no extra cost (it does not require synchronization). It can be exploited by the user to implement optimistic locking and other patterns without paying for the synchronization overhead.

edited Nov 21, 2019 at 13:11

ChrisCantrell

3,853 ● 1 ● 23 ● 14

answered May 8, 2012 at 8:51

Didier Spezia

73.1k ● 12 ● 193 ● 157

7 v4 is a game changer in this respect - see my answer at stackoverflow.com/a/45374864/3160475 :) – Itamar Haber Jul 28, 2017 at 13:43

1 the only thing I don't really like about the answer and comparison is that it makes it seem like concurrency doesn't do work in parallel and it most certainly does as I can test this with running task async and getting work done that is ultimately considered to be in parallel. parallelism in the context of that article is referring to the multicore nature of being able to run on mutliple threads. I.e. why the refer to it being threadsafe. – Christian Matthew Jan 3, 2020 at 0:51

1 Still valid in 2020? – Roberto Manfreda Feb 20, 2020 at 10:33

1 The presentation by Rob Pike linked which is linked here (talks.golang.org/2012/waza.slide#1) is a gem, even after ~10 years ! – Devi May 6, 2021 at 11:56

30

OK, Redis is single-threaded at user-level, OTOH, all asynchronous I/O is supported by kernel thread pools and/or split-level drivers.

'**Concurrent**', to some, includes distributing network events to socket state-machines. It's single-threaded, runs on one core, (at user level), so I would not refer to this as concurrent. Others differ..

'**scale indefinitely in terms of I/O concurrency**' is just being economical with the truth. They may get more belief if they said 'can scale better than one-thread-per-client, providing the clients don't ask for much', though they may

then feel obliged to add 'blown away on heavy loading by other async solutions that use all cores at user level'.

Share Improve this answer

Follow

edited Jul 21, 2016 at 10:01

Abhinav kumar

116 ● 9

answered May 7, 2012 at 21:34

Martin James

24.8k ● 4 ● 38 ● 60

1 May be out of context but does each update operation (as by INCR command) carries a lock? If there are 1000 concurrent requests and one increment operation on a key (per request), does that ensure that the variable gets incremented only 1000 times? – Amanda Sep 23, 2019 at 8:03