designforscalability.com

How Redis Works? - Design For Scalability

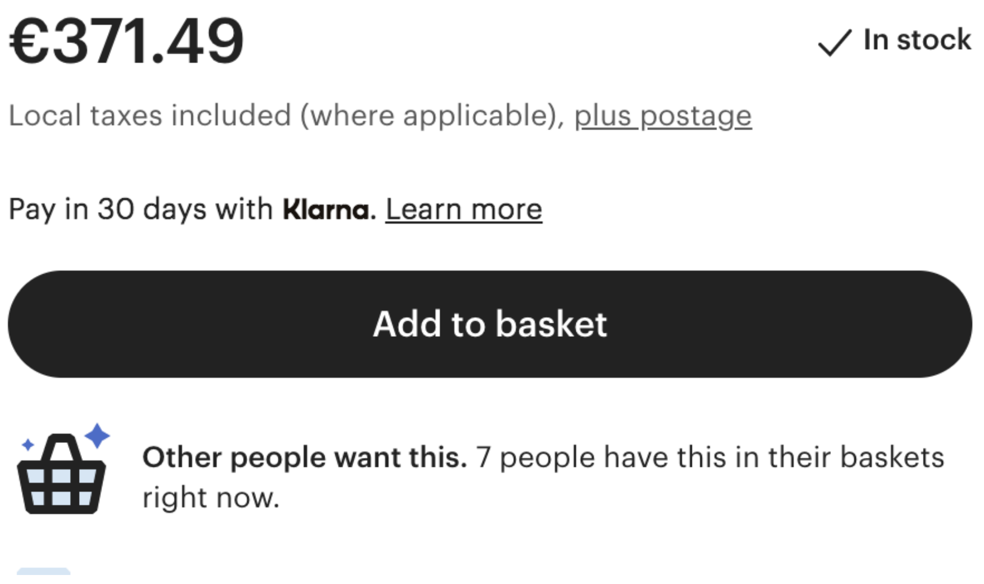
4 minutes

I have been doing design interviews for +10 years, and Redis is the most used component by candidates. It is a great software that solves many problems, but it is not a silver bullet for any design problems.

Let’s discuss how Redis works with a simple use case.

Assume you are working in a successful e-commerce company and your product manager came up with a simple requirement. The requirement is to show the number of people who added the product to their basket.

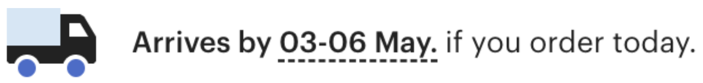
It is also important to note that showing the 100% correct number to users is not necessary. There is not much difference between 9 or 10 users having the product in their basket for a user.


€371.49 ✓ In stock

Local taxes included (where applicable), plus postage

Pay in 30 days with **Klarna**. Learn more

Other people want this. 7 people have this in their baskets right now.


Arrives by 03-06 May. if you order today.

You need to keep a counter for each product. You can increment the counter when a user adds the product to the basket and decrease it when they remove it. (In the real world, it is not trivial to decrement the count as users usually add products to the basket and leave them there for days. But, let’s keep such challenges out of the scope of this post)

Using Redis for the counter

You have many options to implement such a counter, but some possibilities are keeping it in memory, in a database, or in a distributed cache. We will go with the last option for this post and use Redis!

You can keep your counters in Redis, where the key will be ProductID, and the value would be numberOfUsers added product to the basket.

This simple counter approach will work like this

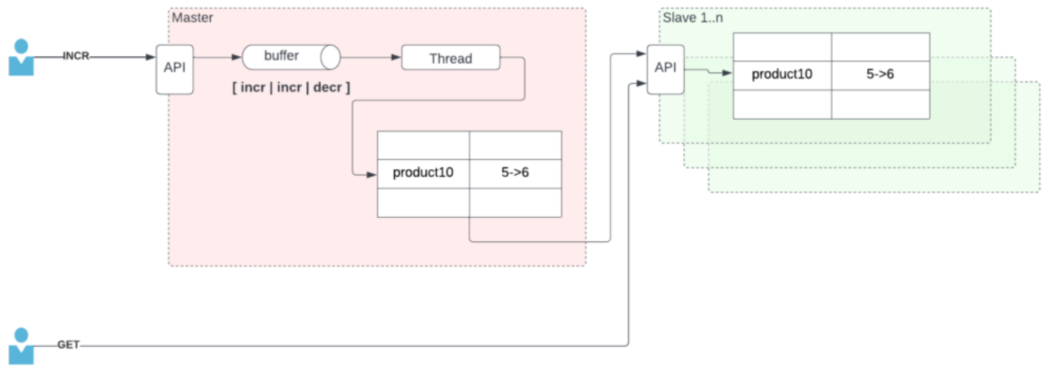
```
# user a added product10 to basket
> INCR product10

# user b added product10 to basket
> INCR product10

# user c added product10 to basket
> INCR product10

# get the counter
> GET product10
3
```

Redis INCR/DECR commands are atomic, which means even if you have multiple services writing simultaneously, Redis will INCR/DECR counters properly. But how can this work? You don't get any write locks, so how does Redis prevent simultaneous updates to shared resources?

How does Redis ensure atomicity?

The answer is simple, Redis has a single thread for command execution. When you make an increment request to Redis, the command execution thread gets the request and increments the counter. Once the thread completes its operation, it receives the subsequent request from the incoming requests.

While all writes go to master, you can set up read replicas to scale your reads. Redis master has some replication tasks that replicate data you have in master to read replicas.



How does Redis handle high traffic with a single-thread approach?

Redis could choose to be a multi-threaded application and use synchronization techniques to access memory. Then why the author (Antirez) didn't choose this approach?

Redis has a single command execution thread, but it can still handle concurrent client requests. Don't forget that concurrency is not parallelism. You can have concurrency with a single thread, and Redis is an excellent example of that.

Redis keeps its data in memory with a hashmap-like structure. So the majority of requests are purely memory operations, which usually have O(1) complexity. The command execution thread uses an event-poll approach to handle incoming requests. Redis has an optimized event library for its use-cases.

I finally understand that the Redis's power comes from its simplicity. It is an excellent software with a concrete design.

I will learn and share more about fantastic designs.