Redis6.0的多线程模型

退休程序猿 2022-10-12 阅读6分钟

背景

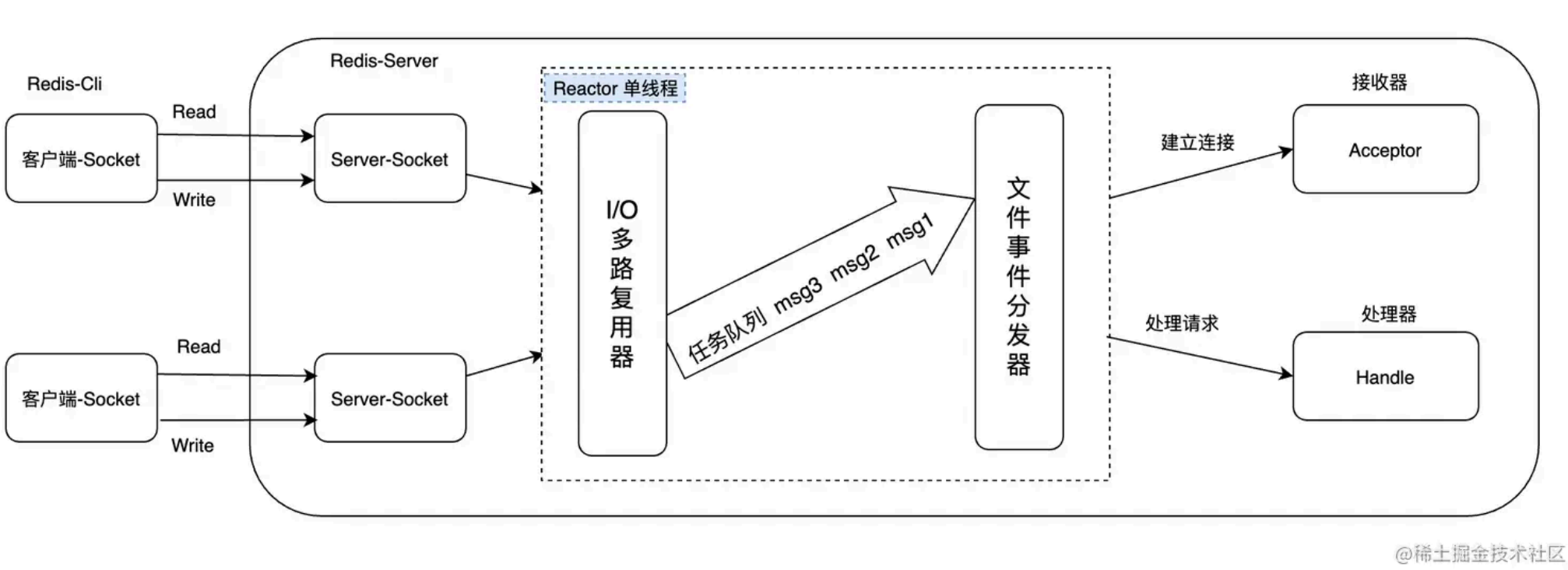
我们在第一篇《Redis系列1：深刻理解高性能Redis的本质》中就已经提到了，Redis 的网络 IO 以及键值对指令读写是由单个线程来执行的，避免了不必要的contextswitch和资源竞争，对于性能提升有很大的帮助。
而到了2020年的5月份，Redis官方 推出了 令人瞩目的 Redis 6.0，提出很多新特性，包含 多线程网络IO 的概念，如下：

新特性	内核优化	应用优化	其他
ACL细粒度权限管控(包括ACL LOG)	过期Key回收优化，增加配置参数	新版本Module API	全面支持SSL协议、并新增TSL协议
客户端缓存(Client side caching)	Resp3协议，兼容Resp2，更加简单、高效	disque消息队列模块	Redis-benchmark支持集群模式
多线程处理网络 IO（Threaded I/O）	优化了INFO命令，效率更高	新增配置，支持Del命令如unlink执行	Systemd支持重写
Redis集群代理（Cluster proxy）	优化阻塞命令，复杂度从O(n)到O(1)	XINFO STREAM FULL流命令	新增配置参数来删除用于在非持久性实例中进行复制的RDB文件
支持linux/bsd系统的CPU和线程(包括子线程如aof、dbIO线程)亲和力绑定	RDB加载速度优化	CLIENT KILL USER username命令	无磁盘复制副本(Diskless replication on replicas)，从测试版优化，目前无磁盘复制在load rdb仍是测试版。
	集群Slots命令优化		
	Psync2优化，修复了5.0的链式复制不一致问题。		
	defrag优化，从试验版到正式版		

这其中比较引人注意的就是 Threaded I/O 和 Client side caching 这两项了。
这时候我们不免疑问，为什么6.0之前是单线程模式的，是基于什么考虑。而现在为什么又要优化成 多线程网络IO模式，主要解决了哪些问题 ，带来了那些变化?
这一篇咱们就详细就来聊下这个 Threaded I/O。

6.0之前的单线程模式

了解单线程模式之前，大家可以先回顾一下Redis系列第一篇 Redis系列1：深刻理解高性能Redis的本质 。
就会明白，Redis所谓的单线程并不是所有工作都是只有一个线程在执行，而是指Redis的网络IO和键值对读写是由一个线程来完成的，Redis在处理客户端的请求时包括获取 (socket 读)、解析、执行、内容返回 (socket 写) 等都由一个顺序串行的主线程处理。
这就是所谓的“单线程”。这也是Redis对外提供键值存储服务的主要流程。
由于Redis在处理命令的时候是单线程作业的，所以会有一个Socket队列，每一个到达的服务端命令来了之后都不会马上被执行，而是进入队列，然后被线程的事件分发器逐个执行。如下图：



至于Redis的其他功能， 比如持久化、异步删除、集群数据同步等等，其实是由额外的线程执行的。 可以这么说，Redis工作线程是单线程的。但是在4.0之后，对于整个Redis服务来说，还是多线程运作的。

那么问题来了，6.0之前为什么要使用单线程，通过 Redis官方的文档 ，我们看到他们有给出了说明：

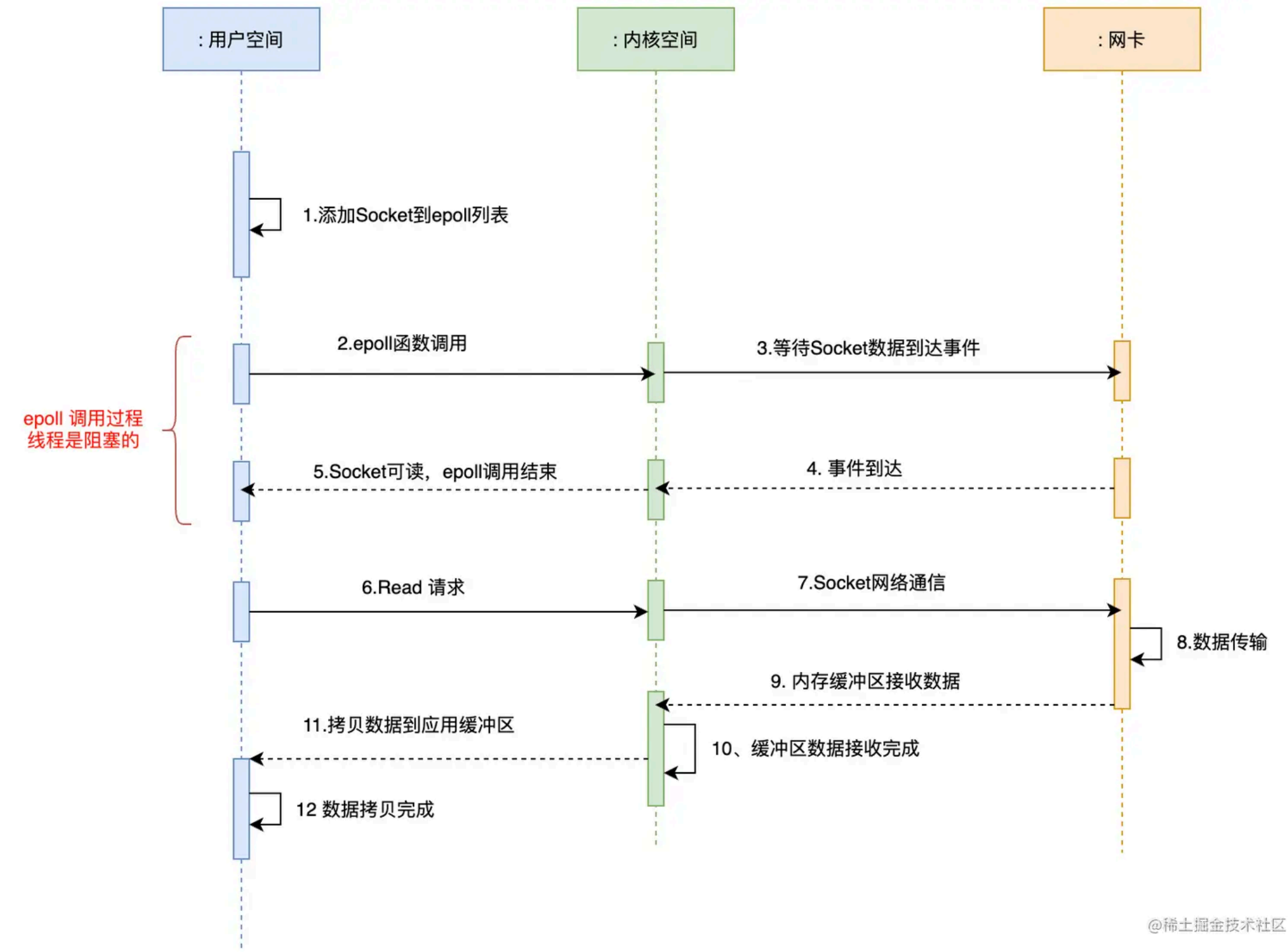
> Redis is single threaded. How can I exploit multiple CPU / cores?
>
> It's not very frequent that CPU becomes your bottleneck with Redis, as usually Redis is either memory or network bound. For instance, using pipelining Redis running on an average Linux system can deliver even 1 million requests per second, so if your application mainly uses O(N) or O(log(N)) commands, it is hardly going to use too much CPU.
>
> However, to maximize CPU usage you can start multiple instances of Redis in the same box and treat them as different servers. At some point a single box may not be enough anyway, so if you want to use multiple CPUs you can start thinking of some way to shard earlier.
>
> You can find more information about using multiple Redis instances in the Partitioning page.
>
> However with Redis 4.0 we started to make Redis more threaded. For now this is limited to deleting objects in the background, and to blocking commands implemented via Redis modules. For future releases, the plan is to make Redis more and more threaded.
>
> @稀土掘金技术社区

- 在使用 Redis 时，Redis 主要受限是在内存和网络上，CPU 几乎没有性能瓶颈的问题。
- 以Linux 系统为例子，在Linux系统上Redis 通过 pipelining 可以处理 100w 个请求每秒，而应用程序的计算复杂度主要是 O(N) 或 O(log(N))，不会消耗太多 CPU。
- 使用了单线程后，提高了可维护性。多线程模型在某些方面表现优异，却增加了程序执行顺序的不确定性，并且带来了并发读写的一系列问题，增加了系统复杂度。同时因为线程切换、加解锁，甚至死锁，造成一定的性能损耗。
- Redis 通过 AE 事件模型以及 IO 多路复用等技术，拥有超高的处理性能，因此没有使用多线程的必要。

可以看出，Redis对CPU计算力的要求并不迫切，相反单线程机制让 Redis 内部实现的复杂度大大降低，同时降低了因为上下文切换和资源竞争造成的性能损耗。那既然单线程这么好用，为什么要引入多线程模式。

6.0之后的多线程主要解决什么问题

我们知道， 近年来底层网络硬件性能越来越好，Redis 的性能瓶颈逐渐体现在网络 I/O 的读写上，单个线程处理网络 I/O 读写的速度跟不上底层网络硬件执行的速度。
从下图我们可以看到，Redis 在处理网络数据时，调用 epoll 的过程是阻塞的，这个过程会阻塞线程。如果并发量很高，达到万级别的 QPS，就会形成瓶颈，影响整体吞吐能力。



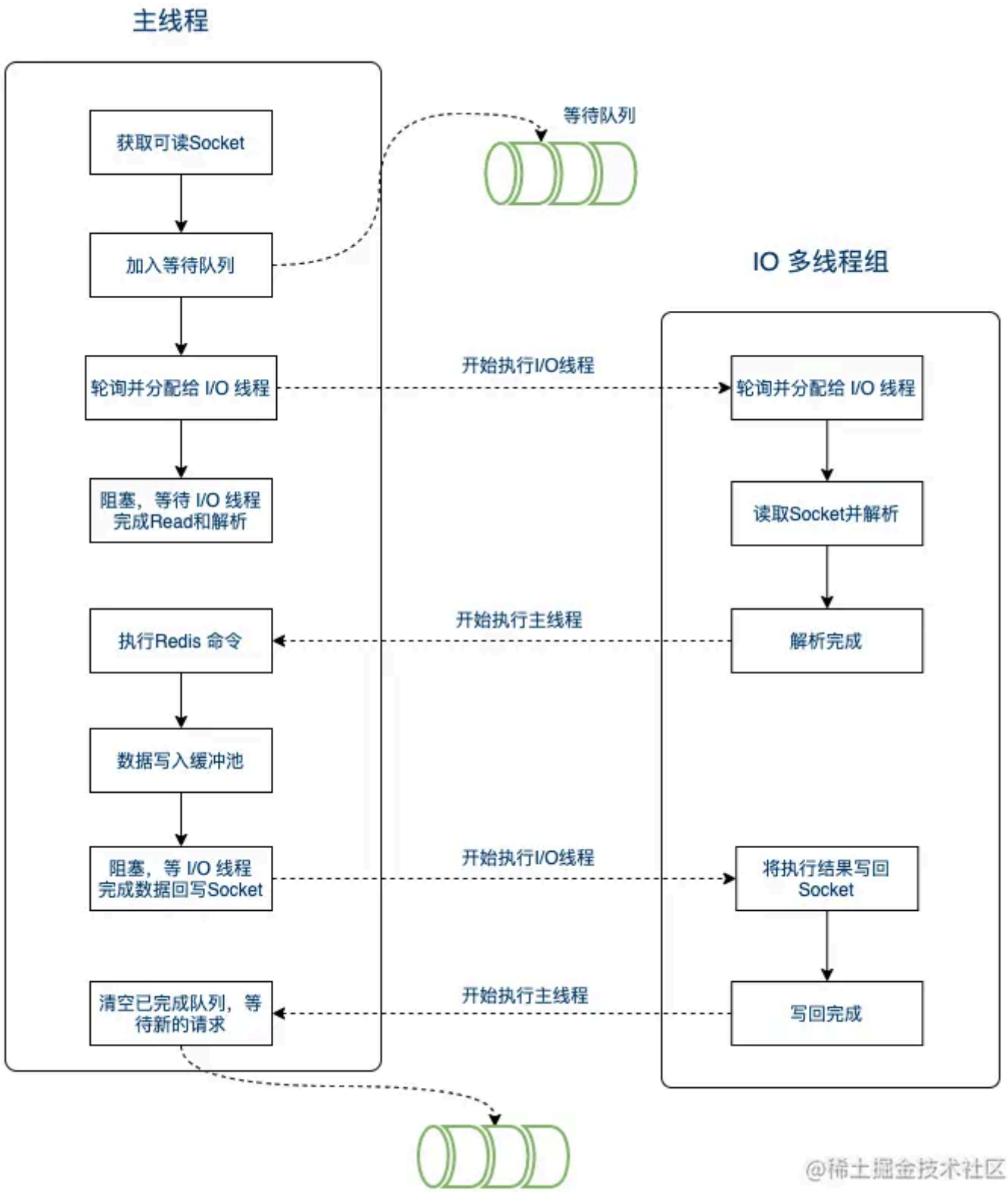
既然读写网络的 read/write 系统调用占用了Redis 执行期间大部分CPU 时间，那么要想真正做到提速，必须改善网络IO性能。我们可以从这两个方面来优化：

- 提高网络 IO 性能，典型实现方式比如使用 DPDK 来替代内核网络栈的方式
- 使用多线程，这样可以充分利用多核CPU，同类实现案例比如 Memcached。

协议栈优化的这种方式跟 Redis 关系不大，所以最便捷高效的方式就是支持多线程。总结起来，redis支持多线程就是以下两个原因：

- 可以充分利用服务器CPU的多核资源，而主线程明显只能利用一个
- 多线程任务可以分摊 Redis 同步 IO 读写负荷，降低耗时

6.0版本优化之后，主线程和多线程网络IO的执行流程如下：



具体步骤如下：

- 主线程建立连接，并接受数据，并将获取的 socket 数据放入等待队列；
- 通过轮询的方式将 socket读取出来并分配给 IO 线程；
- 之后主线程保持阻塞，一直等到 IO 线程完成 socket 读取和解析；
- I/O 线程读取和解析完成之后，返回给主线程 ，主线程开始执行 Redis 命令；
- 执行完Redis命令后，主线程阻塞，直到IO 线程完成 结果回写到socket 的工作；
- 主线程清空已完成的队列，等待客户端新的请求。

本质上是将主线程 IO 读写的这个操作 独立出来，单独交给一个I/O线程组处理。
这样多个 socket 读写可以并行执行，整体效率也就提高了。同时注意 Redis 命令还是主线程串行执行。

开启多线程的方式

Redis6.0的多线程默认是禁用的，只使用主线程。如需开启需要修改redis.conf配置文件：

lua 复制代码

```
# io-threads-do-reads no
io-threads-do-reads yes
```

开启多线程后，还需要设置线程数，否则是不生效的。同样修改redis.conf配置文件。
关于线程数的设置，官方有一个建议：4 核的机器建议设置为 2 或 3 个线程，8核的建议设置为 6 个线程，线程数一定要小于机器核数。
线程数并不是越大越好，官方认为超过了 8 个就很难继续提效了，没什么意义。

bash 复制代码

```
# 假设你的CPU核数是8核，尽量配置成 5~6
io-threads 5
```

总结

- 6.0之前，Redis所谓的单线程并不是所有工作都是只有一个线程在执行，而是指Redis的网络IO和读写是由一个线程来完成的。其他诸如持久化、异步删除、集群数据同步等，其实是由额外的线程执行的。
- 互联网飞速发展，开发人员面临的线上流量场景越来越大，再使用单线程模式会导致在网络 I/O 浪费太多时间，极大的降低吞吐量，而普遍多核的cpu又没有得到有效的利用。
- 使用多线程，这样可以充分利用多核CPU，提高网络的 read/write 效率。
- 配置 Threaded I/O 多线程模式的时候，线程数一定要小于机器核数，否着意义不大。

这期就写到这里了，持续更新，关注公众号：退休程序猿 持续为你带来技术小知识~~