目录

[P100 100\_redis高级篇之开篇闲聊扯淡 1](#_Toc128346482)

[P101 101\_redis高级篇之为什么用单线程 1](#_Toc128346483)

[1.1面试题 1](#_Toc128346484)

[Redis到底是单线程还是多线程 2](#_Toc128346485)

[1.2Redis为什么选择单线程? 2](#_Toc128346486)

[厘清一个事实我们通常说,Redis是单线程究竟何意? 2](#_Toc128346487)

[Redis3.x单线程时代但性能依旧很快的主要原因 3](#_Toc128346488)

[P102 102\_redis高级篇之开始支持多线程和IO 5](#_Toc128346489)

[1.3既然单线程这么好，为什么逐渐又加入了多线程特性? 5](#_Toc128346490)

[单线程的苦恼 5](#_Toc128346491)

[P103 103\_redis高级篇之开启多线程IO特性支持 6](#_Toc128346492)

[P104 104\_redis高级篇之BigKey大厂面试题概览 6](#_Toc128346493)

[P105 105\_redis高级篇之BigKey100W记录案 6](#_Toc128346494)

[P106 106\_redis高级篇之BigKey发现删除优化 6](#_Toc128346495)

[P107 107\_redis高级篇之缓存双写—致性面试 6](#_Toc128346496)

[P108 108\_redis高级篇之缓存双写一致性细则 6](#_Toc128346497)

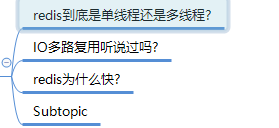
[P109 109 redis高级篇之缓存双写—致件四大 6](#_Toc128346498)

# P100 100\_redis高级篇之开篇闲聊扯淡

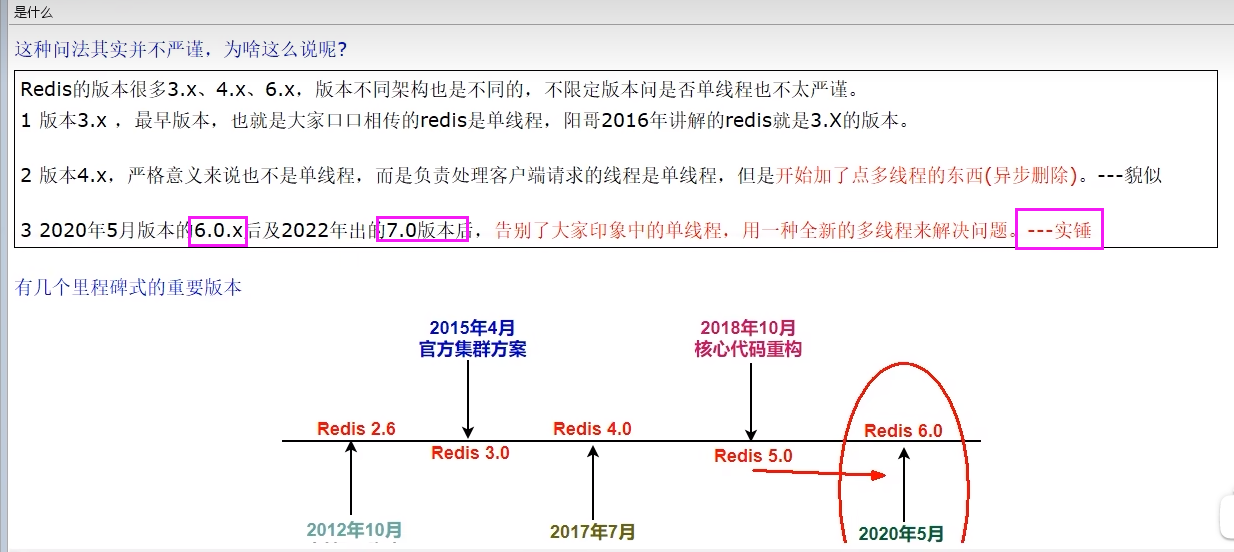
闲聊

# P101 101\_redis高级篇之为什么用单线程

## 1.1面试题



### Redis到底是单线程还是多线程

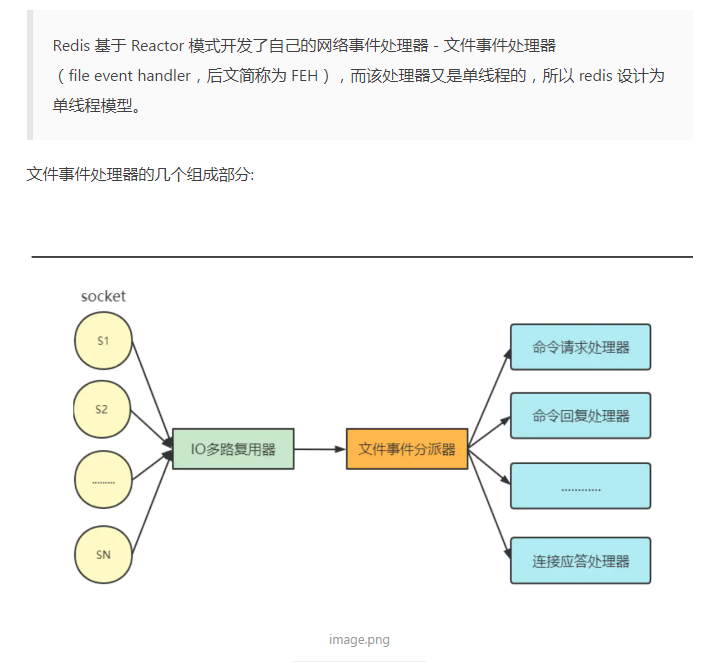


5.0版本是直接升级到6.0版本，对于这个激进的升级，Redis之父antirez表现得很有信心和兴奋，所以第一时间发文来阐述6.0的一些重大功能"Redis 6.0.0 GA is out!"当然，Redis7.0后版本更加厉害

## 1.2Redis为什么选择单线程?

### 厘清一个事实我们通常说,Redis是单线程究竟何意?

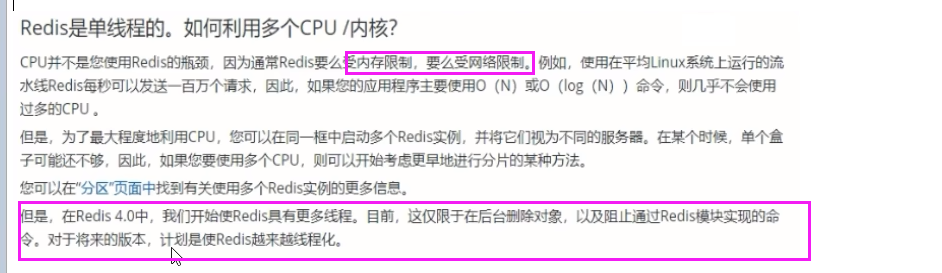




### Redis3.x单线程时代但性能依旧很快的主要原因

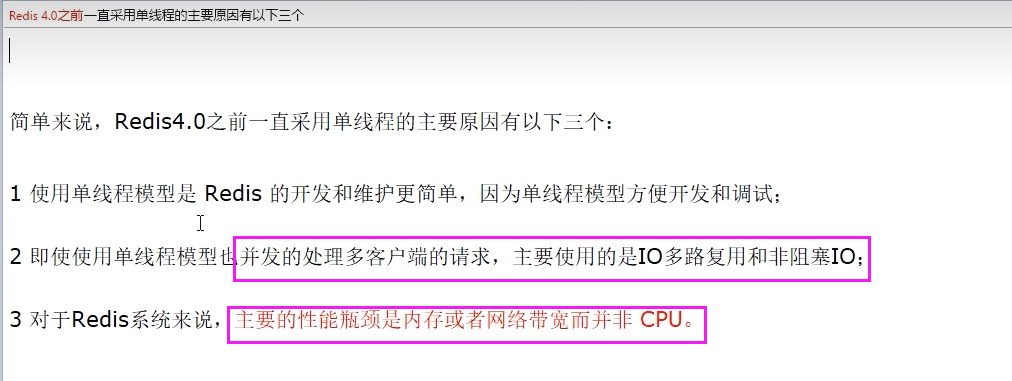
* 基于内存操作: Redis的所有数据都存在内存中，因此所有的运算都是内存级别的，所以他的性能比较高;
* 数据结构简单: Redis的数据结构是专门设计的，而这些简单的数据结构的查找和操作的时间大部分复杂度都是o(1)，因此性能比较高;
* 多路复用和非阻塞I/O: Redis使用I/O多路复用功能来监听多个socket连接客户端，这样就可以使用一个线程连接来处理多个请求，减少线程切换带来的开销，同时也避免了I/O阻塞操作。
* 避免上下文切换:因为是单线程模型，因此就避免了不必要的上下文切换和多线程竞争，这就省去了多线程切换带来的时间和性能上的消耗，而且单线程不会导致死锁问题的发生

#### 官网原话





#### Redis 4.0之前一直采用单线程的主要原因有以下三个



# P102 102\_redis高级篇之开始支持多线程和IO

## 1.3既然单线程这么好，为什么逐渐又加入了多线程特性?

### 单线程的苦恼



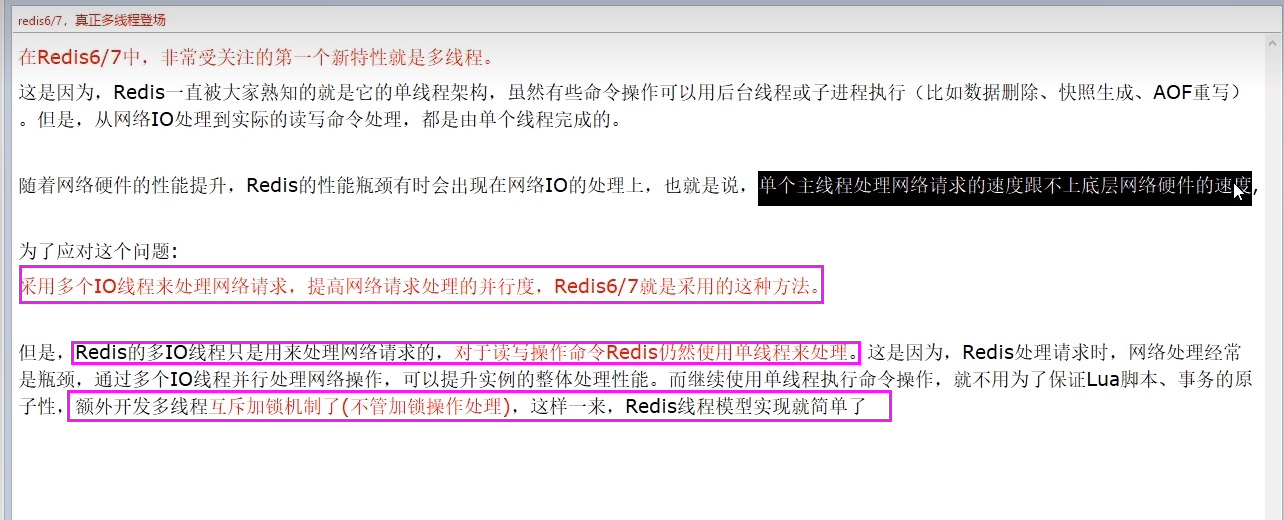
### 解决方式



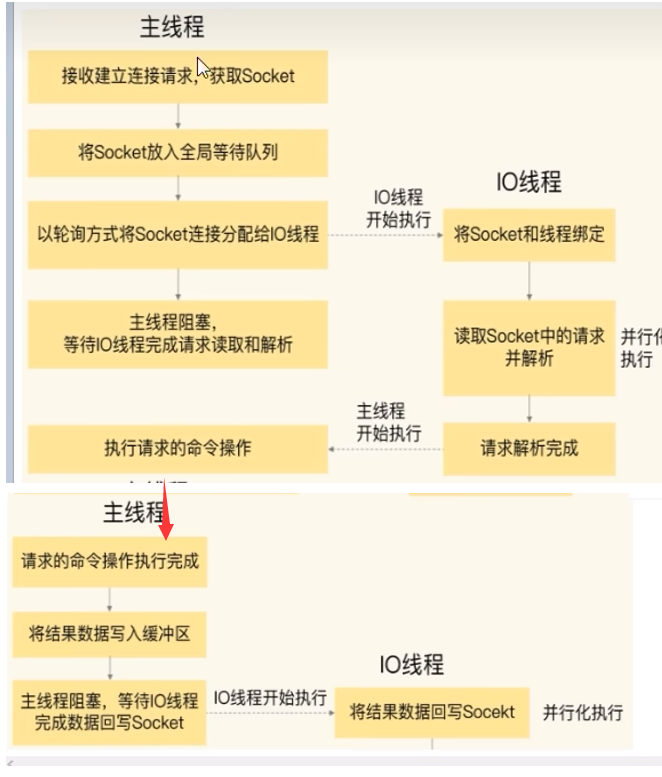
## 1.4redis6/7的多线程特性和IO多路复用入门篇

对于Redis主要的性能瓶颈是内存或者网络带宽而并非CPU。

### redis6/7，真正多线程登场

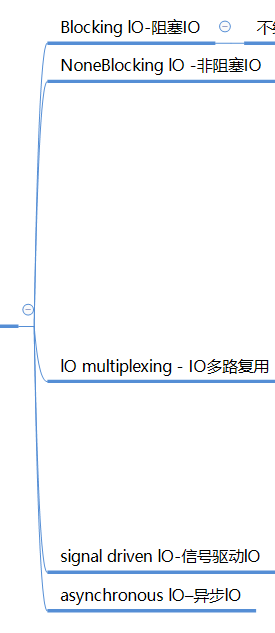


### 主线程和IO线程是怎么协作完成请求处理的-精讲版（四个阶段）



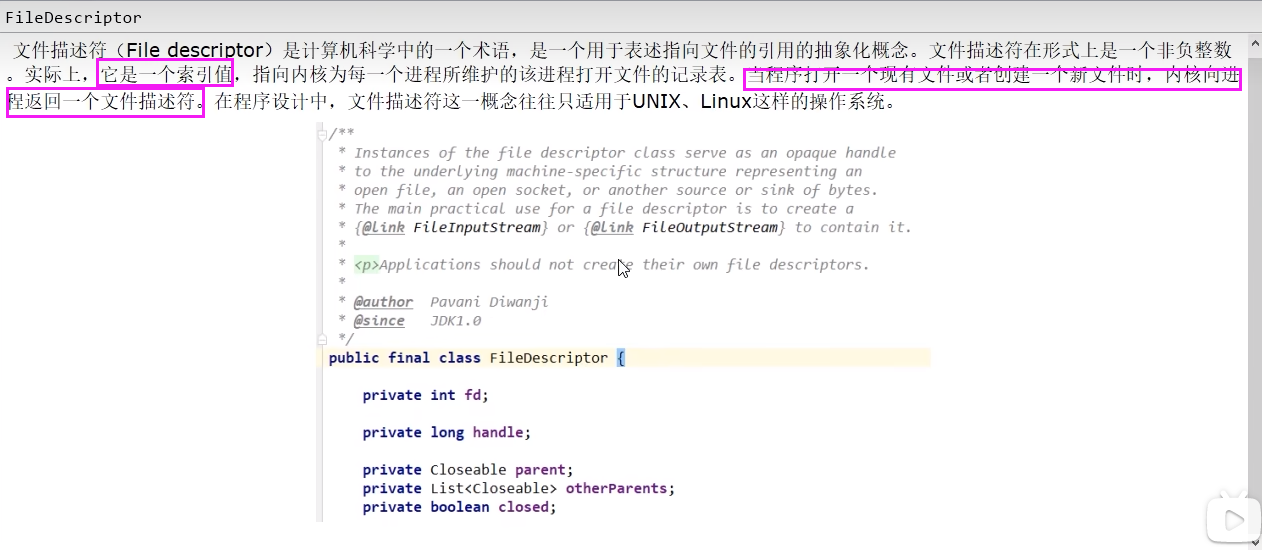


### Unix网络编程中的五种IO模型（redis为啥这么快）

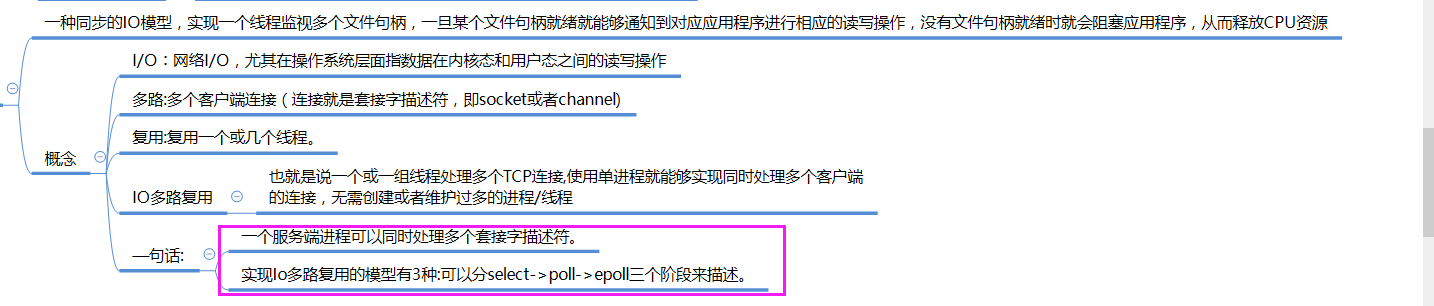


#### lO multiplexing - IO多路复用

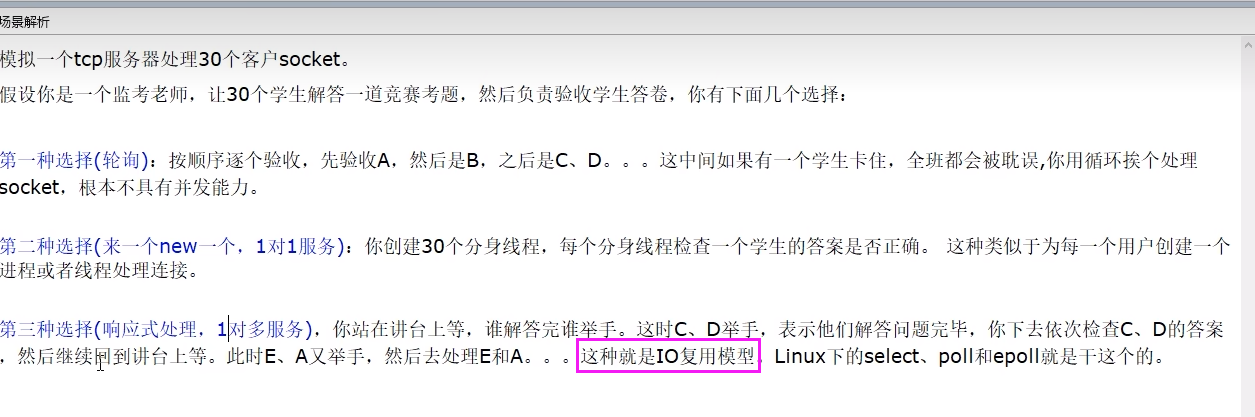
##### 文件描述符概念

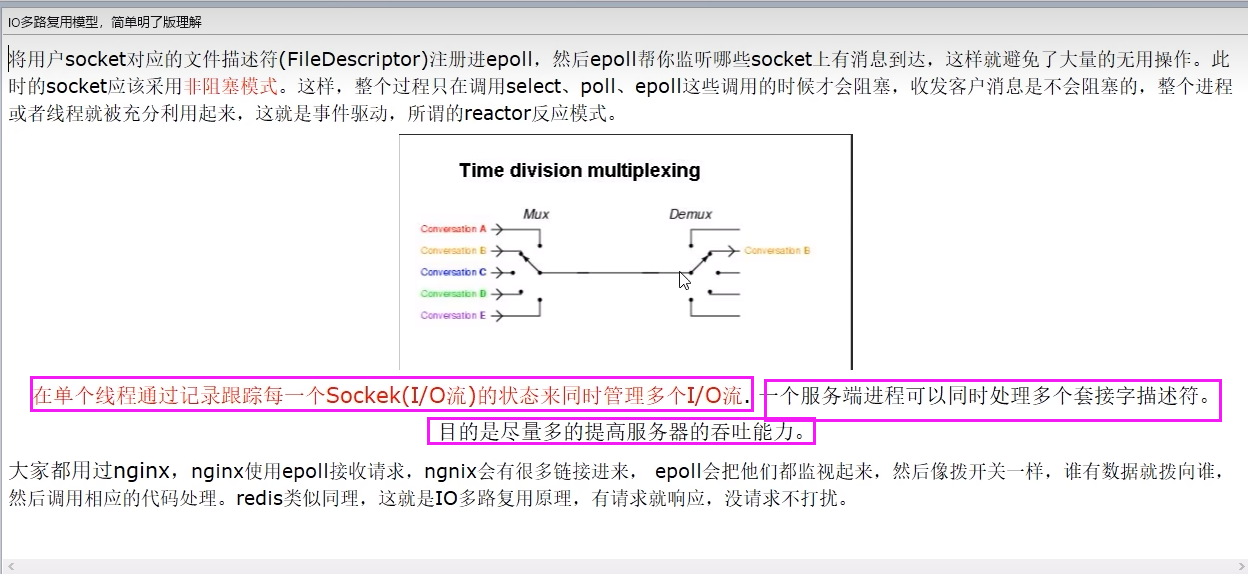


##### 首次浅谈IO多路复用,lo多路复用是什么。



##### 场景体验，说人话引出epoll

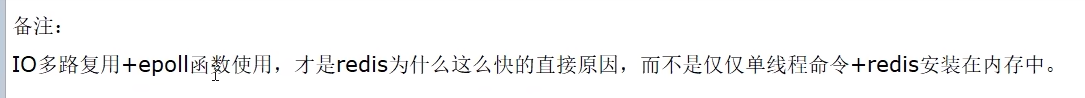




##### 小总结

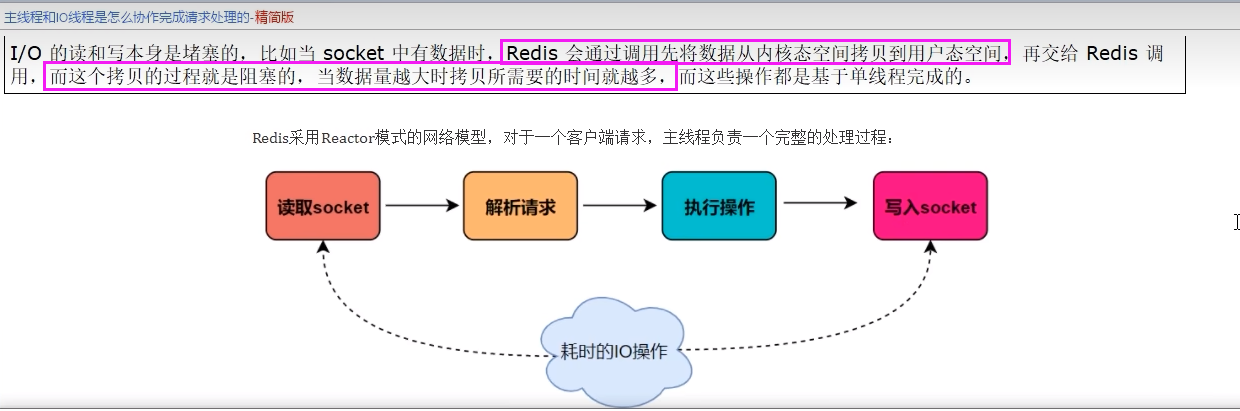
只使用一个服务端进程可以同时处理多个套接字描述符连接

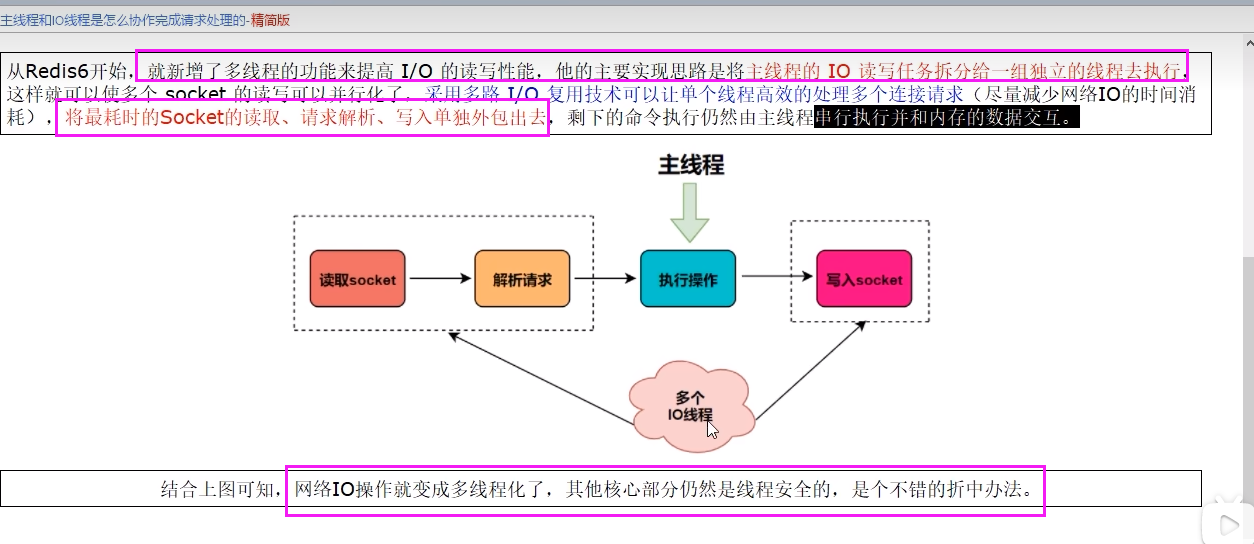
##### Redis为啥这么快



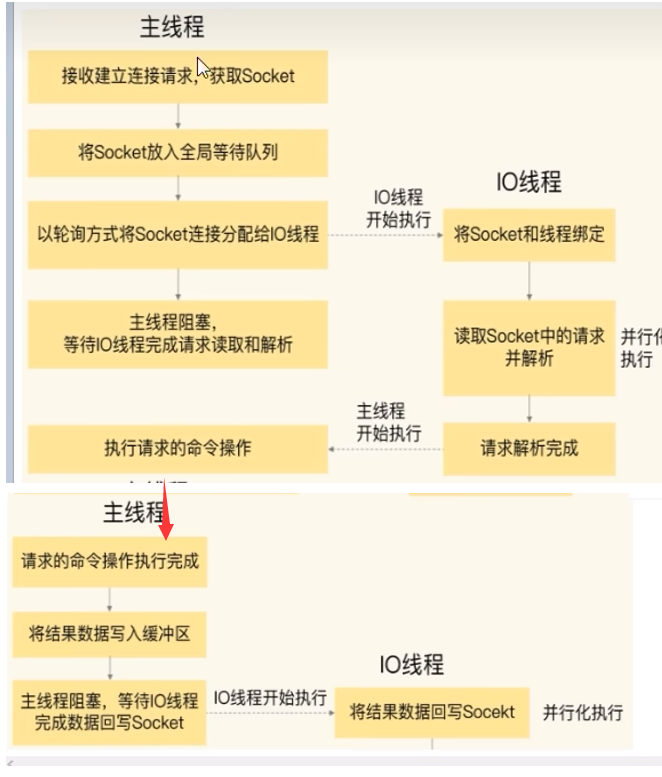
#### 简单说明

Redis工作线程是单线程的,但是，整,个Redis来说，是多线程的;（详细请看，主线程和IO线程是怎么写作完成请求处理的-精讲版）

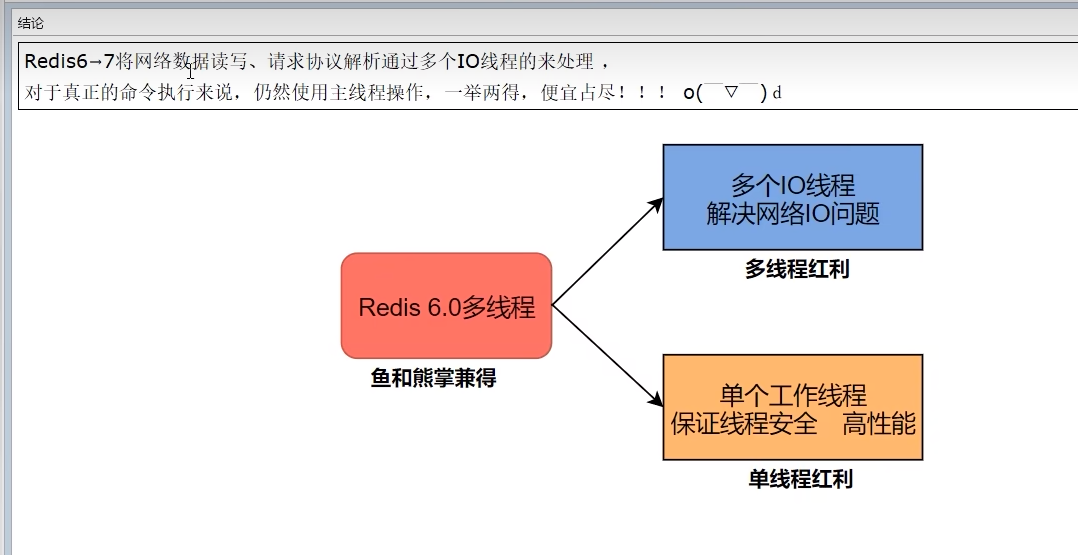




结合上图可知，网络IO操作就变成多线程化了，其他核心部分仍然是线程安全的，是个不错的折中办法。



### \*面试的时候讲这个，redis6/7的多线程特性和IO多路复用

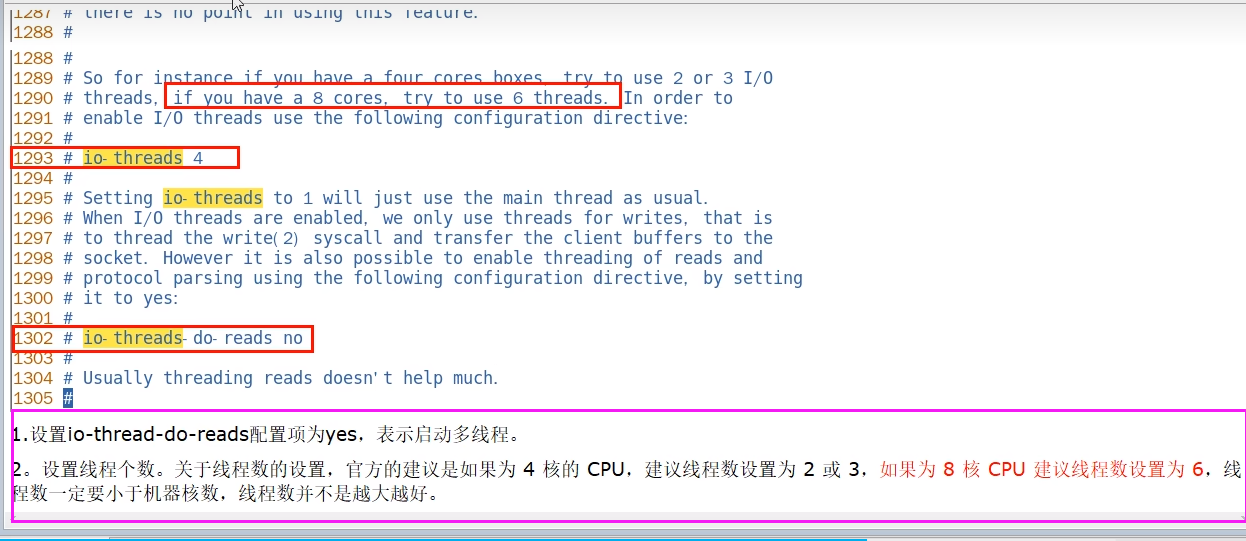


# P103 103\_redis高级篇之开启多线程IO特性支持

## 1.5Redis7默认是否开启了多线程?（默认关闭）

如果你在实际应用中，发现Redis实例的CPU开销不大但吞吐量却没有提升，可以考虑使用Redis7的多线程机制，加速网络处理，进而提升实例的吞吐量





## 1.6我还是曾经那个少年



# P104 104\_redis高级篇之BigKey大厂面试题概览

# P105 105\_redis高级篇之BigKey100W记录案

# P106 106\_redis高级篇之BigKey发现删除优化

# P107 107\_redis高级篇之缓存双写—致性面试

# P108 108\_redis高级篇之缓存双写一致性细则

# P109 109 redis高级篇之缓存双写—致件四大