#### 缓存数据的分类：

实时同步数据：

要求缓存中的数据必须与DB中数据保持一致，如何保证？只要DB中的数据发生了变更，缓存中的数据立即消失。

阶段性同步数据：

其没有必要必须与DB中的数据保持一致，只要大差不差就行；如何实现？为缓存数据添加了生存时长属性。

#### Redis的特性：

性能极高

redis读的速度可以达到11w次/s，写的速度可以达到8w次/s，之所以具有这么高的性能，有以下几点原因：

1. Redis的所有操作都是在内存中发生的
2. Redis是用C语言开发的
3. Redis源码非常精细(集性能与优雅于一身)

简单稳定

Reids源码很少，早期版本中只有2万行左右，从3.0版本开始，增加了集群功能，代码变为5w行左右

持久化

Redis内存中的数据可以进行持久化，有两种方式：RDB与AOF

高可用集群

Redis提供您了高可用的主从集群功能，可以确保系统的安全性

丰富的数据类型

Redis是一个key-value存储系统，支持存储的value类型很多，包括String(字符串)、List(链表)、Set(集合)、Zset(sorted set 有序集合)和Hash(哈希类型mm)等，还有

BitMap(一般用于大数据两的二值性统计)、

HyperLogLog(Hyperlog Log，用于对数据量超级庞大的日志做去重统计)、 Geospatial(地理空间，主要用于地理位置相关计算)。

强大的功能

Redis提供了数据过期功能、发布/订阅功能、简单事务功能，还支持Lua脚本扩展功能

客户端语言广泛

Redis提供了简单的TCP通信协议，编程语言可以方便的接入Redis,所以，有很多开源社区、大公司等开发出了很多语言的Redis客户端。

支持ACL权限控制

之前的权限控制非常笨拙，从Redis6开始引入了ACL模块，可以为不同用户定制不同的用户权限。

ACL，Access Control List,访问控制列表，是一种细粒度的权限管理策略，可以针对任意用户进行权限控制，目前大多数Unix系统与Linux2.6版本已经支持ACL了，Zookeeper早已支持ACL了。

Unix与Linux系统默认使用是UGO(User、Group、Other)权限控制策略，其是一种粗粒度的权限管理策略。

支持多线程IO模型

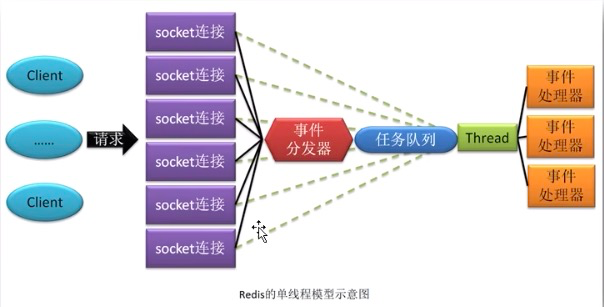
Redis之前版本采用的是单线程模型，从6.0版本开始支持了多线程模型

#### Redis的IO模型：

Redis客户端提交的各种请求是如何最终被Redis处理的？Redis处理客户端请求锁采用的架构，称为Redis的IO模型，不同版本的Redis采用的IO模型是不同的：

##### 单线程模型：

对于Redis3.0及其以前版本，Redis的IO模型采用的是纯粹的单线程模型，即所有客户端的请求全部由一个线程处理，



Redis的单线程模型采用了多路复用技术。

|  |
| --- |
| 对于多路复用器的多路选择算法常见的有三种：selct模型、poll模型、epoll模型  poll模型的选择算法：采用的是轮询算法，该模型对客户端的就绪处理是有延迟的。  epoll模型的选择算法：采用的是回调方式，根据就绪事件发生后的处理方式的不同，又可分为LT模型与ET模型。 |

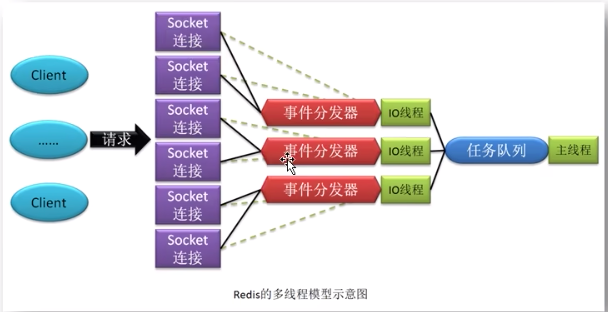
每个客户端若要向Redis提交请求，都需要与Redis监理一个socket连接，并向事件分发器注册一个事件，一旦该事件发生就表明该连接已经就绪，而一旦连接就绪，事件分发器就会感知到，然后获取客户端通过该连接发送的请求，并将由该事件分发器绑定的这个唯一线程来处理，如果该线程还在处理多个任务，则将任务写入到任务队列等待线程处理；之所以称为事件分发器，是因为它会根据不同的就绪事件，将任务交由不同的事件处理器去处理

##### 混合线程模型：

从Redis4.0版本开始，Redis中就开始加入了多线程元素，处理客户端请求仍是单线程模型，但对于一些比较耗时但又不影响对客户端的相应的操作，就由后台其他线程来处理，例如：持久化、对AOF的rewrite、对失效链接的清理等。

##### 多线程模型：

Redis6.0版本，才是真正意义上的多线程模型，因为其对于客户端请求的处理采用的是多线程模型。



多线程IO模型中的”多线程”仅用于接受、解析客户端的请求，然后将解析出的请求写入到任务队列，而对具体任务(命令)的处理，仍是由主线程处理，这样做是的用户不需考虑线程安全问题，无需考虑事务控制，无需考虑LPUSH/LPOP等命令执行顺序问题。

优缺点总结：

单线程模型：

优点：可维护性高，性能高。不存在并发读写情况，所以也就不存在执行顺序的不确定性，不存在线程切换开销，不存子啊死锁问题，不存在未了数据安全而进行的加锁/解锁开销。

缺点：性能会收到影响，且由于单线程只能使用一个处理器，所以造成处理器浪费。

多线程模型：

优点：结合了多线程与单线程的优点，避开了它们所有的不足

缺点：该模型没有明显不足，如果非要找不足的话就是，其并非是一个真正意义上的 “多线程”，因为真正处理”任务”的线程仍是单线程，所以，对性能也是有些影响的。

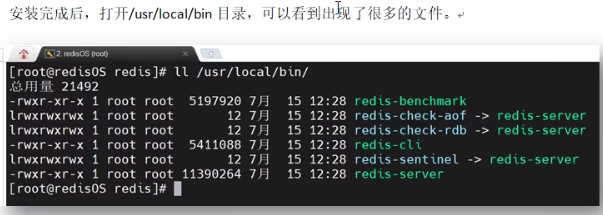
### Redis安装

1. 安装gcc

由于Redis是由C/C++语言编写的，而从官网下载的Redis安装包是需要编译后才可以安装的，所以对其进行编译就必须要使用相关编译器，对于C/C++语言的编译器，使用最多的就是gcc与gcc-c++，而这两款编译器在CentOS7中是没有安装的，所以首先要安装这两款编译器。



1. wget下载Redis源码包
2. 解压Redis
3. 编译 make && make install



1. 启动

前台启动：redis -server

命令式启动：nohup redis-server &

守护进程方式启动：

vim redis.conf

daemon yes

redis-server /opt/apps/redis/redis.conf