# 1、简单介绍一下Redis优点和缺点？

优点：

1、本质上是一个 Key-Value 类型的内存数据库，很像memcached

2、整个数据库统统加载在内存当中进行操作，定期通过异步操作把数据库数据 flush 到硬盘上进行保存

3、因为是纯内存操作，Redis 的性能非常出色，每秒可以处理超过 10 万次读写操作，是已知性能最快的Key-Value DB

4、Redis最大的魅力是支持保存多种数据结构(string,list,set,hash,sortedset)，此外单个 value 的最大限制是 1GB，不像memcached只能保存 1MB 的数据

5、Redis也可以对存入的 Key-Value 设置 expire 时间，因此也可以被当作一个功能加强版的memcached 来用

缺点：

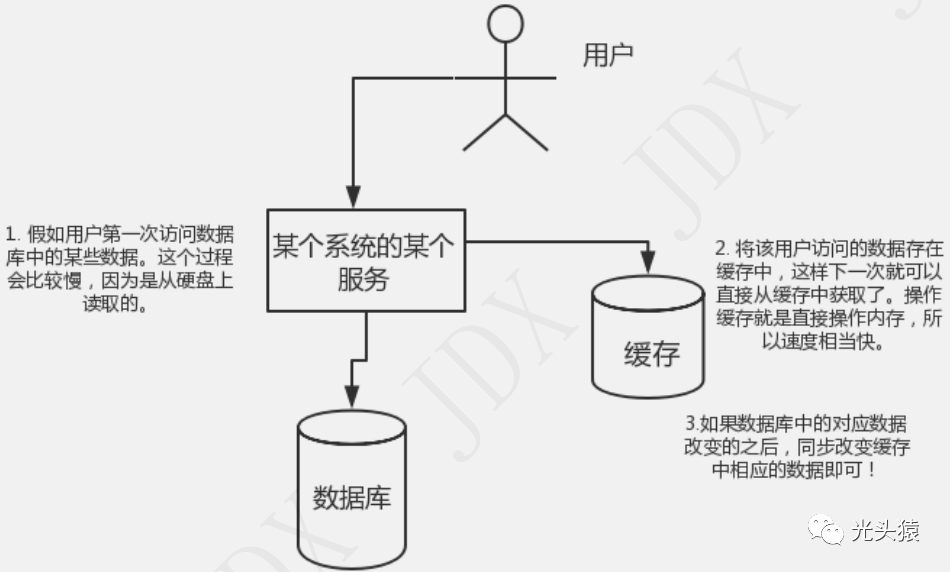
1、Redis 的主要缺点是数据库容量受到物理内存的限制，不能用作海量数据的高性能读写，因此 Redis 适合的场景主要局限在较小数据量的高性能操作和运算上。

# 2、系统中为什么要使用缓存？

主要从“高性能”和“高并发”这两点来看待这个问题。

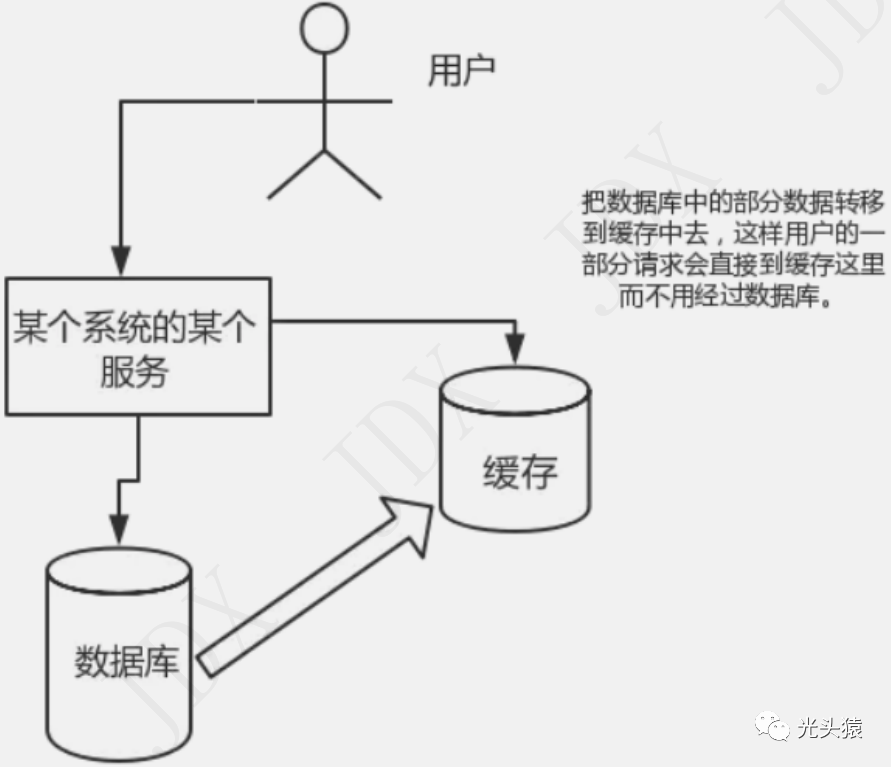
高性能：

假如用户第一次访问数据库中的某些数据。这个过程会比较慢，因为是从硬盘上读取的。将该用户访问的数据存在缓存中，这样下一次再访问这些数据的时候就可以直接从缓存中获取了。操作缓存就是直接操作内存，所以速度相当快。如果数据库中的对应数据改变的之后，同步改变缓存中相应的数据即可！



高并发：

直接操作缓存能够承受的请求是远远大于直接访问数据库的，所以我们可以考虑把数据库中的部分数据转移到缓存中去，这样用户的一部分请求会直接到缓存这里而不用经过数据库。



# 3、常见的缓存同步方案都有哪些？

同步方案：更改代码业务代码，加入同步操作缓存逻辑的代码(数据库操作完毕以后，同步操作缓存)

异步方案：

1、使用消息队列进行缓存同步：更改代码加入异步操作缓存的逻辑代码(数据库操作完毕以后，将要同步的数据发送到MQ中，MQ的消费者从MQ中获取数据，然后更新缓存)

2、使用阿里巴巴旗下的canal组件实现数据同步：不需要更改业务代码，部署一个canal服务。canal服务把自己伪装成mysql的一个从节点，当mysql数据更新以后，canal会读取binlog数据，然后再通过canal的客户端获取到数据，更新缓存即可。

# 4、Redis常见数据结构以及使用场景有哪些？

1、 string

常见命令：set、get、decr、incr、mget等。

基本特点：string数据结构是简单的key-value类型，value其实不仅可以是String，也可以是数字。

应用场景：常规计数：微博数，粉丝数等。

2、hash

常用命令：hget、hset、hgetall等。

基本特点：hash 是一个 string 类型的 field 和 value 的映射表，hash 特别适合用于存储对象，后续操作的时候，你可以直接仅仅修改这个对象中的某个字段的值。

应用场景：存储用户信息，商品信息等。

3、list

常用命令: lpush、rpush、lpop、rpop、lrange等。

基本特点：类似于Java中的list可以存储多个数据，并且数据可以重复，而且数据是有序的。

应用场景：存储微博的关注列表，粉丝列表等。

4、set

常用命令：sadd、spop、smembers、sunion 等

基本特点：类似于Java中的set集合可以存储多个数据，数据不可以重复，使用set集合不可以保证数据的有序性。

应用场景：可以利用Redis的集合计算功能，实现微博系统中的共同粉丝、公告关注的用户列表计算。

5、sorted set

常用命令：zadd、zrange、zrem、zcard 等。

基本特点：和set相比，sorted set增加了一个权重参数score，使得集合中的元素能够按score进行有序排列。

应用场景：在直播系统中，实时排行信息包含直播间在线用户列表，各种礼物排行榜等。

# 5、Redis有哪些数据删除策略？

数据删除策略：Redis中可以对数据设置数据的有效时间，数据的有效时间到了以后，就需要将数据从内存中删除掉。而删除的时候就需要按照指定的规则进行删除，这种删除规则就被称之为数据的删除策略。

Redis中数据的删除策略：

① 定时删除

概述：在设置某个key 的过期时间同时，我们创建一个定时器，让定时器在该过期时间到来时，立即执行对其进行删除的操作。

优点：定时删除对内存是最友好的，能够保存内存的key一旦过期就能立即从内存中删除。

缺点：对CPU最不友好，在过期键比较多的时候，删除过期键会占用一部分CPU时间，对服务器的响应时间和吞吐量造成影响。

② 惰性删除

概述：设置该key过期时间后，我们不去管它，当需要该key时，我们再检查其是否过期，如果过期，我们就删掉它，反之返回该key。

优点：对CPU友好，我们只会在使用该键时才会进行过期检查，对于很多用不到的key不用浪费时间进行过期检查。

缺点：对内存不友好，如果一个键已经过期，但是一直没有使用，那么该键就会一直存在内存中，如果数据库中有很多这种使用不到的过期键，这些键便永远不会被删除，内存永远不会释放。

③ 定期删除

概述：每隔一段时间，我们就对一些key进行检查，删除里面过期的key(从一定数量的数据库中取出一定数量的随机键进行检查，并删除其中的过期键)。

优点：可以通过限制删除操作执行的时长和频率来减少删除操作对 CPU 的影响。另外定期删除，也能有效释放过期键占用的内存。

缺点：难以确定删除操作执行的时长和频率。如果执行的太频繁，定期删除策略变得和定时删除策略一样，对CPU不友好。如果执行的太少，那又和惰性删除一样了，过期键占用的内存不会及时得到释放。另外最重要的是，在获取某个键时，如果某个键的过期时间已经到了，但是还没执行定期删除，那么就会返回这个键的值，这是业务不能忍受的错误。

Redis的过期删除策略：惰性删除 + 定期删除两种策略进行配合使用定期删除函数的运行频率，在Redis2.6版本中，规定每秒运行10次，大概100ms运行一次。在Redis2.8版本后，可以通过修改配置文件redis.conf 的 hz 选项来调整这个次数。

# 6、Redis中有哪些数据淘汰策略？

数据的淘汰策略：当Redis中的内存不够用时，此时再向Redis中添加新的key，那么Redis就会按照某一种规则将内存中的数据删除掉，这种数据的删除规则被称之为内存的淘汰策略。

常见的数据淘汰策略：

noeviction # 不删除任何数据，内存不足直接报错(默认策略)volatile-lru # 挑选最近最久使用的数据淘汰(举例：key1是在3s之前访问的, key2是在9s之前访问的，删除的就是key2)volatile-lfu # 挑选最近最少使用数据淘汰 (举例：key1最近5s访问了4次, key2最近5s访问了9次， 删除的就是key1)volatile-ttl # 挑选将要过期的数据淘汰volatile-random # 任意选择数据淘汰allkeys-lru # 挑选最近最少使用的数据淘汰allkeys-lfu # 挑选最近使用次数最少的数据淘汰allkeys-random # 任意选择数据淘汰，相当于随机

注意：

1、不带allkeys字样的淘汰策略是随机从Redis中选择指定的数量的key然后按照对应的淘汰策略进行删除，带allkeys是对所有的key按照对应的淘汰策略进行删除。

2、缓存淘汰策略常见配置项

maxmemory-policy noeviction *# 配置淘汰策略*maxmemory ?mb *# 最大可使用内存，即占用物理内存的比例，默认值为0，表示不限制。生产环境中根据需求设定，通常设置在50%以上。*maxmemory-samples count *# 设置redis需要检查key的个数*

# 7、Redis中数据库默认是多少个db？

Redis默认支持16个数据库，可以通过配置databases来修改这一数字。客户端与Redis建立连接后会自动选择0号数据库，不过可以随时使用select命令更换数据库。

Redis支持多个数据库，并且每个数据库是隔离的不能共享，并且基于单机才有，如果是集群就没有数据库的概念。

# 8、缓存穿透、缓存击穿、缓存雪崩解决方案？

缓存穿透：

概述：指查询一个一定不存在的数据，如果从存储层查不到数据则不写入缓存，这将导致这个不存在的数据每次请求都要到 DB 去查询，可能导致 DB 挂掉。

解决方案：

1、查询返回的数据为空，仍把这个空结果进行缓存，但过期时间会比较短

2、布隆过滤器：将所有可能存在的数据哈希到一个足够大的 bitmap 中，一个一定不存在的数据会被这个 bitmap 拦截掉，从而避免了对DB的查询

缓存击穿：

概述：对于设置了过期时间的key，缓存在某个时间点过期的时候，恰好这时间点对这个Key有大量的并发请求过来，这些请求发现缓存过期一般都会从后端 DB 加载数据并回设到缓存，这个时候大并发的请求可能会瞬间把 DB 压垮。

解决方案：

1、使用互斥锁：当缓存失效时，不立即去load db，先使用如 Redis 的 setnx 去设置一个互斥锁，当操作成功返回时再进行 load db的操作并回设缓存，否则重试get缓存的方法

2、永远不过期：不要对这个key设置过期时间

缓存雪崩：

概述：设置缓存时采用了相同的过期时间，导致缓存在某一时刻同时失效，请求全部转发到DB，DB 瞬时压力过重雪崩。与缓存击穿的区别：雪崩是很多key，击穿是某一个key缓存。

解决方案：

将缓存失效时间分散开，比如可以在原有的失效时间基础上增加一个随机值，比如1-5分钟随机，这样每一个缓存的过期时间的重复率就会降低，就很难引发集体失效的事件。

# 9、Redis数据持久化有哪些方式？各自有什么优缺点？

在Redis中提供了两种数据持久化的方式：

1、RDB

2、AOF

RDB：定期更新，定期将Redis中的数据生成的快照同步到磁盘等介质上，磁盘上保存的就是Redis的内存快照

优点：数据文件的大小相比于aop较小，使用rdb进行数据恢复速度较快

缺点：比较耗时，存在丢失数据的风险

AOF：将Redis所执行过的所有指令都记录下来，在下次Redis重启时，只需要执行指令就可以了

优点：数据丢失的风险大大降低了

缺点：数据文件的大小相比于rdb较大，使用aop文件进行数据恢复的时候速度较慢

# 10、Redis都存在哪些集群方案？

在Redis中提供的集群方案总共有三种：

1、主从复制

* 保证高可用性
* 实现故障转移需要手动实现
* 无法实现海量数据存储

2、哨兵模式

* 保证高可用性
* 可以实现自动化的故障转移
* 无法实现海量数据存储

3、Redis分片集群

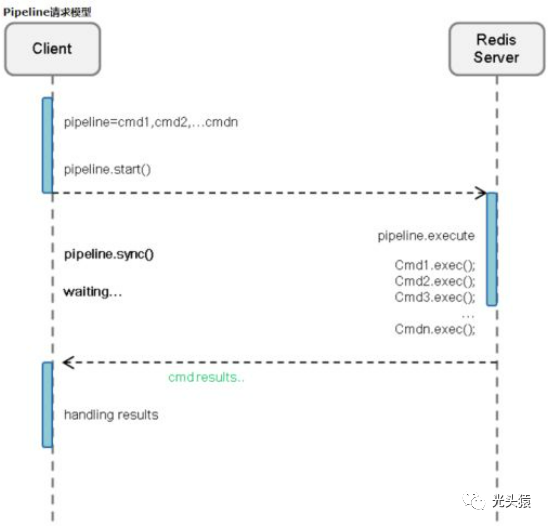
* 保证高可用性
* 可以实现自动化的故障转移
* 可以实现海量数据存储

# 11、说说Redis哈希槽的概念？

Redis 集群没有使用一致性 hash,而是引入了哈希槽的概念，Redis 集群有 16384 个哈希槽，每个 key通过 CRC16 校验后对 16384 取模来决定放置哪个槽，集群的每个节点负责一部分 hash 槽。

**12、Redis中的管道有什么用？**

一次请求/响应服务器能实现处理新的请求即使旧的请求还未被响应，这样就可以将多个命令发送到服务 器，而不用等待回复，最后在一个步骤中读取该答复。



# 13、谈谈你对Redis中事务的理解？

事务是一个原子操作：事务中的命令要么全部被执行，要么全部都不执行。

Redis中的事务：Redis事务的本质是一组命令的集合。事务支持一次执行多个命令，一个事务中所有命令都会被序列化。在事务执行过程，会按照顺序串行化执行队列中的命令，其他客户端提交的命令请求不会插入到事务执行命令序列中。

总结说：Redis事务就是一次性、顺序性、排他性的执行一个队列中的一系列命令。Reids中，单条命令是原子性执行的，但事务不保证原子性，且没有回滚。

# 14、Redis事务相关的命令有哪几个？

事务相关的命令：

1、MULTI：用来组装一个事务

2、EXEC：执行一个事物

3、DISCARD：取消一个事务

4、WATCH：用来监视一些key，一旦这些key在事务执行之前被改变，则取消事务的执行

5、UNWATCH：取消 WATCH 命令对所有key的监视

如下所示：



# 15、Redis如何做内存优化？

尽可能使用散列表（hash），散列表（是说散列表里面存储的数少）使用的内存非常小，所以你应该尽可能的将你的数据模型抽象到一个散列表里面。

比如你的 web 系统中有一个用户对象，不要为这个用户的名称，姓氏，邮箱，密码设置单独的key,而是应该把这个用户的所有信息存储到一张散列表里面。

# 16、Redis是单线的，但是为什么还那么快？

Redis总体快的原因：

1、完全基于内存的

2、采用单线程，避免不必要的上下文切换带来的消耗

3、数据简单，数据操作也相对简单

4、使用多路I/O复用模型，非阻塞IO