## NoSQL数据库简介

### 为什么要使用NoSQL

随着用户访问量大幅度提升，传统的单体应用不能承受住大量用户的访问，

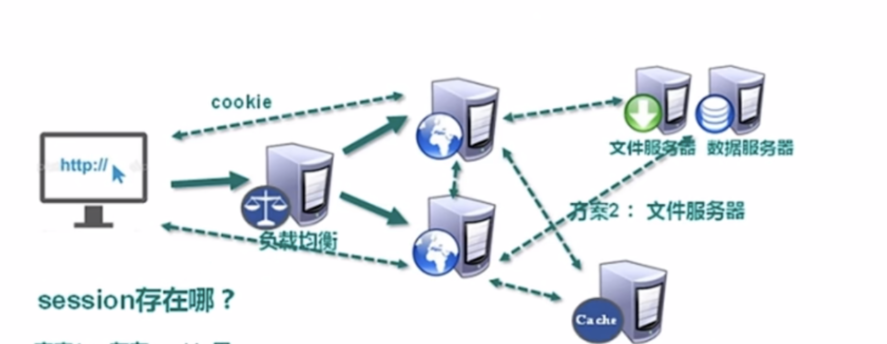
在单体应用中应用服务器的CPU及内存承受着巨大的压力,在访问数据库时，

需要经过IO操作，如果访问量极具增加或者数量量增加，IO压力也会增大。



### 如何解决CPU及内存压力

做分布式部署或集群部署



在分布式或集群情况下，会产生一些问题,如Session存在哪里？

方案1：存在客户端cookie里

1. 不安全
2. 网络负担，效率低

方案2：存放在文件服务器或数据库里

大量的IO效率问题

方案3：session复制

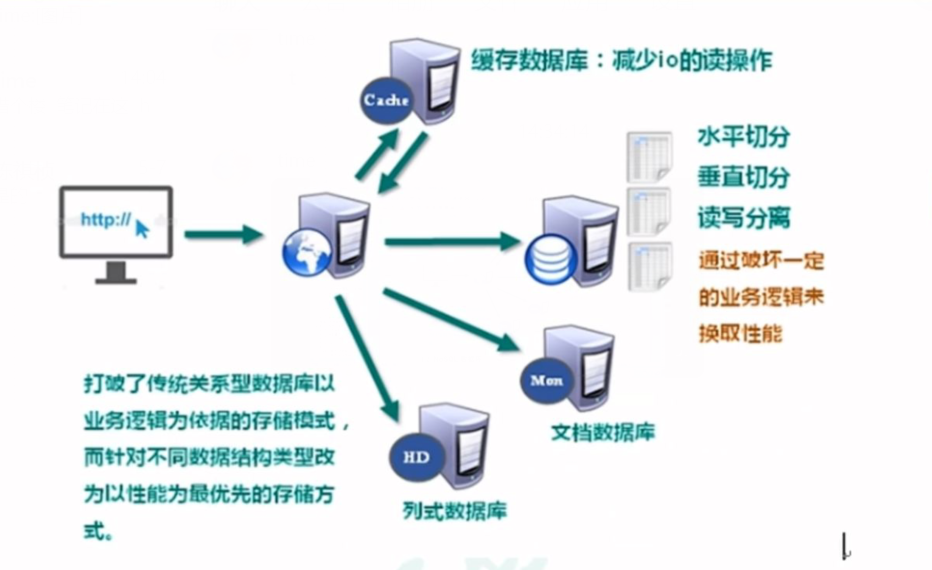
1.Session数据冗余

2.节点越多越浪费越大

方案4：缓存数据库（NoSQL）

1. 完全在内存中；（不需要经过IO）
2. 速度快
3. 数据结构简单

如何解决IO压力



将频繁查询且很少修改的数据放入缓存数据库中，提高数据库查询效率，将一些特有的数据进行列式存储或文档处理;

1. NoSQL能做什么？
2. 降低服务器CPU和内存压力；
3. 降低数据库IO流的压力；

### 什么是NoSQL？

NoSQL(Not Only SQL)非关系型的数据库,NoSQL不依赖业务逻辑方式存储，以简单的key-value模式存储，因此大大增加了数据库的拓展能力。

### NoSQL的特点

1.不遵循SQL标准

2.不支持ACID

3.远超SQl性能

### NoSQL使用场景

对数据高并发的读写

海量数据的读写

### NoSQL不适用场景

需要事务支持

基于SQL的结构查询存储，处理复杂的关系；

### 常见的NoSQL

Memcache

特点：数据存储在内存中，一般不支持持久化；

支持简单的key-value模式，支持类型单一；

一般作为缓存数据库辅助持久化数据库。

Redis

特点：数据存储在内存中，支持持久化，主要用作备份恢复；

处理支持key-value模式，还支持多种数据结构存储，比如list，set，hash,zset等；

一般作为缓存数据库辅助持久化数据库。

MongDB

特点：高性能、开源、模式自由的文档型数据库。

数据都存储在内存中，如果内存不足，把不常用的数据保存到硬盘；

虽然是key-value模式，但是对value（尤其是json）提供了丰富的查询功能；

支持二进制数据和大型对象

## Redis概述

### 什么是Redis

Redis 是一个开源的 key-value 存储系统。支持多种数据结构存储。

### Redis能做什么？

1.配合关系型数据库做高速缓存

a.高频次、热门访问数据，降低数据库的IO

b.分布式架构，做session共享

2.多样的数据机构存储持久化数据

1.最新的N个数据（通过List集合实现）

2.排行榜（通过zset集合）

3.时效性数据，比如手机验证码 （通过expire过期）

4.计数器，秒杀（通过原子性,自增方法incr，decr）

5.去除大量数据中的重复数据（通过set集合）

6.构建队列（通过list集合）

7.发布订阅消息系统（pub/sub模式）

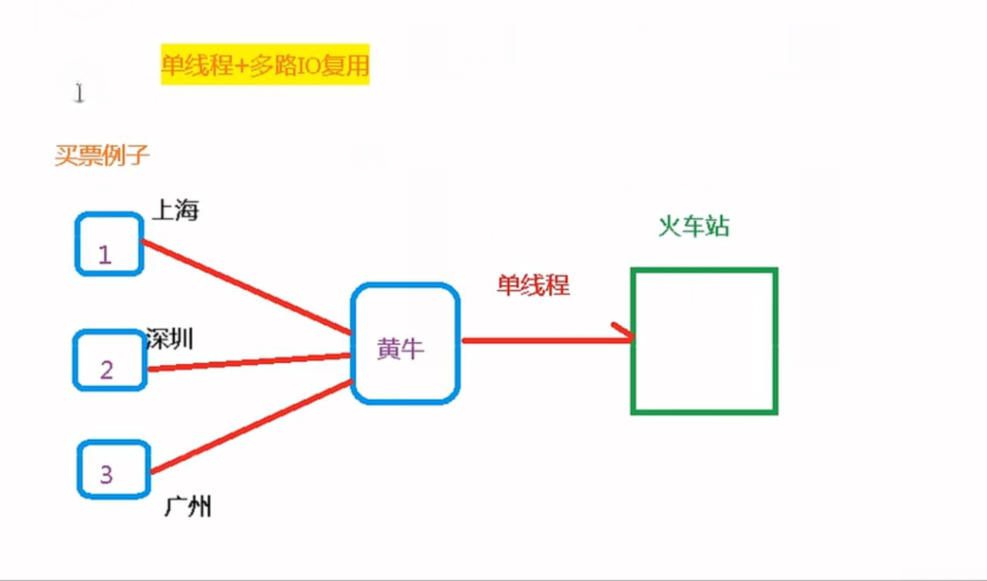
### Redis安装

Redis官网：

<https://redis.io/>

Redis相关知识：

1. 默认16个数据库，类似数组下标从0开始，初始默认使用0号库；使用命令select dbid来切换数据库（select 1），统一密码管理，所有库同样密码；
2. Redis是单线程+多路IO复用技术；



多个人去找黄牛买票，黄牛买好票，通知对应的人取票；

## 常用五大数据类型

### Redis键(key)相关操作

keys \* 查看当前库的所有key

exists key 判断某个key是否存在

type key 查看key的类型

del key 删除指定key的数据

unlink key 根据value选择非阻塞删除

仅将keys从keyspace元数据中删除，真正的删除会在后续异步操作；

expire key 10：给定key的过期时间，单位秒

ttl key 查看还有多少秒过期，-1表示永不过期，-2表示已过期；

select 数据库id：切换数据库

dbsize：查看当前数据库的key的数量

flushdb:清空当前库数据

flushall：清空所有库数据

### 五大数据类型

字符串(String)

列表（List）

集合(Set)

哈希（Hash）

有序集合（Zset）

### 字符串String

#### 基本介绍

1.String是Redis中的基本数据类型，一个key对应一个value；

2.String类型是二进制安全的，Redis中的Stirng可以包含任何类型的数据（图片、序列化对象）

3.一个Redis中字符串value最多可以是512M。

#### 常用命令

set <key> <value> 添加键值对

说明：如果key重复，值会被覆盖

get <key> 查询对应的键值

append <key><value>将给定的<value>追加到原值的末尾

strlen <key> 获取值的长度

setnx <key><value> 只有在key不存在时，设置key的值；

incr <key>将key中储存的数字增1

说明：只能对数字值进行操作，如果为空，新增值为1；

decr <key> 将key中储存的数字值减1

说明：只能对数字值进行操作，如果为空，新增值为-1；

Incryby/decrby <key><步长>将key中储存的数字值增减，自定义步长；

Incr key：对存储在指定key的数值执行原子的加1操作；所谓原子操作是指不会被线程调度机制打断操作；

1. 在单线程中，能够在单条指令中完成的操作都可以认为是“原子操作”，因为中断只会发生在指令之间；
2. 在多线程中，不能被其他进程（线程）打断的操作就叫原子操作。Redis单命令的原子性主要得益于Redis的单线程;

mset <key1><value1><key2><value2>……

说明：同时设置一个或多个key-value对

mget<key1> <key2> ……

说明：同时获取一个或多个value

msetnx <key1><value1><key2><value2>……

说明：同时设置一个或多个key-value对，当且仅当所有给定key都不存在。

原子性，有一个失败则都失败。

getrange <key><起始位置><结束为止>

说明：获取值的范围，类似java中的substr,包前包后；

setrange<key><起始位置><value>

说明：用<value> 覆写<key>所存储的字符串，从<起始位置>开始（索引从0开始）

setex<key> <过期时间><value>

说明：设置键的同时，设置过期时间，单位秒

getset<key><value>

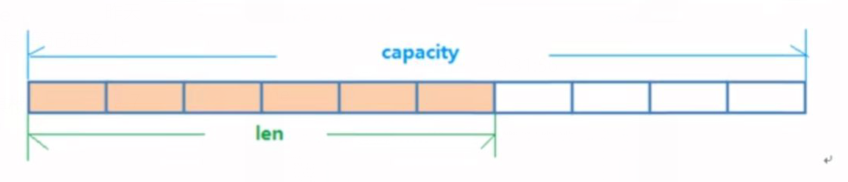
说明：以新换就，设置了新值时获取旧值；

数据结构

String的数据结构为简单动态字符串（Simple Dynamic String，缩写SDS）。

是可以修改的字符串，内部结构实现上类似于java中的arrayList，采用预分配

冗余空间的方式减少内存的频繁分配；



如图所示，内部为当前字符串实际分配的空间capacity一般是高于实际字符串长度len。

当字符串长度小于1M时，扩容都是加倍现有空间，如果超过1M，扩容时一次只会多扩

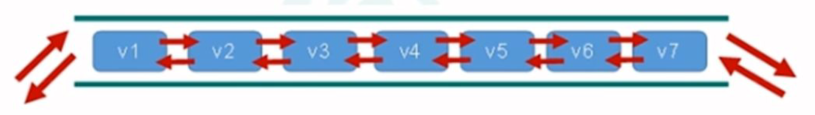
1M的空间，需要注意的是字符串最大长度为512M。

### 列表List

存储的是单键多值的数据；

Redis列表是简单的字符串列表列表，按照插入的顺序排序。可以添加一个元素到列表的头部（左边）或者尾部（右边）。

它的底层实际上个双向列表 ，对两端的操作性能很高，通过索引下标的操作中间的节点性能会较差。



#### 常用命令

lpush/rpush <key> <value1> <value2>……从左边或右边插入一个或多个值

lpop/rpop <key>从左边/右边吐出一个值。值在键在，值光键亡。

rpoppush <key1><key2> 从key1列表右边吐出一个值，插入到key2列表左边；

lrange<key><start><stop> 按照索引下标获得元素（从左到右）

lrange mylist 0 -1 0左边第一个 ；-1是右边第一个，（0 -1表示获取所有）

lindex<key><index> 按照索引下标获取元素(从左到右)

llen<key> 获取列表长度

linsert <key> before <value><newvalue>在<value>的后面插入<newvalue>值。

lrem<key><n><value> 从左边删除n个value（从左到右）

lset<key><index><value>将列表key下标为index的值替换成value

#### List的数据结构

List的数据结构为快速链表quickList

首先在元素较少的情况下会使用一块连续的内存存储，这个结构是ziplist，即是压缩列表。

它将所有的元素紧挨着一起存储，分配的是一块连续的内存，当数据量比较多的时候会改成quicklist。

因为普通的链表需要的附加指针空间太大，会比较浪费空间，比如一个列表存的只是int类型的数据，结构上还需要两个额外的指针prew和next。

Redis将链表和ziplis结合起来组成了quickList。也就是将多个ziplist使用双向指针串起来使用。这样既满足了快速插入删除的心梗，又不会出现太大的空间冗余。



### Set集合

Redis set对外提供的功能和lis相似是一个列表的功能，特殊之处在于set是可以自动排重的，当你需要存储一个列表数据，又不希望出现重复数据时，set一个很好的选择，并且set提供了判断某个成员是否在一个set集合内的重要接口，这是list所不能提供的。

Redis的set是String类型的无序集合，它的底层是一个value为null的hash表，所以添加、删除、查找的复杂度都是最低的；

#### 常用命令

sadd <key><value1> <value2>……

说明：将一个或多个member元素加入到集合key中，已经存在的m素会member被忽略。

smembers <key>

说明：取出改集合的所有值

sismember <key><value>

说明：判断集合<key>是否含有该<value>值，有1，没有0

scard <key>

说明：返回该集合的元素个数；

srem <key> <value1><value2>……

说明：删除集合中的某个元素；

spop <key>

说明：随机从该集合中获取一个值；

srandmember<key><n>

说明：随机从集合中取出n个值，不会从集合中删除；

smove <source><destination>value

说明：把集合中的一个值从一个集合一到另一个集合。

sinter<key1><key2>

说明：返回两个集合的交集元素；

sunion <key1><key2>

说明：返回两个集合的并集

sdiff<key1><key2>

说明：返回两个集合的差集元素（key1中的，不包含key2中的）

#### 数据结构

Set数据结构dict字典，字典是使用哈希表实现的；

Java中的HashSet的内部实现使用的是HashMap，只不过所有的value都指向同一个对象。

Redis的set结构也是一样的，它的内部也是使用了hash结构，所有的value都指向同一个内部值。

### 哈希Hash

Redis hash是一个键值对集合

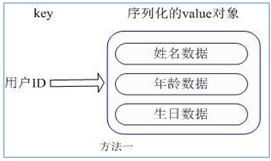
Redis hash是一个String类型的field和value的映射表，hash特别适合用于存储对象；

类似java中的Map<String,Object> 用户ID为查找的key，存储的value为对象；

主要有两种存储方式:

第一种：将对象ID作为key，将对象作为value序列化存放

User:{id=1,name=”张三”}



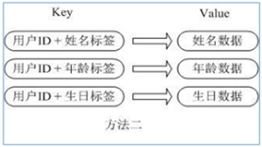
缺点：如果要修改其中的一个属性，需要先反序列化，然后转为对象进行修改，改好后再序列化回去，开销大；

第二种：将数据分开存储

User：id 1

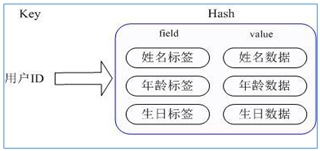
User：name zhangsan

缺点：数据分散，用户ID数据冗余



第三种：hash方式存放

通过key(对象ID)+field（属性标签），就可以操作对应的属性数据了，既不需要重复存储数据，也不会带来序列化和并发修改控制的问题；



#### 常用命令：

hset <key><field><value>

说明：给key设置键值对<field><value>

hget <key><field>

说明：从<key>中取出<field>对应的值；

hmset <key1><field1><value1> <key2><field2><value2> ……

说明：批量设置hash的值

hexists <key><field>

说明：查看哈希表中key中的field是否存在；

hkeys <key>

说明：列出hash中所有的field

Hvals <key>

说明：列出hash中所有的value

hincrby <key><field><increment>

说明：为哈希表<key>中的 <field>的值上增量1或-1

hsetnx<key><field><value>

说明：将哈希表<key>中的<field>的值设置为value，当且仅当field不存在；

#### 数据结构：

Hash类型对应的数据结构是两种：

Ziplist(压缩列表)，hashtable(哈希表)

当filed-value长度较短且个数较少时，使用ziplist，否则使用hashtable。

### 有序集合Zset（sorted set）

Redis有序集合zset与普通的set非常相似，是一个没有重复元素的字符串集合；

不同之处在于有序集合的每个成员都关联了一个评分(score),这个评分是被用来按照从最低分到最高分的方式排序集合中的成员。集合的成员是唯一的，但是评分是可以重复的。

因为元素是有序的，所有可以很快的根据评分（score）或者次序（position）来获取一个范围的元素。

访问有序集合的中间元素也是非常快的，因此可以使用有序集合作为一个没有重复的成员的智能列表。

#### 常用命令：

zadd <key> <score1><value1> <score2><value2>……

说明：将一个或多个member元素及score值加入到有序集key当中；

zrange <key><start><stop>**[ withscores]**

说明：返回有序集合中，下标在<start><stop>之间的元素，带withscores，可以让分数一起和值返回到结果集。

zrangeby score key min max [ withscores] [limit offset count]

说明:返回有序集合key中，所有score值介于min和max之间（包含min和max）的成员。

zrevrange score key min max [ withscores] [limit offset count]

说明:返回有序集合key中，所有score值介于min和max之间（包含min和max）的成员。

从大到小排列；

zincrby <key><increment><value>

说明：为元素的score加上增量

zrem <key><value>

说明：删除该集合下指定值的元素；

zcount <key><min><max>

说明：统计该集合，分数区间内的元素个数；

zrank <key><value>

说明：返回该值在集合中的排名，从0开始；

案例：如何利用zset实现文章访问量的排行榜？

#### 数据结构：

SortedSet (zset) 是 Redis 提供的一个非常特别的数据结构，一方面它等价于 Java 的数据结构 Map<String, Double>，可以给每一个元素 value 赋予一个权重 score，另一方面它又类似于 TreeSet，内部的元素会按照权重 score 进行排序，可以得到每个元素的名次，还可以通过 score 的范围来获取元素的列表。

zset 底层使用了两个数据结构：

hash，hash 的作用就是关联元素 value 和权重 score，保障元素 value 的唯一性，可以通过元素 value 找到相应的 score 值。

跳跃表，跳跃表的目的在于给元素 value 排序，根据 score 的范围获取元素列表。

#### 跳跃表

简介

有序集合在生活中比较常见，例如根据成绩对学生排名，根据得分对玩家排名等。对于有序集合的底层实现，可以用数组、平衡树、链表等。数组不便元素的插入、删除；平衡树或红黑树虽然效率高但结构复杂；链表查询需要遍历所有效率低。Redis 采用的是跳跃表，跳跃表效率堪比红黑树，实现远比红黑树简单。

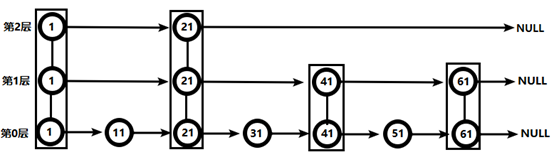
实例

对比有序链表和跳跃表，从链表中查询出 51：

有序链表

image-20210618205641992​ 要查找值为 51 的元素，需要从第一个元素开始依次查找、比较才能找到。共需要 6 次比较。

跳跃表



从第 2 层开始，1 节点比 51 节点小，向后比较；

21 节点比 51 节点小，继续向后比较，后面就是 NULL 了，所以从 21 节点向下到第 1 层；

在第 1 层，41 节点比 51 节点小，继续向后，61 节点比 51 节点大，所以从 41 向下；

在第 0 层，51 节点为要查找的节点，节点被找到，共查找 4 次。

从此可以看出跳跃表比有序链表效率要高

## Redis6配置文件

1. 配置大小单位，支持bytes，不支持bit；
2. 大小写不敏感；

Include部分

一个文件可以包含另外一个配置文件

网络相关配置

1. 默认情况bind = 127.0.0.1只能接收本机的访问请求；
2. 不写的情况下，无限制接收任何ip地址访问；
3. 如果开启了保护模式（protected-mode）,那么在没有设定bind ip 且没有设置密码的情况下，Redis只允许接受本机的响应；
4. tcp-backlog

设置tcpde backlog,backlog其实是一个连接队列，在高并发环境下需要设置一个backlog值来避免慢客户端连接问题；

1. timeout 超时时间
2. tcp-keepalive 检查心跳时间

## Redis的发布和订阅

什么是发布和订阅

Redis 发布订阅 (pub/sub) 是一种消息通信模式：发送者 (pub) 发送消息，订阅者 (sub) 接收消息。

Redis 客户端可以订阅任意数量的频道。

Redis中的新数据类型

BitMaps