# Redis学习

## 入门

1、单机mysql的美好年代

2、Memcached(缓存)+Mysqk+垂直拆分

3、Mysql主从读写分离

4、分表分库+水平拆分+mysql集群

5、mysql 的扩展性瓶颈

6、今天是什么样子

7、为什么用NoSql

### 易扩展

NOSQL数据库种类繁多，但是一个共同的特点都是去掉关系数据库关系特性

### 大数据量高性能

NoSql数据库具有非常高的读写性能，尤其在大数据量下，同样表现优秀

这得益于它的无关系性，数据库的结构简单

一般Mysql使用query cache ,每次表的更新cache 就失效，是一种大粒度的cache

在针对web2.0 的交互频繁的应用，Cache性能不高。而Nosql的Cache是记录级的

是一种细粒度的Cache,所以NoSql 在这个层面上来说就是性能高很多

### 多样灵活的数据模型

NoSql 无需事项为存储的数据建立字段，随时可以存储自定义

### 3V+3高

大数据时代的3v

海量volume

多样variety

实时velocity

互联网需要的3高

高并发、

高扩展、

高性能

### 经典应用

1、商品的基本信息

名称、价格、出厂日期、生产厂商

冷数据存在mysql

2、商品描述、详情、评价信息（多文字类）

大文字类mongo

3、商品的图片

商品图片展现类

分布式的文件系统中 淘宝自己的TFS 、谷歌的GFS Hadoop 的GDFS

4、商品的关键字

多隆ISearch、搜索引擎、淘宝内用

5、商品的波段性的热点高频信息

内存数据库

tair 、redis

6、商品的交易、价格计算、积分累计

7、总结大型互联网应用（大数据、高并发、难点与解决方案）

统一数据平台服务层

UDSL

### NOSQL 数据模型简介

以一个电商客户，订单、订购、地址模型来设计关系型数据库和非关系型数据库

聚合模型

Nosql你如何设计

什么是BSON

给学生用BSON患处构建的数据模型

高并发的操作是不太建议有关联查询

互联网公式冗余数据来避免关联查询

### 聚合模型

Kv键值 （典型介绍）

新浪：berkeleyDB+redis

美团：redis+tair

阿里、百度：memcache+redis

BSON ( 典型介绍)

couchdb

mongoDb

列族

cassandra,Hbase

分布式文件系统

图形

社交网络

推荐系统

Neo4j ,infoGrid

### NoSql数据库的四大分类

### CAP+BASE

1、传统的ACID分别是什么

A(Atomicity)原子性

C(Consistency)一致性

I(Isolation)独立性

D(Durability) 持久性

2、CAP

C：Consistency(强一致性)

A：Availability(可用性)

P:Partition tolerance(分区容错性)

### CAP的3进2

Cap理论就是说在分布式存储系统中，最多实现上面的俩点而由于当前的网络硬件肯定会出现延迟丢包等问题

所以分区容错性我们必须需要实现的

所以我们只能在一致性和可用性之间进行权衡，没有NoSql系统能同时保证这三点

C:强一直性、A、高可用性，P分布式容忍性

CA 传统Oracle 数据库

Ap 大多数网站架构的选择

CP Redis 、Mongodb

### 经典CAP图

### BASE

BASE就是为了解决关系数据库强一致性引起的问题而引起的可用用性降低而提出的解决方案

BASE其实是下面三个术语的缩写

基本可用（BASICALLY Available）

软状态（Soft state）

最终一致（Eeventually consistent）

### 分布式+集群

1、分布式：不同的堕胎服务器上面部署不同的服务模块（工程），他们之间通过Rpc/Rmi之间的通信和调用，对外提供服务和组内协作

2、集群：不同的多台服务器上面部署相同的服务模块，通过分布式调度软件进行统一的调度，对外提供服务和访问

### Redis 入门介绍

Redis :Remote DICtionary Server(远程字典服务器)

是完全开源免费的，用C语言编写的，遵守BSD协议

三个特点

Redis 支持数据的持久化，可以将内存中的数据保持在磁盘中，重启的时候可以再次加载进行使用

Redis不仅仅支持简单的key-value类型的数据，同时还提供list ,set,zset,hash等数据结构的存储

Redis支持数据的备份，即master-slave模式的数据备份

能干吗

内存存储和持久化：redis 支持一部将内中的数据写到硬盘上，同时不影响继续服务

取最新的N个数据的操作，如：可以将最新的10条评论的ID放到Redis的List集合里面

模拟类似于HttpSession这种需要设定过期时间的功能

发布、订阅消息系统

定时器、计数器

去哪下

http://redis.io/

http://www.redis.cn/

/opt目录下，解压命令：tar-zxvf redis-3.0.4.tag.gz

解压完成后出现文件夹 redis-3.0.4

进入目录：cd redis-3.0.4

在redis-3.0.4目录下执行make命令

如果make完成后继续执行make install

查看默认安装目录：usr/local/bin

启动

永远的helloworld

关闭

wget http://download.redis.io/releases/redis-6.0.6.tar.gz

$ tar xzf redis-6.0.6.tar.gz

$ cd redis-6.0.6

$ make

依赖gcc

sudo yum install centos-release-scl

sudo yum install devtoolset-7-gcc\*

scl enable devtoolset-7 bash

报错处理执行

make MALLOC=libc

加载

make install

cd /usr/local/bin

cd /usr/local/bin

切换到redis bin目录下

ls -l 查看文件列表

启动redis-sever 命令：redis-sever /myredis/redis.conf(j加载指定要加载的配置文件)

redis-cli -p 6379

对应的版本号：

[https://github.com/redis/redis-hashes/blob/master/README](https://github.com/redis/redis-hashes/blob/master/README" \t "C:/Users/57210/AppData/Roaming/Foxmail7/Temp-12904-20211205113254/_blank)

## 杂项知识学习

1、单进程

单进程模型来处理客户端的请求，对读写等事件的响应

是通过对epoll函数的包装来做到的，Redis的实际处理速度完全依靠主进程的执行效率

Epoll是linux内核为处理大批量文件描述符而做了改进的epoll,是linux下多路复用IO接口select/poll的增强版本，它能显著提高程序在大量并发连接中只有少量活跃的情况下的系统CPU利用率

2、默认16个数据库，类似数组下标从零开始，初始默认使用零号库

命令：set k1 "111"

get k1

select [0-15]

3、select 命令切换数据库

4、Dbsize 查看当前数据库的key的数量

dbsize

key\*

key k ?

5、Flushdb:清空当前库

6、Flushall:通杀全部库

7、统一密码管理，16个库都是同样的密码，要么都OK,y要么一个也连接不上

8、Redis索引都是从零开始

9、为什么默认端口是6379

### redis常见数据类型命令

<http://redisdoc.com/>

## string（字符串）

string 是redis 最基本的类型，你可以理解成和memcached一模一样的类型，一个key对应一个value

string 类型是二进制安全的，意思是redis 的string可以包含任何数据，比如jpg图片或者序列化的对象

string类型是Redis最基本的数据类型，一个redis中字符串最多可以是512M

#### 案例

set/get/del/append/strlen

127.0.0.1:6379[1]> clear

127.0.0.1:6379[1]> get k1

(nil)

127.0.0.1:6379[1]>

127.0.0.1:6379[1]>

127.0.0.1:6379[1]>

127.0.0.1:6379[1]> get k1

(nil)

127.0.0.1:6379[1]> set k1 ty

OK

127.0.0.1:6379[1]> get k1

"ty"

127.0.0.1:6379[1]> append k1 12345

(integer) 7

127.0.0.1:6379[1]> get k1

"ty12345"

127.0.0.1:6379[1]> STRLEN k1

incr/decr/incrby/decrby,一定要是数字才能进行加减

127.0.0.1:6379> incrby k1 3

(integer) 22

127.0.0.1:6379> get k1

"22"

127.0.0.1:6379> decrby k1 2

getrange/setrange 字符串的设置和取值

127.0.0.1:6379> GETRANGE k1 0 -1

"14"

127.0.0.1:6379> GETRANGE k4 0 1

setex(set with expire) 键秒值/setnx(set if not exist) 不存在设置

127.0.0.1:6379> flushdb

OK

127.0.0.1:6379> setex k1 10 1 // 设置有效期

OK

127.0.0.1:6379> setex k1 10 1

OK

127.0.0.1:6379> setex k1 10 2

OK

127.0.0.1:6379> get k1

"2"

127.0.0.1:6379> set k2 2

OK

127.0.0.1:6379> setex k2 20 1

OK

127.0.0.1:6379> get k2

"1"

127.0.0.1:6379> set k3 3

OK

127.0.0.1:6379> setnx k3 2// 已存在不改变

(integer) 0

127.0.0.1:6379> get k3

"3"

127.0.0.1:6379> setnx k4 1// 不存在才设置

(integer) 1

127.0.0.1:6379> get k4

"1"

mset/mget/msetnx

127.0.0.1:6379> mset k1 1 k2 2 k3 3

OK

127.0.0.1:6379> mget k1 k2 k3

1) "1"

2) "2"

3) "3"

127.0.0.1:6379> msetnx k1 2 k2 1

(integer) 0

127.0.0.1:6379> msetnx k4 4 k5 5

(integer) 0

127.0.0.1:6379> key \*

(error) ERR unknown command `key`, with args beginning with: `\*`,

127.0.0.1:6379> keys \*

1) "k4"

2) "k3"

3) "k2"

4) "k1"

127.0.0.1:6379> msetnx k5 1 k6 6

(integer) 1

127.0.0.1:6379> keys \*

1) "k3"

2) "k1"

3) "k5"

4) "k6"

5) "k4"

6) "k2"

getset(先get在set)

## Hash

Hash(哈希)

Redis hash键值对集合

Redis hash 是一个string 类型的field和value的映射表，hash特表适合用于存储对象

KV 模式不变，但是V是一个键值对

常见命令：

hset/hget/hmget/hdel

hlen

hexists key 在key里面的的某个值得key

hincrby/hincrbyfloat

hsetnx

127.0.0.1:6379> hset h1 people name

(integer) 1

127.0.0.1:6379> hget h1 people

"name"

127.0.0.1:6379> hmget h1 people

1) "name"

127.0.0.1:6379> hdel h1 people

(integer) 1

127.0.0.1:6379> hget h1 people

(nil)

127.0.0.1:6379> hmget h1 people

1) (nil)

127.0.0.1:6379> helen h1

(error) ERR unknown command `helen`, with args beginning with: `h1`,

127.0.0.1:6379> hlen h1

(integer) 0

127.0.0.1:6379> hset h1 people name

(integer) 1

127.0.0.1:6379> hexists h1 people

(integer) 1

127.0.0.1:6379> hset people age 1

(integer) 1

127.0.0.1:6379> hincrby pepople age 2

(integer) 2

127.0.0.1:6379> hincrby pepople age 2

(integer) 4

127.0.0.1:6379> hincrby pepople age 2

(integer) 6

127.0.0.1:6379> hincrby pepople age 2

(integer) 8

127.0.0.1:6379> hincrby pepople age 2

(integer) 10

127.0.0.1:6379> hgdetail people

(error) ERR unknown command `hgdetail`, with args beginning with: `people`,

127.0.0.1:6379> HGETall people

1) "age"

2) "1"

127.0.0.1:6379> hsetnx people age 2

(integer) 0

127.0.0.1:6379> hsetnx people wight 4.5

(integer) 1

127.0.0.1:6379> hincrbyfloat people wight 5

"9.5"

## List(l列表)

Redis列表是简单的字符串列表，按照插入顺序排序，你可以添加一个元素到列表的头部（左边）或者尾部（右边）他的底层实际是个链表

单值多value

lpush/rpush/lrange

lpop/rpop

lindex,按照索引下标获得元素（从上到下）

llen

lrem key 删除N个value

Itrim key 开始index结束index，截取指定范围内的值后再赋值给key

rpoplpush 源列表 目的列表

lset key index value

linsert key before/after 值1 值2

性能总结：

它是一个字符串链表，left,right 都可以插入添加

如果键不存在，创建新的链表

如果键已存在，新增内容

如果全移除，对应的键也就消失了

链表的操作无论是头和尾效率都极高，但假如是对中间元素进行操作，效率就很低了

127.0.0.1:6379> lpush list1 1 2 3 3 4 6

(integer) 6

127.0.0.1:6379> rpush list2 1 2 3 3 4 6

(integer) 6

127.0.0.1:6379> lrange list1 0 -1

1) "6"

2) "4"

3) "3"

4) "3"

5) "2"

6) "1"

127.0.0.1:6379> lrange list2 0 -1

1) "1"

2) "2"

3) "3"

4) "3"

5) "4"

6) "6"

127.0.0.1:6379> lpop list1

"6"

127.0.0.1:6379> lrange list1 0 -1

1) "4"

2) "3"

3) "3"

4) "2"

5) "1"

127.0.0.1:6379> rpop list 1

(error) ERR wrong number of arguments for 'rpop' command

127.0.0.1:6379> rpop list1

"1"

127.0.0.1:6379> lrange list1 0 -1

1) "4"

2) "3"

3) "3"

4) "2"

127.0.0.1:6379> lindex list1 2

"3"

127.0.0.1:6379> llen list1

(integer) 4

127.0.0.1:6379> lrem list1 2 3

(integer) 2

127.0.0.1:6379> lrange list1 0 -1

1) "4"

2) "2"

127.0.0.1:6379> lrange list2 0 -2

1) "1"

2) "2"

3) "3"

4) "3"

5) "4"

127.0.0.1:6379> ltrim list2 1 3

OK

127.0.0.1:6379> lrange list2 0 -2

1) "2"

2) "3"

127.0.0.1:6379> lrange list1 0 -1

1) "4"

2) "2"

127.0.0.1:6379> rpoplpush list2 list1

"3"

127.0.0.1:6379> lrange list1 0 -1

1) "3"

2) "4"

3) "2"

127.0.0.1:6379> lset list1 0 1

OK

127.0.0.1:6379> lrange list1 0 -1

1) "1"

2) "4"

3) "2"

127.0.0.1:6379> linsert list1 before 5 4

(integer) -1

127.0.0.1:6379> linsert list1 before 5 4

(integer) -1

127.0.0.1:6379> lrange list1 0 -1

1) "1"

2) "4"

3) "2"

127.0.0.1:6379> linsert list1 after 5 4

(integer) -1

127.0.0.1:6379> lrange list1 0 -1

1) "1"

2) "4"

3) "2"

127.0.0.1:6379> linsert list1 before 5 2

(integer) -1

127.0.0.1:6379> linsert list1 before 5 0

(integer) -1

127.0.0.1:6379> RPUSH mylist "Hello"

(integer) 1

127.0.0.1:6379> RPUSH mylist "World"

(integer) 2

127.0.0.1:6379> LINSERT mylist BEFORE "World" "There"

(integer) 3

127.0.0.1:6379> LRANGE mylist 0 -1

1) "Hello"

2) "There"

3) "World"

127.0.0.1:6379> LINSERT mylist BEFORE "go" "let's"

(integer) -1

127.0.0.1:6379> EXISTS fake\_list

(integer) 0

127.0.0.1:6379> LINSERT fake\_list BEFORE "nono" "gogogog"

## set(集合)

无序无重复

单值多value

sadd/smembers/sismember

scard，获取集合中的元素个数

srem key value 删除集合中的元素

srandmember key 某个整数（随机出几个值）

spop key 随机出栈

smove k1 k2 zai k1中的某个值，作用是将k1中的某个值赋值给k2

数学集合类

差集sdiff

交集sinter

并集sunion

sismember:

O(1) 判断 member 元素是否集合 key 的成员。 返回值 如果 member 元素是集合的成员，返回 1 。 如果 member 元素不是集合的成员，或 key 不存在，返回 0 。

SMOVE source destination member 可用版本： >= 1.0.0 时间复杂度: O(1) 将 member 元素从 source 集合移动到 destination 集合。 SMOVE 是原子性操作。 如果 source 集合不存在或不包含指定的 member 元素，则 SMOVE 命令不执行任何操作，仅返回 0 。否则， member 元素从 source 集合中被移除，并添加到...

127.0.0.1:6379> sadd set1 1 2 2 2 2 3 4 5

(integer) 5

127.0.0.1:6379> SMEMBERS set1

1) "1"

2) "2"

3) "3"

4) "4"

5) "5"

127.0.0.1:6379> SISMEMBER set1 3

(integer) 1

127.0.0.1:6379> SMEMBERS set1

1) "1"

2) "2"

3) "3"

4) "4"

5) "5"

127.0.0.1:6379>

[shen@localhost ~]$ redis-cli

127.0.0.1:6379> scard set1

(integer) 5

127.0.0.1:6379> srem set1 3

(integer) 1

127.0.0.1:6379> srandmember set1 3

1) "5"

2) "2"

3) "1"

127.0.0.1:6379> SMEMBERS set1

1) "1"

2) "2"

3) "4"

4) "5"

127.0.0.1:6379> spop set1 1

1) "4"

127.0.0.1:6379> SMEMBERS set1

1) "1"

2) "2"

3) "5"

127.0.0.1:6379> sadd set2 1 2 3 a b

(integer) 5

127.0.0.1:6379> smove set1 set2 1

(integer) 1

127.0.0.1:6379> smebers set2

(error) ERR unknown command `smebers`, with args beginning with: `set2`,

127.0.0.1:6379> SMEMBERS SET2

(empty array)

127.0.0.1:6379> SMEMBERS set2

1) "1"

2) "a"

3) "3"

4) "b"

5) "2"

127.0.0.1:6379> smembers set1

1) "2"

2) "5"

127.0.0.1:6379> sdiff set1 set2

1) "5"

127.0.0.1:6379> sinter set1 set2

1) "2"

127.0.0.1:6379> sunion set1 set2

1) "5"

2) "1"

3) "b"

4) "2"

5) "a"

6) "3"

## zset(集合)

zset（sorted set:有序集合)

Redis zset 和set一样也是string 类型的元素结合，且不允许重复的成员

不同的是每个元素都会关联一个double类型的分数

redis正式通过分数来为集合中的成员进行从小到大的排序

zset的成员是唯一的，但分数（score)却可以重复

有序无重复

zadd/zrange

zrangebyscore key 开始score结束score

zrem key 某score下对用的value,作用是删除元素

zcard/zcount key score区间/zrank key values 值，作用是获得下标值/zscore key 对应值，获取分数

zrevrank key values值，作用是逆序获取下标值

zrevrange

zrevrangebyscore key 结束score 开始score

zadd zkey1 10 v1 20 v2 30 v3

(integer) 3

127.0.0.1:6379> zrange zkey1

(error) ERR wrong number of arguments for 'zrange' command

127.0.0.1:6379> zrange zkey1 0 -1

1) "v1"

2) "v2"

3) "v3"

127.0.0.1:6379> zscore zkey1 20

(nil)

127.0.0.1:6379> zscore zkey1 v2

"20"

127.0.0.1:6379> zrangebyscore zkey1 20 30

1) "v2"

2) "v3"

127.0.0.1:6379> zrem zkey1 v2

(integer) 1

127.0.0.1:6379> zrange zkey1 0 -1

1) "v1"

2) "v3"

127.0.0.1:6379> zcard zkey1

(integer) 2

127.0.0.1:6379> zcard zkey1

(integer) 2

127.0.0.1:6379> zscore zkey1 v1

"10"

127.0.0.1:6379> zrevrange zkey1 0 -1

1) "v3"

2) "v1"

127.0.0.1:6379> zrevrank zkey1 v3

(integer) 0

127.0.0.1:6379> ZREVRANGEBYSCORE zkey1 30 20