NoSql

RDBMS (Relational DataBase Management System) 关系数据库管理系统

按照预先设置的组织结构,将数据存储在物理介质上

数据之间可以做关联操作

RDBMS软件 (主流)

Mysql

MariaDB

Oracle

DB2

SQL server

NoSql (NoSql = Not Only SQL)

--意思是不仅仅是sql

--泛指非关系型数据库

--不需要预先定义数据存储结构

--每条记录可以有不同的数据类型和字段类型

NoSql软件 (主流软件)

Memcached

Redis

MongoDB

CouchDB

Neo4j

FlockDB

Redis介绍

---Remota Dicionary Server (远程字典服务器)

---是一款高性能的分布式内存数据库

--支持数据持久化(定期把内存里的数据存储到硬盘)

--支持多种数据类型 string list hash set zset

--支持master-slave模式数据备份

Redis数据类型:

String（字符串）: string 类型是 Redis 最基本的数据类型，string 类型的值最大能存储 512MB (一个 key 对应一个 value)

List（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）

Hash（哈希）: Redis hash 是一个键值(key=>value)对集合;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

Set（集合）: Redis 的 Set 是 string 类型的无序集合,集合是通过哈希表实现的，所以添加，删除，查找的复杂度都是 O(1)

zset(sorted set：有序集合): Redis zset 和 set 一样也是string类型元素的集合,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正是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

配置服务器运行参数

]# ./utils/install\_server.sh //初始化配置

---端口 6379

---主配置文件 /etc/redis/6379.conf

---日志文件 /var/log/redis\_6379.log

---数据库目录 /var/lib/redis/6379

---服务器启动程序 /usr/local/bin/redis-server

---命令行连接命令 /usr/local/bin/redis-cli

管理服务

]# /etc/init.d/redis\_6379 stop //停止服务

Redis pid文件: /var/run/redis\_6379.pid

]# /etc/init.d/redis\_6379 start //启动服务

!!!若修改了 ip(默认回环地址127.0.0.1) 端口号 (默认6379) 添加了密码(requirepass) 都无法通过默认脚本关闭与开启

!!!若仍要通过脚本管控 可以修改脚本参数

43 $CLIEXEC -p 6350 -h 192.168.4.50 -a 123456 shutdown

]# ps -C redis-server //查看进程

]# netstat -ntulp|grep :6379 //查看端口

常用命令

mset key 值 .. //存储多个key

mget //获取多个key值

select 数据库编号0-15 //切换库

exists key名 //测试key名是否存在

ttl key名 //查看key的生存时间 -2 永久生效 ; -1 已经失效

type key名 //查看key的类型

move key名 库编号 //移动key到指定库

expire key名 数字 //设置key的有效时间

del key名 //删除指定的key

flushall //删除内存里所有的key

flushdb //删除所有库的所有key

save //保存所有key到硬盘

内存管理

内存清楚策略

-volatile-lru //最近最少使用(针对设置了TTL的key)

-allkeys-lru //删除最少使用的key(针对所有的key)

-allkeys-lfu //从所有key中清楚使用频率最少的key

-volatile-lfu //从所有配置了过期时间的key中清除使用频率最少的key

-volatile-random //在设置了TTL的key里随机移除

-allkeys-random //随意移除key

-volatile-ttl(minor TTL) //移除最近过期的key

-noeviction //不删除,写满时报错

优化设置

--maxmemory //最大内存

--maxmemory-policy //定义使用策略

--maxmemory-samples //选取key模板的个数(针对lru和ttl的策略)

phpize 生成配置文件与 ./configure 命令

######################################################################################################

Redis 集群

1.加入的集群的Redis需要开启集群功能 (集群通信端口 1XXXX(原端口的基础上 加 1万))

2.连接集群中任意服务器必须加 -c 参数(包括本地登录),否则获取集群中数据会报错

3.Master宕机后对应的slave自动被选举为Master,原Master启动后会自动配置为当前Master的Slave,并且会自动同步当前Master的数据

4.工作原理:存储规则:crc16 算法 与 变量名 进行 hash 计算 得出的数字 与 总槽数(0-16383个)取余 获得一个槽值然后对应存储

5.添加时不指定主机角色,默认为Master,并且需要手动分配hash槽

6.添加Slave主机时不指定Master节点的id默认把新节点随机添加到从节点最少的主库,[--Master-id id值]参数指定主机

7.添加服务器时集群信息必须为空(rm -rf /var/lib/redis/6379/\* 或者进入Redis数据库后执行 cluster reset 清空集群信息) (存在nodes-6379.conf文件记录了老的集群信息)

8.移除Slave服务器 (从服务器没有hash槽,直接移除即可,移除时指定从服务器id值,会停止移除主机的Redis服务)

9.移除Master服务器,必须先释放战友的hash槽,然后移除主机

10.若当前集群配置不平衡(例如Slave分布状态 1 0 0 2 ),不指定Master添加服务器时(Redis-trib.rb add-node --slave ip:port ip:port),会重新分配当前集群配置(Slave分布状态为 1 1 1 1).

11.查看当前数据库的主从信息命令:info replication

1.创建集群

redis-trib.rb create --replicas 1 (定义每台主库的从库个数,不加此参数 所有的服务器都为Master主服务) 192.168.4.51:6351....(等)

2.查看集群信息(在管理主机查看)

redis-trib.rb info 192.168.4.51:6351(集群中任意一台) //查看集群信息

redis-trib.rb check 192.168.4.51:6351(集群中任意一台) //查看集群主机对应角色

3.添加master服务器(添加时不指定主机角色,默认为Master,并且需要手动分配hash槽)

redis-trib.rb add-node 192.168.4.58:6358(新的开启集群功能的服务器) 192.168.4.51:6351(集群中任意一台)

redis-trib.rb reshard 192.168.4.51:6351(集群中任意一台) //重新分片

--移出的hash槽个数

--接受的hash槽主机id

--移出的hash槽主机id

How many slots do you want to move (from 1 to 16384)? 4096 //移出的hash槽个数

What is the receiving node ID? f9c26dafbfc85e29f16c639351cd4b7e4709d5d7 //接受的hash槽主机id(check命令查看)

Please enter all the source node IDs.

Type 'all' to use all the nodes as source nodes for the hash slots.

Type 'done' once you entered all the source nodes IDs.

Source node #1:all //移出的hash槽主机id (全部 all)

4添加从服务器(添加Slave主机时不指定Master节点的id默认把新节点随机添加到从节点最少的主库,[--Master-id id值]参数指定主机)

redis-trib.rb add-node --slave --master-id f9c26dafbfc85e29f16c639351cd4b7e4709d5d7 192.168.4.59:6359 192.168.4.51:6351(任意一台)

5移除Slave服务器 (从服务器没有hash槽,直接移除即可,移除时指定从服务器id值,会停止移除主机的Redis服务)

redis-trib.rb del-node 192.168.4.51:6351(集群中任意一台) 4807e16c8dd6395944665a1554753d96ec4dbe2a(对应的id值 check查看)

6移除Master服务器(必须先释放战友的hash槽,然后移除主机)

redis-trib.rb reshard 192.168.4.51:6351(任意一台)

--移出的hash槽个数

--接受的hash槽主机id

--移出的hash槽主机id

How many slots do you want to move (from 1 to 16384)? 4096 //移出的hash槽个数

What is the receiving node ID? bc5c4e082a5a3391b634cf433a6486c867cfc44b //接受的hash槽主机id(check命令查看) //要移动给谁的id即目标主机（这里可以随机写一个master的ID）

Please enter all the source node IDs.

Type 'all' to use all the nodes as source nodes for the hash slots.

Type 'done' once you entered all the source nodes IDs.

Source node #1:f9c26dafbfc85e29f16c639351cd4b7e4709d5d7 //指定移出的Master主机id

Source node #2:doen

############################################################################################################################################################

Redis 主从复制

工作原理

--Slave向Master发送sync命令

--Master启动后台存盘进程,并收集所有的修改数据命令

--Master完成后台存盘后(文件),传送整个数据文件到Salve

--Slave接受数据文件,加载到内存中完成首次完全同步

--后续有新数据产生时,Master继续收集数据修改命令依次传给Slave,完成同步

1.所有Redis服务器默认都是主服务器

2.从服务器首次同步主服务器时,会覆盖本地数据

3.哨兵服务只监视主服务器,宕机的主服务器修复重新开启后,会做为当前主服务器的从服务器

4.若开启了aof功能,开启Redis数据库服务后,优先读aof (首次通过配置文件开启aof功能,重启后会清空数据库,因为aof文件为空文件)

配置从库

1.命令行修改 (临时)

192.168.4.52:6352> slaveof host port //设置主服务器

192.168.4.52:6352> slaveof no one //清空主从关系

192.168.4.52:6352> config set requirepass 123456 //设置登录密码

192.168.4.52:6352> config set Masterauth 123456 //设置连接主服务器使用密码

2.修改配置文件(永久)

192.168.4.52:6352> config rewrite //命令行配置写入

vim /etc/redis/6379.conf

slave host port

哨兵服务

--监视Master服务器

--发现Master宕机后,将从服务器升级成主服务器

--主配置文件 sentinel.conf

--模板文件:redis-4.0.8/sentinel.conf

# vim /etc/sentinel.conf //创建主配置文件

sentinel monitor server51(定义主机名) 192.168.4.51 6351 1 //监视主服务器

bind 0.0.0.0 //哨兵服务地址（表示本机所有网络接口）

sentinel auth-pass server51(与监视主机名一致) 123456 //主服务器密码

RDB介绍

Redis数据库文件,全称Redis DataBase

--数据持久化方式之一

--数据持久化默认方式

--按照指定时间间隔,将内存中的数据集快照写入硬盘

--快照术语叫snapshot

--恢复时,将快照文件直接读入内存

定义RDB文件名

- dbfilename "dump.rdb"

RDB优点与缺点

优点

--高性能持久化实现----创建一个子进程来执行持久化,先将数据写入临时文件,持久化过程结束后,再用这个临时文件替换上次持久化好的文件;过程中主进程不做任何IQ操作

--比较适合大规模数据恢复,且对数据完整性要求不是非常高的场合

缺点

--意外宕机时,丢失最后一次持久化的所有数据

AOF介绍

Append Only File

--追加方式记录写操作的文件

--记录Redis服务所有的写操作

--不断将新的写操作,追加到文件的末尾

--默认没有启用

--使用cat命令可以查看文件内容

AOF优点与缺点

AOF优点

--可以灵活设置持久化方式(always everysec no)

--出现意外宕机时,仅可能丢失一秒的数据(默认AOF文件记录写操作方式 everysec)

AOF缺点

--持久化文件的体积通常会大于RDB方式

--执行fsync策略时的速度可能会比RDB方式慢

启用AOF

config set appendonly yes

config rewrite

AOF日志文件触发重写

auto-aof-rewrite-min-size 64mb //首次重写触发值

auto-aof-rewrite-percentage 100 //再次重写,增长百分比

AOF文件修复

redis-check-aof --fix appendonly.aof

Redis数据类型:

String（字符串）: string 类型是 Redis 最基本的数据类型，string 类型的值最大能存储 512MB (一个 key 对应一个 value)

List（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）

Hash（哈希）: Redis hash 是一个键值(key=>value)对集合;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

Set（集合）: Redis 的 Set 是 string 类型的无序集合,集合是通过哈希表实现的，所以添加，删除，查找的复杂度都是 O(1)

zset(sorted set：有序集合): Redis zset 和 set 一样也是string类型元素的集合,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正是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

string 字符串 (0代表第一个字符)

set x 9 ex 10 //单位秒

set y 29 px 10 //单位毫秒

set y 39 NX //不存在赋值

set y 49 xx //变量存在赋值

strlen first //统计字串长度

set first "hello world"

setrange first 6 "Redis" //从第6个字符改写(第一个字符为0)

append myname jacob //append key value 存在则追加，不存在则创建key及value，返回key长度

setbit key offset value

setbit bit 0 1

对key所存储字串，设置或清除特定偏移量上的位(bit)，value值可以为1或0，offset为0~2^32之间，key不存在，则创建新key bitcount key 统计字串中被设置为1的比特位数量

bitcount bits

decr key 将key中的值减1，key不存在则先初始化为0，再减1

decrby key decrement 将key中的值，减去decrement

DECRBY cc 20 //定义每次减少20（步长）

getrange key start end 返回字串值中的子字串，截取范围为start和end，负数偏移量表示从末尾开始计数，-1表示最后一个字符，-2表示倒数第二个字符

set x 123456789

getrange x -5 -1

"56789"

getrange x 0 4

"12345"

list 列表 (0代表第一个字符)

lpush key value [value…] 将一个或多个值value插入到列表key的表头，Key不存在，则创建key

lpush list a b c //list值依次为c b a

>lrange key start stop 从开始位置读取key的值到stop结束

lrange list 0 2 //从0位开始，读到2位为止

1) "c"

2) "b"

3) "a"

>lrange list 0 -1 //从开始读到结束为止

1) "c"

2) "b"

3) "a"

> lrange list 0 -2 //从开始读到倒数第2位值

1) "c"

2) "b"

lpop key 移除并返回列表头元素数据，key不存在则返回nil

> lpop list //删除表头元素，可以多次执行

"c"

llen key 返回列表key的长度

> llen list

(integer) 1

lindex key index 返回列表中第index个值

> lindex list 1

"c"

lset key index value 将key中index位置的值修改为value

lset list 3 test //将list中第3个值修改为test

rpush key value [value…] 将value插入到key的末尾

> rpush list3 a b c //list3值为a b c

(integer) 3

>lrange list 0 -1 //从开始读到结束为止

1) "a"

2) "b"

3) "c"

rpush list3 d //末尾插入d

(integer) 4

rpop key 删除并返回key末尾的值

> RPOP list3

"d"