Redis基础

标签　　空

新增属性

redis数据库读写速度快，性能高。可设置过期时间，可用于缓存。

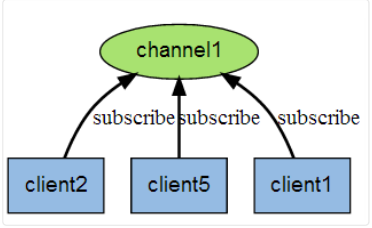
> Redis的Value支持如下几种数据类型或结构：

- **String：字符串**
- **Hash：散列表**：string 类型的 field（字段）和 value（值）的映射表，hash 适合存储对象。
- **List：列表** :简单的字符串列表，按照插入顺序排序
- **Set：集合**:String 类型的无序集合，集合成员是唯一的。
- **Sorted Set：有序集合**：有序的set

```SQL
AUTH "password"  #密码连接

PING  # 是否在运行

QUIT  # 关闭连接

SELECT index  # 切换到指定库
```

1.发布订阅模式

多个接收者订阅发送者，发送者发送消息，所有订阅的接收者都能收到。



在这里插入图片描述

> 创建并订阅频道：可同时订阅多个

```SQL
SUBSCRIBE 频道名    #订阅

PUBSUB 频道名      #查看订阅状态
```

```
4
5  PUNSUBSCRIBE 频道名   #取消订阅
```

> 发送消息到频道：

```
SQL
1  PUBLISH 频道名 "hello world"
```

2.事务

Redis事务可以一次执行多条命令，这些命令在发送 EXEC 命令前被放入队列缓存，收到EXEC 命令后进入事务执行，如果某条命令失败，其他命令照常执行。事务过程种，用户的其他命令不会插入到队列。

- 开始事务。
- 命令入队。
- 执行事务。

> 操作：可以理解为预设指令，最后统一执行

```
SQL
1  MULTI    #开始事务
2
3  SET key name   #操作1
4  SET key2 name2 #操作2
5
6  EXEC    #开始执行
```

> redis事务不是原子性的。其它命令：

```
SQL
1  DISCARD  #取消事务
2  WATCH key  #监视一个或多个key，执行前若被改动则打断事务
3  UNWATCH    #取消减少事务
```

redis 中 key 的增删改查

```
SQL
1  //设置key
2  SET keyname value
3  //删除key
4  DEL keyname
5  //修改keyname
6  RENAME keyname newkeyname
7  //查询key
8  KEYS *     //查询全部key
9  KEYS ab*   //查询ab开头的key
10 //移动key，【SELECT 库名】切换数据库
11 MOVE keyname 目标库名
```

key过期时间设置

```
SQL
1  //设置过期时间
2  EXPIRE keyname 秒
3  PEXPIRE keyname 毫秒
4  EXPIREAT keyname 秒时间戳
5  PEXPIREAT keyname 毫秒时间戳
6  //移除过期时间
7  PERSIST keyname
8  //查询过期时间
9  TTL keyname  //返回秒
10 PTTL keyname //返回毫秒
```

服务管理命令

```bash
# 设置开机自启动
redis-server --service-install redis.windows-service.conf --loglevel verbose

# 开启服务
redis-server --service-start
# 停止服务
redis-server --service-stop
# 卸载服务
redis-server --service-uninstall
```

Redis 缓存

数据库中数据被缓存在Redis中，当请求获取数据时会从redis中获取，如果redis获取不到数据则会访问数据库。这样做可以使大量请求访问到redis就返回了，减少了数据库压力。

缓存穿透

- 出现原因：请求访问的数据在redis、数据库中都不存在，但会访问数据库查询，大量访问则加大数据库压力。
- 解决办法：将请求的内容放入redis，并设置失效时间。

缓存击穿

- 出现原因：热点的键失效，大量请求突然去查询数据库。
- 解决办法：增加热点键缓存时间，或者设置为永不失效。

缓存雪崩

- 出现原因：大量键同时失效。
- 解决办法：使用哈希算法将缓存的各个键的过期时间均匀分配。

Redis 高可用

Redis通过持久化将内存的数据保存到磁盘，可以避免服务器重启导致数据丢失。但如果主机发生故障数据依旧会丢失，此时可以将数据复制多份并部署在不同的主机上。

主从复制

部署多个Redis节点，其中主节点可以进行读写，从节点只读并且数据从主节点同步（快照）。主从模式可以避免由于Redis所在主机宕机导致数据丢失。

启动主从节点：

```bash
# 启动主节点
redis-server --port 6379

# 启动从节点（此处使用命令指定，也可以在redis的conf文件加入slaveof ip port）
redis-server --port 6380 --slaveof 192.168.0.167 6379
redis-server --port 6381 --slaveof 192.168.0.167 6379
```

查看主从情况：

```sql
info replication
```

哨兵 会自动检测主从节点运行状态，如果主节点不可用，则会自动将从节点升格为新的主节点，若原本的主节点恢复，则原节点变为从节点。

设置哨兵：哨兵配置文件 redis-sentinel sentinel.conf

```bash
# 监控主节点，会自动监控所有从节点
sentinel monitor 主节点名称 主节点IP 主节点端口 1
```

redis 集群

主从模式中，每个节点都会存储所有的数据，比较浪费内存。使用分布式集群可以将数据分散到不同的节点上，即分布式存储。

集群至少需要3个主节点，因为投票容错机制要求超过半数节点认为某个节点挂了该节点才是挂了，并且每个节点都需要有从节点。

用集群工具（redis/src 目录下的 redis-trib.rb 文件）进行集群，该工具用 ruby 编写，所以需要先安装 ruby：

```Bash
yum install ruby
yum install rubygems
gem install redis
```

构建集群：首先启动所有节点的 redis 服务。

```Bash
./redis-trib.rb create --replicas 1
127.0.0.1:6380
127.0.0.1:6381
127.0.0.1:6382
127.0.0.1:6383
127.0.0.1:6384
127.0.0.1:6385
```

查看集群：

```Bash
# 查看集群信息
cluster info

# 查看集群里有多少个节点
cluster nodes
```

Redis 分布式锁

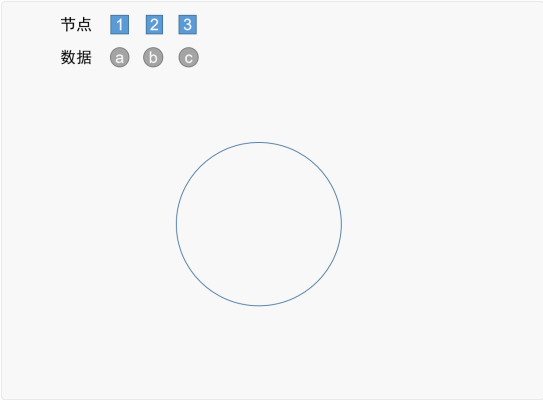
Redis 分布式算法

传统分布式算法

对数据（key）进行哈希运算，然后均匀分配到各个节点中。

一致性哈希算法

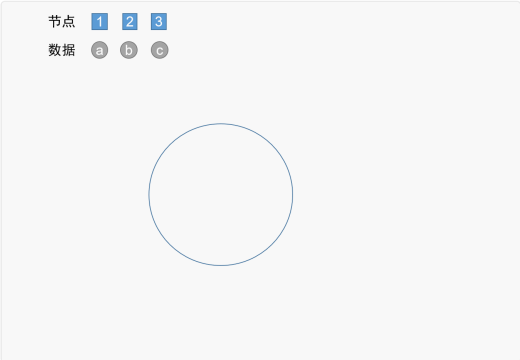
在一个环形哈希空间（取值 0~2^32-1），对节点（IP、port 等）进行哈希运算，均匀地分布在环上。然后对数据（key）进行哈希运算并分布在环上，数据将会被存入顺时针方向距离自身最近的节点中。



优势：当 redis 节点增加、删除时，仅对最近数据产生影响。其它数据依然有效，可以防止雪崩和穿透。

hash 倾斜性

当节点数量很少时，使用一致性 hash 算法会出现节点分配不均的情况（假设 3 个节点，刚好两个节点在一起，那么数据在运算后存放的就不均匀了，环中距离较大的节点压力会比较大。）



对于hash倾斜性，一致性哈希算法引入虚节点的概念，大量虚节点均匀分布在环上，数据hash运算后与虚节点建立映射关系，虚节点通过一系列哈希运算最终指向真实节点。