# Redis高级-集群

## Redis集群工作原理

### 数据存储原理

#### 知识回顾

分片数据存储原理: 根据hash一致性算法实现数据存储.

Hash一致性运算发生在服务器端.

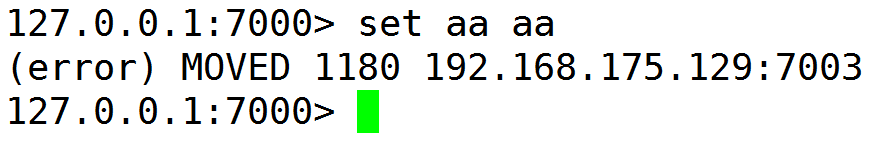
在存储数据前已经计算完成存储到哪台redis服务器中.

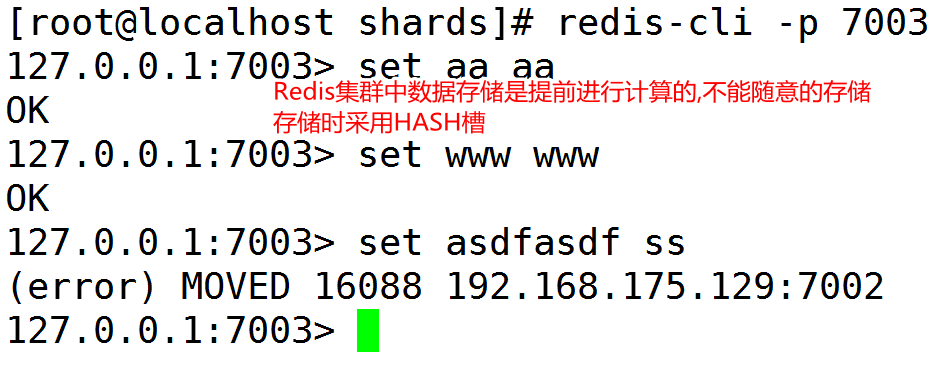
分片中存在的问题:

分片的数据是用户通过服务器程序计算进行存储.用户也可以利用redis的客户端进行非法的操作.修改分片中的数据!!!

#### 集群中数据存储

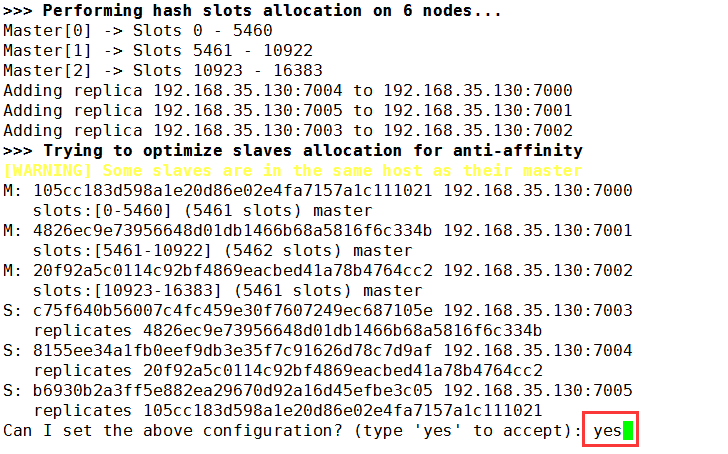
1.当在从节点中存储数据时.不允许操作,只有主节点能够set/get数据

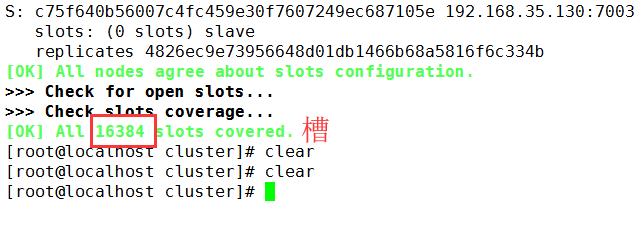




2.Redis集群与分片明显不同.存储数据必须满足条件.

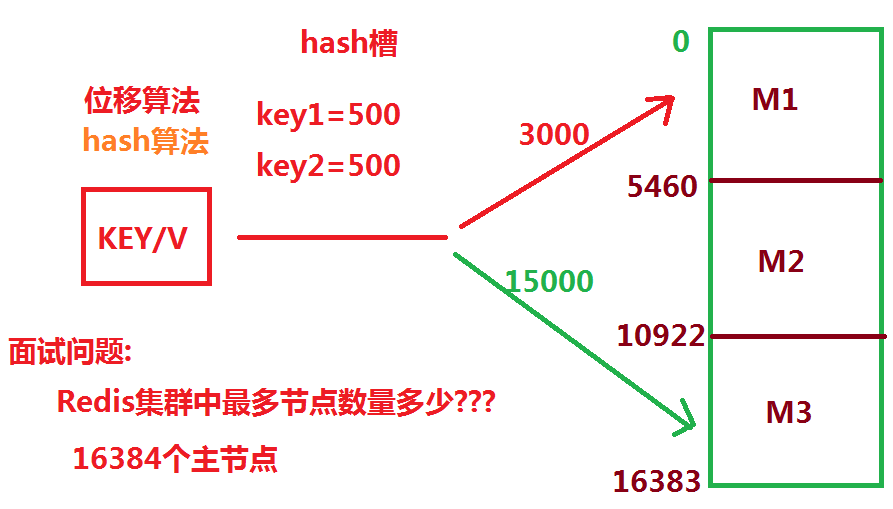
#### Redis集群储存条件





说明:redis集群一共有**16384**个槽位.这些操作会**均匀**的分配给**主节点**.

当数据进行存取时首先根据key进行计算.计算槽位的位置.之后从节点中获取数据.



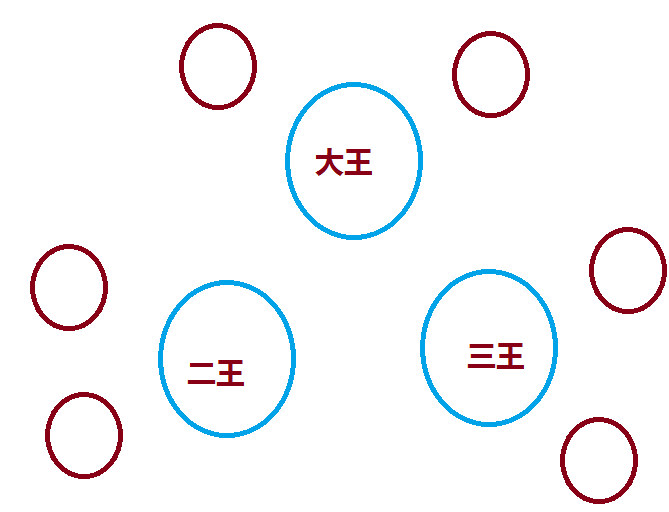
Crc16算法

### Redis集群宕机条件

宕机条件:Redis集群宕机的条件是**主节点宕机**则集群奔溃

例子:9个redis节点 3主6从 节点宕机多少次集群奔溃

5-7台集群奔溃



条件说明:Redis集群中当节点宕机后,由主节点进行推选.如果当前节点没有从节点时.从其他主节点中借用**多余的从节点**.

## SpringBoot整合Redis集群

### 入门案例

@Test

**public** **void** test01() {

Set<HostAndPort> sets = **new** HashSet<>();

sets.add(**new** HostAndPort("192.168.175.129",7000));

sets.add(**new** HostAndPort("192.168.175.129",7001));

sets.add(**new** HostAndPort("192.168.175.129",7002));

sets.add(**new** HostAndPort("192.168.175.129",7003));

sets.add(**new** HostAndPort("192.168.175.129",7004));

sets.add(**new** HostAndPort("192.168.175.129",7005));

JedisCluster cluster = **new** JedisCluster(sets);

cluster.set("1902", "集群搭建完成");

System.***out***.println("获取集群数据:"+cluster.get("1902"));

}

### 编辑pro文件

redis.nodes=192.168.175.129:7000,192.168.175.129:7001,192.168.175.129:7002,192.168.175.129:7003,192.168.175.129:7004,192.168.175.129:7005

### 编辑配置类

//表示redis配置类

@Configuration //xml

@PropertySource("classpath:/properties/redis.properties")

**public** **class** RedisConfig {

//redis.nodes=192.168.175.129:7000,192.168.175.129:7001,192.168.175.129:7002,192.168.175.129:7003,192.168.175.129:7004,192.168.175.129:7005

@Value("${redis.nodes}")

**private** String redisNodes;

@Bean

**public** JedisCluster jedisCluster() {

Set<HostAndPort> nodes = **new** HashSet<>();

//1.根据,号拆分为多个node

String[] strNode = redisNodes.split(",");

//IP:端口

**for** (String node : strNode) {

String host = node.split(":")[0];

**int** port = Integer.*parseInt*(node.split(":")[1]);

HostAndPort hostAndPort =

**new** HostAndPort(host, port);

nodes.add(hostAndPort);

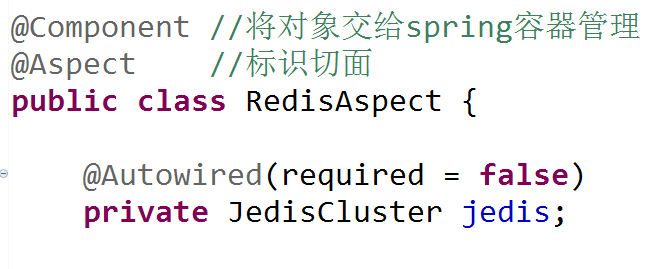
}

**return** **new** JedisCluster(nodes);

}

}

### 修改AOP切面



## 京淘后台复习

### SpringBoot

#### 启动步骤

1. 编辑pom.xml文件

包含了框架引用的全部jar包版本信息,以及依赖信息

其中jar包中标识了配置文件信息

1. 编辑主启动类

自动配置了包扫描/自动实现了jar包的自动配置

1. 编辑YML配置
2. 可视化的形式配置(自动提示)
3. 配置文件层级结构明显.
4. 功能强大 多环境测试

### Mybatis-plus单表CURD

1. 表与对象一一映射
2. 表字段与对象属性一一映射
3. 封装API接口简化用户操作 CRUD
4. 将接口中的方法最终转化sql语句

### 分布式规则

1. 垂直拆分

按照功能模块拆分

1. 水平拆分

按照业务层级拆分

问题1:jar包如何管理 自定义jt-parent项目 管理jar包 继承

问题2:工具类如何管理 自定义jt-common管理java类型 依赖

聚合项目(POM)构建

### RESTFul

功能:实现页面通用跳转/简化get请求方式

1. url:/参数使用/分割
2. 服务器:

@RequestMapping("/page/{moduleName}")

**public** String itemAdd(@PathVariable String moduleName) {

**return** moduleName;

}

### 业务实现:

1. 通用页面跳转
2. 商品列表展现 EasyUI 表格数据回显要求 分页操作
3. 商品分类信息 EasyUI Tree结构

数据查询方式 id=”” parentId=

1. 文件上传
2. Nginx

反向代理/负载均衡

1. Windows项目部署
2. 学习Redis:

利用AOP实现Redis集群缓存处理