Redis客户端使用指导

# 背景

Redis是一个完全开源免费的key-value内存数据库 。它具有速度快（所有数据都在内存中完成，读写速度分别达到10万/20万），易扩展（使用客户端Sharding技术，横向扩展）等特点。

DSDP采用是redis官方推荐的Jedis客户端API。客户端通过一致性Hash算法进行分片计算的方式，进行横向扩展。

针对不同的组网模式，能力中心提供了不同的连接池。并在此基础上对不同的连接池进行融合，提供了一套适应不同组网的统一API。

下面分别介绍连接池的用法

# Redis shard组网

独立安装的多个redis节点，在客户端使用一致性Hash算法选取某一个redis进行读写操作。此组网方式简单，扩展性好，假设有n个redis，当1个redis挂掉，整个集群1/n失效。

我们在ShardedJedisPool的基础上，提供了一个带黑名单的ShardedJedisPool。在客户端维护了一个黑名单列表，按照一定的配置策略，定期检测redis的连通性情况，以加入和移出黑名单。当客户端每次读写redis之前，先比较shard的结果是不是在黑名单里面，如果在黑名单，直接抛异常，用来达到快速失败的目的。

## 逻辑组网



## API介绍

### 连接池初始化

|  |
| --- |
| JedisPoolConfig jedisPoolConfig = **new** JedisPoolConfig();  jedisPoolConfig.setConnectionTimeout(1000);  jedisPoolConfig.setMaxIdle(10);  jedisPoolConfig.setMaxTotal(30);  jedisPoolConfig.setMaxWaitMillis(1000);  jedisPoolConfig.setMutiKeyTimeOut(200);  jedisPoolConfig.setSingleKeyTimeOut(100);    String host = "127.0.0.1";  List<JedisShardInfo> shards = **new** ArrayList<JedisShardInfo>();  JedisShardInfo jedisShardInfo = **new** JedisShardInfo(host);  shards.add(jedisShardInfo);  ShardedJedisPoolWithBlacklist shardedJedisPool = **new** ShardedJedisPoolWithBlacklist(jedisPoolConfig, shards);    shardedJedisPool.setCheckIntervalMills(1000);  shardedJedisPool.setMessageTimeoutMillis(1000);  shardedJedisPool.setMessageTimeoutTimes(5); |

### 连接池销毁

|  |
| --- |
| ShardedJedisPoolWithBlacklist shardedJedisPool；  shardedJedisPool.destroy(); |

### 获取ShardedJedis

|  |
| --- |
| @Override  **public** ShardedJedis getShardedJedis()  {  ShardedJedis sjedis = shardedJedisPool.getResource();  **return** sjedis;  } |

### 归还ShardedJedis

|  |
| --- |
| @Override  **public** **void** returnShardedJedis(ShardedJedis shardedJedis)  {  **try**  {  shardedJedisPool.returnResource((ShardedJedisWithBlacklist)shardedJedis);  }  **catch** (Throwable e)  {  shardedJedisPool.returnBrokenResource((ShardedJedisWithBlacklist)shardedJedis);  }  } |

# Redis Sentinel+主备

该方案是根据redis官方文档推荐的高可用方案进行设计开发。所有的redis的节点都有一个备节点做故障转移。Sentinel是redis自带的一个哨兵监控进程。他定期检测监控的redis主节点的状态，一旦发现主节点挂掉，就会把备节点提升为主节点，同时发布一个主备切换事件，通知redis访问模块，新的主节点是谁。

## 逻辑组网



## API介绍

### 连接池初始化

|  |
| --- |
| ShardedJedisSentinelPool shardedJedisSentinelPool = **null**;    /\*\*  \* jedis连接池配置  \*/  JedisPoolConfig poolConfig = **null**;    @Override  **public** **void** initCluster(String redisUrl, **final** GenericObjectPoolConfig poolConfig)  {  shardedJedisSentinelPool = **new** ShardedJedisSentinelPool(redisUrl, (JedisPoolConfig)poolConfig);  **this**.poolConfig = (JedisPoolConfig)poolConfig;  } |

### 连接池销毁

|  |
| --- |
| ShardedJedisSentinelPool shardedJedisSentinelPool = **null**;  shardedJedisSentinelPool.destroy(); |

### 获取ShardedJedis

|  |
| --- |
| @Override  **public** ShardedJedis getShardedJedis()  {  ShardedJedis sjedis = shardedJedisSentinelPool.getResource();  **return** sjedis;  } |

### 归还ShardedJedis

|  |
| --- |
| @Override  **public** **void** returnShardedJedis(ShardedJedis shardedJedis)  {  **try**  {  shardedJedisSentinelPool.returnResource(shardedJedis);  }  **catch** (Throwable e)  {  shardedJedisSentinelPool.returnBrokenResource(shardedJedis);  }  } |

# 统一API

## 物理组网

根据url列表判断自动判断组网模式是上文说的shard组网还是sentinel组网。如果url列表全是sentinel节点，那么就认为是sentinel组网，如果url列表全是redis节点，那么就认为是shard组网。

## API介绍

### 连接池初始化

|  |
| --- |
| /\*\*  \* redis或者sentinel地址，多个地址用|分割，如10.47.12.6:6381|10.47.12.12:6382  \*/  **private** String connectUrls;  **private** JedisPoolConfig jedisPoolConfig;  **private** JedisClientCluster client;  client = **new** CommonClusterImpl();  client.initCluster(connectUrls, jedisPoolConfig); |

### 连接池销毁

|  |
| --- |
| client.destroyCluster |

### 获取ShardedJedis

|  |
| --- |
| ShardedJedis jedis = **null**;  **try**  {  jedis = client.getShardedJedis();  Jedis jedisClient = jedis.getShard(key);  ……  }  **finally**  {  returnJedis(jedis);  } |

### 归还ShardedJedis

|  |
| --- |
| client.returnShardedJedis(jedis); |