Jedis分片Sentinel连接池实验

原创

cdai 于 2015-08-29 08:15:42 发布

分类专栏： Redis 文章标签： redis jedis sharding

本文链接：https://blog.csdn.net/dc_726/article/details/48084373

Jedis分片Sentinel连接池实验

1.起因

众所周知，Redis官方HA工具Sentinel已经问世很久了，但令人费解的是，Jedis官方却迟迟没有更新它的连接池。到目前Maven库中最新的2.7.3版本为止，都只能要么使用分片连接池，要么使用不分片的Sentinel连接池。如果既进行了Sharding，又对每组的主从实例配置Sentinel进行监控，怎么办？答案是只能自己开发了，晕！还好万能的GitHub上已经有人提供了一个简单可用的分片Sentinel连接池实现，于是就直接拿来用用。

2.Sentinel配置

我的Redis是这样配置的：一个主6379带一个从16379，名字叫mymaster-6379；另一个主6380带另一个从16380，名字叫mymaster-6380。Sentinel按照惯例，起三个实例26379,26380,26381。以其中一个为例，其他的Sentinel配置只是端口号不同：

```
port 26379
dir /tmp

sentinel monitor mymaster-6379 192.168.241.220 6379 2
sentinel down-after-milliseconds mymaster-6379 30000
sentinel parallel-syncs mymaster-6379 1
sentinel failover-timeout mymaster-6379 180000

sentinel monitor mymaster-6380 192.168.241.220 6380 2
sentinel down-after-milliseconds mymaster-6380 30000
sentinel parallel-syncs mymaster-6380 1
sentinel failover-timeout mymaster-6380 180000
```

如果想要Sentinel快速failover，就把sentinel down-after-milliseconds mymaster-6379 30000中的30秒改短点，这个时间是Sentinel Ping不到Master多久后开始failover。具体Redis主从复制和Sentinel如何配置就不赘述了，请参考我以前写的Redis主从和HA配置。

3.Jedis客户端扩展

3.1 测试代码

将GitHub那个开源的连接池引到工程里，使用起来很简单，只需通过构造方法传入所有Sentinel实例的连接地址以及master的名字列表。连接池配置可以使用默认的。

注意：一定要每次重新从pool中取一个连接，否则会一直访问老master。

```
        public static void main(String[] args) throws Exception {
            Set<String> sentinels = new HashSet<String>();
            sentinels.add("192.168.111.222:26379");
            sentinels.add("192.168.111.222:26380");
            sentinels.add("192.168.111.222:26381");

            ShardedJedisSentinelPool pool = new ShardedJedisSentinelPool(
                    Arrays.asList("mymaster-6379", "mymaster-6380"),
                    sentinels
            );

            while (true) {
                String ts = new SimpleDateFormat("hh:mm:ss").format(new Date());
                ShardedJedis jedis = pool.getResource();
                try {
                    System.out.println(ts + ": hello=" + jedis.get("hello"));
                } catch (Exception e) {
                    System.out.println(ts + ": Cannot connect to Redis");
                } finally {
                    jedis.close();
                }
                Thread.sleep(500L);
            }
        }
```

3.2 连接泄露Bug

循环时发现ShardedJedisSentinelPool连接池有时竟然满了！毕竟默认是8个连接，但为什么会这样？查了下代码，果然是这个原因！这个新扩展的pool没有为getResource()出来的ShardedJedis设置DataSource，所以关闭ShardedJedis时没法正常还回到连接池中。

补丁很简单，就是参照Jedis的ShardedJedisPool，覆写一些资源管理的方法。

```
        @Override
        public ShardedJedis getResource() {
            ShardedJedis jedis = super.getResource();
            jedis.setDataSource(this);
            return jedis;
        }

        @Override
        public void returnBrokenResource(final ShardedJedis resource) {
            if (resource != null) {
                returnBrokenResourceObject(resource);
            }
        }

        @Override
        public void returnResource(final ShardedJedis resource) {
            if (resource != null) {
                resource.resetState();
                returnResourceObject(resource);
            }
        }
```

3.3 KeyTag

此外，ShardedJedisSentinelPool的构造方法中没提供传入KeyTagPattern的地方，这也需要手动传入到ShardedJedisFactory中。需要的话，自己动手加一下吧！

4.Failover实验

以key=hello为例，事先在6380实例上准备好key的值。

4.1 实验结果

首先，启动Java程序不断访问。然后，在后台kill掉6380那个Redis实例。控制台输出会警告无法连接Redis。大概30秒左右，本地连接池注册的Sentinel监听器会收到failover消息。于是，从pool.getResource()拿到的连接自动切换到Slave实例了，又可以查询到hello对应的值了。

```
10:13:42: hello=nihaonihao111
10:13:42: Cannot connect to Redis
10:13:43: Cannot connect to Redis
10:13:45: Cannot connect to Redis
10:13:48: Cannot connect to Redis
10:13:50: Cannot connect to Redis
10:13:53: Cannot connect to Redis
10:13:56: Cannot connect to Redis
10:13:58: Cannot connect to Redis
10:14:01: Cannot connect to Redis
10:14:03: Cannot connect to Redis
10:14:06: Cannot connect to Redis
10:14:08: Cannot connect to Redis
10:14:11: Cannot connect to Redis
八月 24, 2015 10:14:13 上午 redis.clients.jedis.ShardedJedisSentinelPool initPool
信息: Created ShardedJedisPool to master at [192.168.111.222:6379 192.168.111.222:16380 ]
10:14:13: hello=nihaonihao111
10:14:14: hello=nihaonihao111
10:14:14: hello=nihaonihao111
```

4.2 如何恢复

如果想反复试验怎么办？很简单，重新启动6380实例，但此时16380已经变成了Master，6380成了它的Slave。没关系！随便连接到一个Sentinel实例上，执行sentinel failover mymaster-6380就可以强制failover一次。这样6380和16380的主从关系就又恢复了！

5.总结

总体来说这个连接池还是不错的，虽然有点小问题，但免去了开发的麻烦！不知道用于生产环境的话，是否还有其他“坑”，不管怎样可以先用着，感谢作者的分享！