Redis 开发规范

参考：https://yq.aliyun.com/articles/531067

一、键值设计

1. key 名设计

- (1)【建议】：可读性和可管理性

  以业务名(或数据库名)为前缀(防止 key 冲突)，用冒号分隔，比如业务名:表名:id

```
ugc:video:1
```

- (2)【建议】：简洁性

  保证语义的前提下，控制 key 的长度，当 key 较多时，内存占用也不容忽视，

  例如：

```
user:{uid}:friends:messages:{mid}简化为 u:{uid}:fr:m:{mid}。
```

- (3)【强制】：不要包含特殊字符

  反例：包含空格、换行、单双引号以及其他转义字符

  详细解析

2. value 设计

- (1)【强制】：拒绝 bigkey(防止网卡流量、慢查询)

  string 类型控制在 10KB 以内，hash、list、set、zset 元素个数不要超过 5000。

  反例：一个包含 200 万个元素的 list。

  非字符串的 bigkey，不要使用 del 删除，使用 hscan、sscan、zscan 方式渐进式删除，同时要注意防止 bigkey 过期时间自动删除问题(例如一个 200 万的 zset 设置 1 小时过期，会触发 del 操作，造成阻塞，而且该操作不会不出现在慢查询中(latency 可查))，查找方法和删除方法

  详细解析

- (2)【推荐】：选择适合的数据类型。

  例如：实体类型(要合理控制和使用数据结构内存编码优化配置,例如 ziplist，但也要注意节省内存和性能之间的平衡)

  反例：

```
set user:1:name tom
set user:1:age 19
set user:1:favor football
```

  正例:

```
hmset user:1 name tom age 19 favor football
```

3.【推荐】：控制 key 的生命周期，redis 不是垃圾桶。

建议使用 expire 设置过期时间(条件允许可以打散过期时间，防止集中过期)，不过期的数据重点关注 idletime。

二、命令使用

1.【推荐】 O(N)命令关注 N 的数量

例如 hgetall、lrange、smembers、zrange、sinter 等并非不能使用，但是需要明确 N 的值。有遍历的需求可以使用 hscan、sscan、zscan 代替。

2.【推荐】：禁用命令

禁止线上使用 keys、flushall、flushdb 等，通过 redis 的 rename 机制禁掉命令，或者使用 scan 的方式渐进式处理。

3.【推荐】合理使用 select

redis 的多数据库较弱，使用数字进行区分，很多客户端支持较差，同时多业务用多数据库实际还是单线程处理，会有干扰。

4.【推荐】使用批量操作提高效率

```
原生命令：例如 mget、mset。
非原生命令：可以使用 pipeline 提高效率。
```

但要注意控制一次批量操作的**元素个数**(例如 500 以内，实际也和元素字节数有关)。

注意两者不同：

1. 原生是原子操作，pipeline 是非原子操作。
2. pipeline 可以打包不同的命令，原生做不到
3. pipeline 需要客户端和服务端同时支持。

5.【建议】Redis 事务功能较弱，不建议过多使用

Redis 的事务功能较弱(不支持回滚)，而且集群版本(自研和官方)要求一次事务操作的 key 必须在一个 slot 上(可以使用 hashtag 功能解决)

6.【建议】Redis 集群版本在使用 Lua 上有特殊要求：

- 1.所有 key 都应该由 KEYS 数组来传递，redis.call/pcall 里面调用的 redis 命令，key 的位置，必须是 KEYS array，否则直接返回 error，"-ERR bad lua script for redis cluster, all the keys that the script uses should be passed using the KEYS array"
- 2.所有 key，必须在 1 个 slot 上，否则直接返回 error, "-ERR eval/evalsha command keys must in same slot"

7.【建议】必要情况下使用 monitor 命令时，要注意不要长时间使用。

三、客户端使用

1.【推荐】

避免多个应用使用一个 Redis 实例

正例：不相干的业务拆分，公共数据做服务化。

2.【推荐】

使用带有连接池的数据库，可以有效控制连接，同时提高效率，标准使用方式：

```
执行命令如下：
Jedis jedis = null;
try {
    jedis = jedisPool.getResource();
    //具体的命令
    jedis.executeCommand()
} catch (Exception e) {
    logger.error("op key {} error: " + e.getMessage(), key, e);
} finally {
    //注意这里不是关闭连接，在 JedisPool 模式下，Jedis 会被归还给资源池。
    if (jedis != null)
        jedis.close();
}
```

下面是 JedisPool 优化方法的文章:

- Jedis 常见异常汇总

- JedisPool 资源池优化

3.【建议】

高并发下建议客户端添加熔断功能(例如 netflix hystrix)

4.【推荐】

设置合理的密码，如有必要可以使用 SSL 加密访问（阿里云 Redis 支持）

5.【建议】

根据自身业务类型，选好 maxmemory-policy(最大内存淘汰策略)，设置好过期时间。

默认策略是 volatile-lru，即超过最大内存后，在过期键中使用 lru 算法进行 key 的剔除，保证不过期数据不被删除，但是可能会出现 OOM 问题。

其他策略如下：

- allkeys-lru：根据 LRU 算法删除键，不管数据有没有设置超时属性，直到腾出足够空间为止。
- allkeys-random：随机删除所有键，直到腾出足够空间为止。
- volatile-random:随机删除过期键，直到腾出足够空间为止。
- volatile-ttl：根据键值对象的 ttl 属性，删除最近将要过期数据。如果没有，回退到 noeviction 策略。

- noeviction：不会剔除任何数据，拒绝所有写入操作并返回客户端错误信息"(error) OOM command not allowed when used memory"，此时 Redis 只响应读操作。

四、相关工具

1.【推荐】：数据同步

redis 间数据同步可以使用：redis-port

2.【推荐】：big key 搜索

redis 大 key 搜索工具

3.【推荐】：热点 key 寻找(内部实现使用 monitor，所以建议短时间使用)

facebook 的 redis-faina

> 阿里云 Redis 已经在内核层面解决热点 key 问题，欢迎使用。

五 附录：删除 bigkey

> 1. 下面操作可以使用 pipeline 加速。
> 2. redis 4.0 已经支持 key 的异步删除，欢迎使用。

1. Hash 删除：hscan + hdel

```
public void delBigHash(String host, int port, String password, String
bigHashKey) {
    Jedis jedis = new Jedis(host, port);
    if (password != null && !"".equals(password)) {
        jedis.auth(password);
    }
    ScanParams scanParams = new ScanParams().count(100);
    String cursor = "0";
    do {
        ScanResult<Entry<String, String>> scanResult =
jedis.hscan(bigHashKey, cursor, scanParams);
        List<Entry<String, String>> entryList = scanResult.getResult();
        if (entryList != null && !entryList.isEmpty()) {
            for (Entry<String, String> entry : entryList) {
                jedis.hdel(bigHashKey, entry.getKey());
            }
        }
        cursor = scanResult.getStringCursor();
    } while (!"0".equals(cursor));

    //删除 bigkey
    jedis.del(bigHashKey);
}
```

2. List 删除：ltrim

```
public void delBigList(String host, int port, String password, String
bigListKey) {
```

```
    Jedis jedis = new Jedis(host, port);
    if (password != null && !"".equals(password)) {
        jedis.auth(password);
    }
    long llen = jedis.llen(bigListKey);
    int counter = 0;
    int left = 100;
    while (counter < llen) {
        //每次从左侧截掉 100 个
        jedis.ltrim(bigListKey, left, llen);
        counter += left;
    }
    //最终删除 key
    jedis.del(bigListKey);
}
```

3. Set 删除：sscan + srem

```
public void delBigSet(String host, int port, String password, String
bigSetKey) {
    Jedis jedis = new Jedis(host, port);
    if (password != null && !"".equals(password)) {
        jedis.auth(password);
    }
    ScanParams scanParams = new ScanParams().count(100);
    String cursor = "0";
    do {
        ScanResult<String> scanResult = jedis.sscan(bigSetKey, cursor,
scanParams);
```

```
        List<String> memberList = scanResult.getResult();
        if (memberList != null && !memberList.isEmpty()) {
            for (String member : memberList) {
                jedis.srem(bigSetKey, member);
            }
        }
        cursor = scanResult.getStringCursor();
    } while (!"0".equals(cursor));

    //删除 bigkey
    jedis.del(bigSetKey);
}
```

4. SortedSet 删除：zscan + zrem

```
public void delBigZset(String host, int port, String password, String
bigZsetKey) {
    Jedis jedis = new Jedis(host, port);
    if (password != null && !"".equals(password)) {
        jedis.auth(password);
    }
    ScanParams scanParams = new ScanParams().count(100);
    String cursor = "0";
    do {
        ScanResult<Tuple> scanResult = jedis.zscan(bigZsetKey, cursor,
scanParams);
        List<Tuple> tupleList = scanResult.getResult();
        if (tupleList != null && !tupleList.isEmpty()) {
            for (Tuple tuple : tupleList) {
```

```
                jedis.zrem(bigZsetKey, tuple.getElement());
            }
        }
        cursor = scanResult.getStringCursor();
    } while (!"0".equals(cursor));

    //删除 bigkey
    jedis.del(bigZsetKey);
}
```