**（线上问题系列三）记录一次线上Redis性能优化**

记录一次线上Redis优化：

一、 现象描述

线上环境Redis采用哨兵模式，版本为3.0.7，服务器采用4C8G，TPS在20000左右，当设备（NB）在线用户数超过12万后，主节点单核CPU从30%左右直线上升至将近60%， CPU负荷高居不下且不稳定间接影响业务量的提升，亟待优化。

由于redis单线程的设计特性，任何一条阻塞的命令都会引起redis整个实例的阻塞，所以在使用过程中，我们首先从Linux环境情况、命令分布，key情况，客户端情况等方面检查。

二、 检查指标

1. 1. 内存使用情况

redis使用内存为800M，linux内存使用率85%，未使用SWAP。（因为有其他java进程共用服务器），属于正常情况。

1. 2. 单核CPU使用情况



1. 3. 整体CPU使用情况

整体25%左右，正常范围。

1. 4. IO，磁盘IO和网络IO使用情况

我们发现磁盘IO比较低，而网络出口流量达到15M，说明读数据量比较大。

1. 5. 网络延迟情况ICMP response time

检查网络响应时间在5ms以内，属于正常范围。

1. 6. 命令执行分布和时间，info commandstats命令。

检查发现，在短时间内有大量的hgetall调用，hgetall方法是将hash table里面的所有内容一次性全部读出来，如果哈希值的内容比较大，极容易导致堵塞。是一个优化点。

1. 7. TPS，检查TPS的方法可以统计total\_commands\_processed增长量

当前TPS不到2.5万，未达到瓶颈点。

1. 8. 检查慢查询slowlog get 128，这个可以定位优化。

可以看到慢查询命令是主要是hmset，hgetall和scan等，不过整体慢查询时间能接受。

1. 9. 检查客户端列表，检查连接时间和内存消耗omem的客户端。

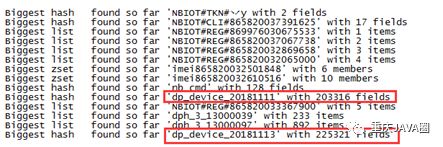
redis-cli -p 6379 -h xxxx –a xxxx client list |grep -v “omem=0”

发现有较多长期连接的客户端，总的客户端数量650。

1. 10. bigkey检查，优化比较大的key

redis-cli -p 6379 -h xxxx –a xxxx –-bigkeys

发现较多bigkey，后期需要进行优化。



三、 检查服务器内核参数（Redis官方建议设置，服务器默认未设置，需要手工设置）

1. 1. 内存分配

Redis是内存操作，需要优先使用内存。设置overcommit 为1。

1. 2. THP禁用

THP开启后，最大内存页为2M，因为fork 调用的copy-on-write机制是基于操作系统页这个单位的，也就是只有有写入的脏页会被复制，但是一般系统不会在短时间内所有的页都发生了写入而导致复制，所以每次写命令引起的内存复制从4K变成了2M，从而拖慢写操作的执行时间，导致大量写操作慢查询。所以确保禁用Linux内核特性transparent huge pages，通过命令echo never >  
sys/kernel/mm/transparent\_hugepage/enabled来设置。

1. 3. swap使用概率

防止内存溢出OOM，故设置允许使用swap。（Redis本身不建议使用SWAP，OOM的情况万不得已）。echo {bestvalue} > /proc/sys/vm/swappiness

1. 4. ulimit设置

需要设置比最大客户端连接数要大至少32。因为此服务器还有java应用程序，所以需要设置得更大。

1. 5. TCPbacklog 设置

设置TCP等待连接队列长度为最大值。

echo 511 > /proc/sys/net/core/somaxconn

四、 检查配置文件

1. 1. 内存设置

若是启用了Redis快照功能，应该设置“maxmemory”值为系统可使用内存的45%，设置一个明确的maxmemory参数来限制你的实例，以便确保实例会报告错误而不是当接近系统内存限制时失败。此Redis服务器内存空间较多，暂未设置。

1. 2. 绑定本机网卡

检查日志会发现陆陆续续有较多错误：Error condition on socketfor SYNC: Connection refused。发现是连接问题，需要在主节点配置文件绑定本机IP，增加bind x.x.x.x

1. 3. 检查数据持久化策略

数据落磁盘尽可能减少性能损坏，以空间换时间。设置如下命令：

rdbcompression no

rdbchecksum no

1. 4. 客户端timeout

设置一个超时时间，防止无用的连接占用资源。设置如下命令：

timeout 150

tcp-keepalive 150

1. 5. 优化AOF和RDB，减少占用CPU时间

主库可以不进行dump操作或者降低dump频率。

取消AOF持久化。命令如下:

appendonly no

五、 优化后情况

经过优化后观察，业务量已经增加了30%，TPS达到了3.5万，网络流量已经增加了30%，从15M增加到了20M多，而单核cpu平均值反而降到了50%以下。所有CPU平均值在25%左右。Redis用户连接数减少了60%，当前连接在260左右。观察日志，未发现有错误信息。



六、 后续考虑改进方案

1. 考虑读写分离。

2. 减少时间复杂度比较高的操作，比如hgetall。

3. bigkey优化。

4. 其他无相关业务迁移。