# Redis做内容缓存改善大集群性能问题

mnstory.net

启动虚拟机前，需要选择运行节点，运行节点的选择主要是评估每个节点的内存和CPU使用，之前的PERL端是通过datareport向所有节点发送请求，走NFS通道，收集结果后打包回复PERL，大集群是48个节点，最开始尝试批量启动1000台虚拟机一天都没能启动完成。

评估每个节点的内存和CPU很必要，但是过中消耗时间太久，平均每台机器启动消耗2分钟，1000台需要33小时。

小打小闹优化效果并不明显，对比了几种优化手段，最后选择了使用Redis内存数据做缓存。

## 为什么选择Redis？

首先，不用引入新组件，因为我们的系统一直有它，但被忽视了很久。

其次，Redis的代码写的不错，短小精悍，运行稳定，效率非常高。

我测试从48个节点并发1600个链接不停读写主控的数据库，消耗资源为4.7% CPU, RSS内存消耗22MB，读写时间在毫秒级别：

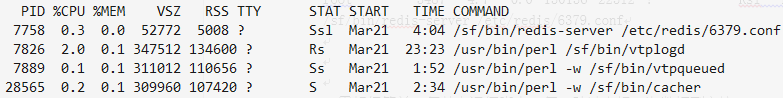
[3487] 03 Mar 20:32:33.162 - DB 0: 16 keys (0 volatile) in 32 slots HT.

[3487] 03 Mar 20:32:33.162 - 1602 clients connected (0 slaves), 75839232 bytes in use

# ps auxf | grep redis

root 3487 4.7 0.0 130136 22512 ? Rsl Mar02 78:43 /sf/bin/redis-server /etc/redis/6379.conf

在实际使用的时候呢？资源消耗还不如一个简单的PERL程序：



最主要的是，运行这么多天，从来没有遇到redis-server出问题！

## 缓存程序

缓存的思想很简单，一般是用空间换时间，而我们这个缓存，是将集中的时间消耗分散，即原来需要统计内存消耗的时候才去各节点获取数据，现在提前准备好，这算不得一种高明策略，或者说是一种浪费资源的策略。

但是，这种策略适用于大集群，因为，消耗资源少，而获得的价值高。

缓存原理：每节点运行缓存脚本，每4秒钟上报一次数据到主控Redis，主控在需要的时候从本机的Redis里面取节点内存和CPU信息。

4秒是怎么设计的？有没有触发机制或者关闭机制以节省资源？

啊！没有，代码很简单，小学毕业的写法：

sub main {

init();

for (my $idx = 0; $idx != -1 ; ++$idx) {

my $startTime = [gettimeofday];

if (0 == ($idx % 4)) {

callWapper('onNodeMemoryInstant');

} elsif (0 == ($idx % 7)) {

callWapper('onNodeInterfacesStatus');

} elsif (0 == ($idx % 19)) {

callWapper('onNodeMemoryStatic');

callWapper('onNodeHWCfgInfo');

}

callWapper('onNodeInstantLoadInfo');

my $costTime = int(1000\*tv\_interval($startTime));

if ($costTime > 500) {

ldebug("call $costTime \_\_ALL\_\_ sleep ". (1000 - $costTime));

}

if ((1000-$costTime)>0 && (1000-$costTime)<1000) {

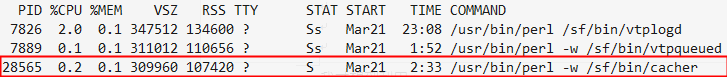
usleep((1000 - $costTime)\*1000);

}

}

}

It’s work! 并运行得挺好，只是我们VMP里的PERL库先天繁重导致一启动就会消耗不少内存，好在不耗CPU：



最开始我只是为了缓存节点的内存信息，但是，你也看到了，大家开始往里面加代码……变成了上面的很多回调……

引入这种设计，还需考虑：

1. 集群分裂和主控不可访问等异常。

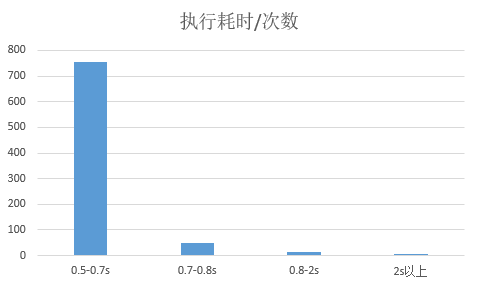
这种情况，在分裂的时候，重新获取主控。如果主控掉线，选择一个节点替代redis server.

1. cacher缓存更新时间间隙大于两次获取时间，导致两次获取结果一直。

这种情况，理论上有问题，实际情况没啥大碍，因为虚拟机启动最快1s左右，且消耗掉内存的过程不会快到秒级别的突变，多台主机更新时会有错落，整体上来说4s间隙内任何时间点获取的数据一定有些差异。大集群测试觉察不出影响，集群分布均匀。

1. 某些回调耗时太长导致影响其他更新

这个没有处理，整体耗时多在500毫秒一下（500ms以上占824/228848=0.36%），偶尔aSAN出问题会导致vs脚本卡主会导致耗时不正常，待处理。统计了当天超过500毫秒的cacher耗时情况（最极端的有一次vs异常导致的25235毫秒卡顿）分布如下：

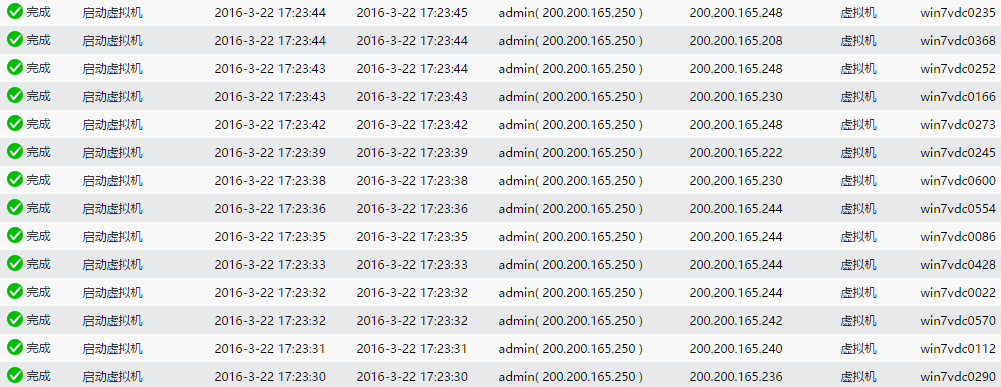


## 效果

效果当然是不错的，之前的启动1000台，平均一台2分钟，1天都没有启动完成。

现在实测600台启动耗时662秒，时间跨度17:16:08 - 17:27:10，启动1台平均1.1s。

截图一小幅图感受下（第三列是开始时间，第四列是结束时间）：



2016/3/28