服务压力测试报告

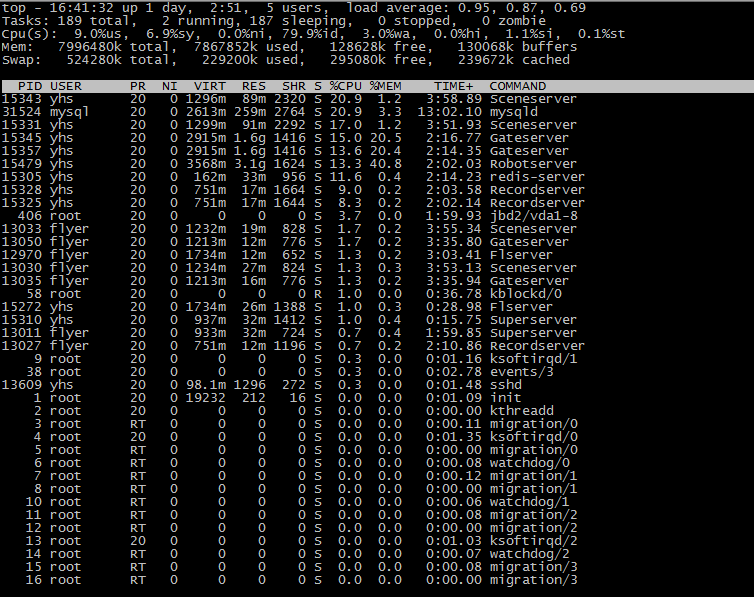
2015-12-09

测试环境：cpu： 2394.454 MHZ ;8 核

内存：8G

硬盘：100G

**第一轮测试**：监控数据 ：视图



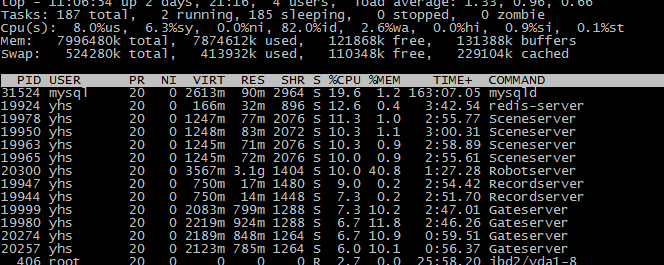
压入5000人，配置 2个scen ,2个gate,2个 record,1个flserver,操作：登陆 ，相互访问。上图可以看见：scen压力存在于cpu,进一步分析，瓶颈在于存档序列化。gate压力存在于内存。record单步调试时发现，存储较慢 1秒10 个玩家。

测试不顺当，因为外网网络不是很稳定，然而我的机器人采用的前台模式，因此机器人时常掉线。

测试时长约4个小时，以scen的崩溃而结束。

---------------------------------------------------------------------------------------------------------------------------------**第二轮测试**：修改崩溃的bug，将消息处理多线程改为单线程。机器人增加守护模式，自动登录，自动访问

压入5000人，配置 4个scen ,4个gate,2个 record,1个flserver,操作：登陆 ，相互访问。



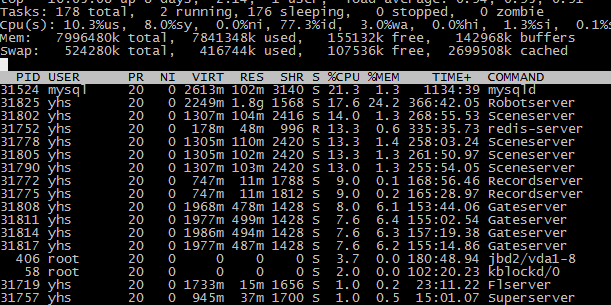
可以看见：gate scene 的资源消耗做到了平均分摊，消息处理速度也得到了改善。

测试时长 24小时，内存无异常，性能也稳定，日志较多。

---------------------------------------------------------------------------------------------------------------------------------

**第三轮测试**：对socket链接的缓存优化，内存缩减到原来的40%左右，因此压入更多人，调高了日志等级。

压入10000人，配置 4个scen ,4个gate,2个 record,1个flserver,操作：登陆 ，相互访问。



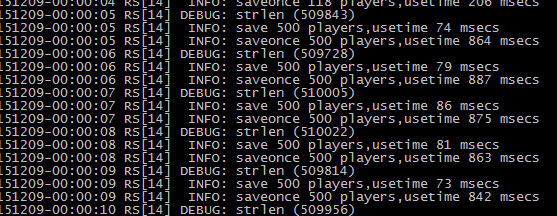
内存得到了减少。

测试时长 28小时，内存无异常，性能也稳定，日志调整了等级，得到改善。

---------------------------------------------------------------------------------------------------------------------------------**第四轮测试** ：该轮的重点在于，优化数据库存储速率。配置如前。

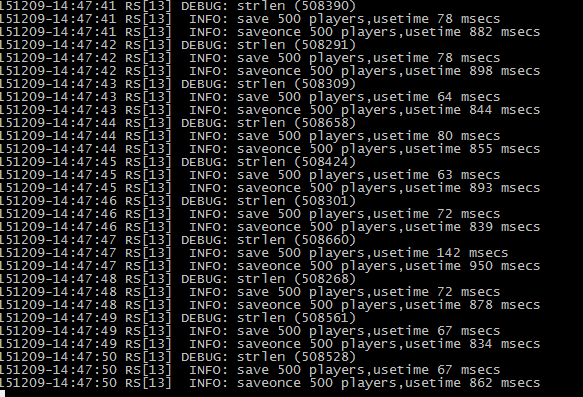
第一步：改善代码，使其支持批量存储，一次存多少，指标不清楚，暂为20。目的是测试其存储逻辑。在内网进行，测试成功。

第二步：原则上一批存储越多越好，但是也有个限制，那就是mysql语句长度限制，因此目标是只要不超过长度限制，存储越多越好。取出mysql长度限制，对字节进行计数。拉到外网测试



由上图，500个用户用时860毫秒左右，其中写入数据库约80毫秒（sql语句500k长度），效果较为理想。大部分时间耗在从redis取数据，存储对象的构建。因为redis取数据目前无法进行优化，那么存储对象构建就成为了重点。

第3步，存储对象预先构建：优点：省去了对象的构造和析构开销，同时避免了过多内存碎片 缺点：必须预先分配一块内存，提前构建对象，并且必须指定一个上限值，目前设置为1000 ，以后如果机器性能更厉害，那么还可以提高。

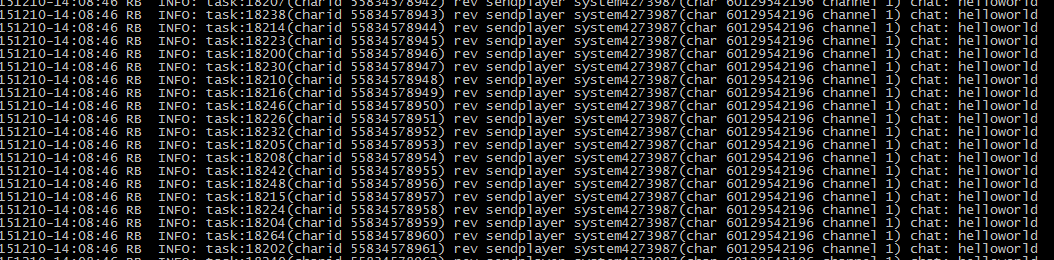


如上图：性能并无明显提升，可见问题主要在于redis取速度，代码保留，因为毕竟比先前版本性能更优（主要从cpu执行效率来讲）。

第四轮测试基本达到目的。

第五轮：测试目的：消息反应速度，世界聊天为例。

压入10000人，配置 4个scen ,4个gate,2个 record,1个flserver



反应速度非常快，因为这部分工作，主要由gate做的，因此速度不用担心，在一秒内所有玩家（10000）收到消息。

**本次压测结束 结论：10000人 服务器配比 fl：gate:sene:record 为 1:4:4:2比较合理，内存需求除去redis ,单个玩家内存为300k（10000个玩家约需求3G).**