云风的 BLOG
思绪来得快去得也快，偶尔会在这里停留

« Skynet 的服务监控及远程调用 | 返回首页 | skynet lua 服务的内存管理优化 »

skynet 服务启动优化

我们开发 6 个月的手游即将上线，渠道要求我们首日可以承受 20 万同时在线，100 万活跃用户的规模。这是一个不小的挑战，我们最近在对服务器做压力测试。

我们的服务器基于 skynet 构架，之前并没有实际跑过这么大用户量的应用，在压力测试时许多之前理论预测的问题都出现了，也发现了一些此前没有推测到的现象。

首先，第一个性能瓶颈出现在数以万计的机器人同时登陆系统的时候。这是我们预测到的，之前有为此写过一篇 blog 。

为了解决这个拥塞问题，我的建议是用这样一个系列的方案：

1. 用户认证不要接入 agent 服务。即，不要因为每个新连接接入都启动一个新的 agent 为它做认证过程。而应该统一在 watchdog 分发认证请求。当然这样就不可以用 skynet 默认提供的 watchdog 了。skynet 的源代码库中之所以实现了一份简单的 watchdog ，更多的是一个简单的示范。我们自己开发的两个项目都自己定制了它。
2. 认证的具体业务逻辑（例如需要接入数据库等），实现在一个独立的服务中，做成无状态服务，可以任意启动多份。由 watchdog 用简单的均匀负载的方式来使用。如有需要，再实现一套排队流程（参考 1, 参考 2 ），也由 watchdog 调度。
3. 我们目前这个项目的设计是唯一大服，所有用户在一个服务器中，要求承担百万级用户同时在线。所以我们在每台物理机上都配备了一个 watchdog ，通过内部消息在中心服务器统一协调。如果不这样设计，watchdog 会实现的更简单。watchdog 只负责维护用户在线状态，没有具体的计算压力，所以很难成为性能热点。
4. 当用户认证成功后，watchdog 启动一份 agent ，通知 gate (连接网关) 把用户连接重定向到 agent 上。后续用户的业务逻辑，都有一对一的 agent 为它服务。

由于 agent 是 lua 编写的，所以启动 agent 始终是一个开销很大的过程。加载 lua 代码比加载一个 C 编写的动态库要慢上不只一个数量级。在实际测试中，agent 的启动环节还需要通过 skynet 消息向一个中心（目前是 service manager）索取其它服务的地址。我刻意没有使用 skynet 提供的名字服务，因为 service manager 是用 lua 编写的，更容易定制。但这也会使启动 agent 更容易拥塞在某个单点。

启动 agent 过慢这个性能热点出现前，我们已经预备了一个方案。印证它的确是一个问题后，我们启动的预案：

我们在整个服务启动过程中，预算启动了 1000~5000 个 agent 待用。启动完毕后才开启对外端口。用一个定时器检查备用 agent 池是否枯竭，定期补充。这个方案只需要不到 100 行 lua 代码就可以完成，简单有效。躲过开服高峰期后，这就不会再是热点了。

注：不需要把 agent 池实现成可回收的。即不必在 agent 退出时归还。这浪费了 lua 作为沙盒的好处（用户断开连接就清理所有相关状态），也没有带来什么明显的性能好处，还增加了 agent 池实现的复杂度。

我们启动 agent 池方案后，发现预启动 1000 个 agent 在我们的服务器上居然长达 40 秒。平均每个居然要 40ms 秒之多。

我们的 agent 启动过程比较复杂，为了观测到热点在哪里，我 增加了 boot time 的统计 。不出所料，95% 以上的时间是花在 lua 脚本的加载上。我们的 1000 个 agent 加载的是相同的脚本，但是加载到不同的沙盒中。每个都要调用文件 IO 且 parser 源码。这两天我花了点时间把一直想做没做的功能实现了：在 skynet 中 cache 加载过的 lua 代码文件 ，不用每次都通过文件 IO 读取，并可以 cache 住源代码 parser 的结果。我用的是非侵入式方案，把自己写的 loader 注入到 lua 的 package.searchers 里。这个依赖 lua 5.2 的特性，可能在 luajit 上会有一点小问题。我们的项目没有使用 luajit ，所以暂时不会完善它。

做了这个简单的 code cache 后，启动时间从 40 秒下降到 20 秒，提高了一倍。

另一个困扰我两天，得不到合理解释的奇怪现象是：

如果我串行启动 1000 个 agent，每个启动完毕才启动下一个；比我并发启动 1000 个 agent，不用等待成功回应，居然要快一倍！

我剖析了启动时间，那 95% 的启动时间花在把 lua opcode 加载到 lua state 中。我们知道独立的 lua state 之间是没有任何关联的，不会有任何形式的锁，理论上并行不会有任何冲突。

我已经把 lua 服务的启动做成二步式，从 skynet 发起启动一个新的 lua state 和在 lua state 上加载代码是两个过程。所以 launcher 启动一个 agent 后，会有另一个工作线程去完成加载代码的工作。当启动串行时，可以大致看成 A 线程发起启动，创建 skynet service ； B 线程顺着在新启动的 service 上装载 lua 代码。

如果并行启动 1000 个 agent，势必让所有工作线程都同时启动，以流水线方式装载初始化这些沙盒。这些工作都是相互独立的，在多核环境下，理论上应该快一些；但实际上却更慢，且慢了整整一倍。

实际运行时分别在串行和并行环境下测试，并行环境下 CPU 负荷也高的多，但最终实际消耗的时间却长的多（人可以直接感受到时间差别）。排除了 skynet 中少量的 spin lock 可能造成的浪费，我不太明白何以造成这样的结果。难道是源于 CPU L1 Cache 的利用率不同？接下来我想花点时间仔细研究一下为什么。

最终我们的压力测试结果还是很让人满意的。我们配置有 64G 内存 6 core * 2 的服务器 6 台，可以轻松支撑 10 万用户在线，且游戏操作感觉流畅。我们单台机器的上限大约在 3 万用户（受限于内存），远远超出一开始的设计容量（之前我们希望可以做到单台机器 1 万用户就可以了）当然实际情况要等游戏上线才能明确了。

云风 提交于 December 18, 2013 03:46 PM | 固定链接

COMMENTS

多线程启动Lua VM比单线程的慢，这个问题解决了吗？是因为cache的问题？

Posted by: spin6lock | (18) October 24, 2014 06:01 PM

看了一段时间的skynet，也跟着重写了一些，有个问题想问下，在处理网络数据这一块是不是去掉了连接服务器的，还是说直接使用skynet作为一个单独的连接服务器（这样好像效率不太好），再有就是gate只负责读数据，发送直接用send，这样在下行数据较多的时候是否可行？

Posted by: yellowbug | (17) March 19, 2014 10:25 AM

并行化越多，可能会导致load balance在多核间切换时的cache失效，直接导致单个线程的启动速度降低，可以观察一下系统的io，如果频繁读取，那就一定是这个原因了。类似的问题也会出现在视频的多核转码上，一个数量级的差距都是可能的。

Posted by: reatdoom | (16) February 8, 2014 03:34 PM

skynet交流群 340504014 欢迎大家的加入

Posted by: 天空 | (15) January 8, 2014 05:46 PM

关注~求游戏名，想体验一下！

Posted by: Chocolate | (14) January 3, 2014 10:40 PM

是那个coc like的手游吗？这种基本上只有离线互动的游戏，单台机器只能支持3w人？太少了点吧。

Posted by: qiaojie | (13) December 27, 2013 12:16 AM

你们的手游叫什么名字呢，准备第一时间去体验下

Posted by: lee | (12) December 20, 2013 10:24 PM

并行的时候难道都花在了内核时间上，期待大神的解答

Posted by: Julius | (11) December 20, 2013 04:22 PM

刚才一想发现我说的不对劲，这么牛B的服务器哪能和PC是一个道理呢，就当我没说～

Posted by: luoye | (10) December 19, 2013 10:25 PM

我猜测是线程数太高，导致上下文切换的开销过大，书上说如果线程数等于CPU数量可以达到最佳性能，菜鸟路过啊，大神别虐我~

Posted by: luoye | (9) December 19, 2013 10:00 PM

64G 内存 6 core * 2 的服务器。

老外写的服务器引擎BigWorld终于败给了时间，可以光荣的退出历史舞台了。YEAH！

实力不够，民族来凑！性能不够，Money来凑！

另外，这是要从千万富翁升级为亿万富翁的节奏啊。恭喜恭喜！

Posted by: XXX | (8) December 19, 2013 04:12 PM

请问什么类型的手游呢

Posted by: txal | (7) December 19, 2013 12:48 PM

你们的游戏数据存储是用什么方案啊？
如果瞬间断电，会有大量数据丢失的可能吗？

Posted by: wi | (6) December 19, 2013 11:50 AM

大神！什么时候，开源你们的 2d游戏引擎呀！

Posted by: dave | (5) December 19, 2013 10:58 AM

Linux自从2.6内核开始，就会把不同的线程交给不同的核心去处理。Windows也从NT.4.0开始支持这一特性。线程切换也是要开销的，当你增加一个线程的时候，增加的额外开销要小于该线程能够消除的阻塞时间，这才叫物有所值。

Posted by: smite | (4) December 19, 2013 10:26 AM

有错别字，“一经正式”，应该是“一经证实”吧？

简悦公司鼓励玩家报告游戏系统、程序、设定等方面存在的问题或漏洞，一经正式将可以获得奖励。

Posted by: Anonymous | (3) December 18, 2013 09:49 PM

请教一个问题，同步模式下（发一个收一个，再发一个收一个），8线程，每个线程32个socket，使用epoll来管理数据包的收发，总吞吐量可达5w个数据包每秒，数据包的平均时延在2ms内。
而在异步模式下，每个线程一个socket，控制每秒内发送的频数，每次发N个包，例如每秒发1000次，每次10个数据包，但是吞吐量一旦超过某指，时延非常大，5s,而且吞吐量超不过2w，什么原因？

Posted by: remap | (2) December 18, 2013 07:29 PM

感觉像这种多线程效率反而没有单线程高的问题，问题出在 cache 上的概率极大～

Posted by: Grissiom | (1) December 18, 2013 05:30 PM

POST A COMMENT

非这个主题相关的留言请到：留言本

名字：

Email 地址：

为了验证您是人类，请将六加一的结果（阿拉伯数字七）填写在下面：

URL:

记住我的信息?

留言:
（不欢迎在留言中粘贴程序代码）

提交