« 开发笔记(24)：Lua State 间的数据共享 | 返回首页 | Skynet 集群及 RPC »

Skynet 开源

最近两天是我们项目第二个里程碑的第一个检查点。我们的服务器在压力测试下有一些性能问题。很多方面都有一个数量级的优化余地，我们打算先实现完功能，然后安排时间重构那些值得提升性能的独立模块。

我最近两周没有项目进度线上的开发任务。所以个人得以脱身出来看看性能问题。前几天已经重新写了许多觉得可能有问题的模块。在前几天的 blog 里都有记录。

虽然没有明显的证据，但是感觉上，我们的服务器底层框架 skynet 有比较大的开销。这个东西用 Erlang 开发的，性能剖析我自己没有什么经验。总觉得 Erlang 本身代码基有点庞大，不太能清晰的理解各个性能点。

其实底层框架需要解决的基本问题是，把消息有序的，从一个点传递到另一个点。每个点是一个概念上的服务进程。这个进程可以有名字，也可以由系统分配出唯一名字。本质上，它提供了一个消息队列，所以最早我个人是希望用 zeromq 来开发的。

现在回想起来，无论是利用 erlang 还是 zeromq ，感觉都过于重量了。作为这个核心功能的实现，其实在 2000 行 C 代码内就可以很好的实现。事实上，我最近花了两个整天还不错的重新完成了这个任务，不过千余行 C 代码。当然离现在已有的框架功能，细节上还远远不够，但能够清晰的看到性能都消耗到哪些位置了。其实以后不用这个 C 版本的底层框架，作为一个对比测试工具，这半周时间也是花得很值得的。

我将这两天的工作开源到了 github 上，希望对更多人有帮助。从私心上讲，如果有同学想利用这个做开发，也可以帮助我更快发现 bug 。有兴趣的同学可以在这里跟踪我的开发进度 。

关于接口，我在上面提到的 blog 中已经列过了。这次重新实现，发现一些细节上不合理的地方，但是不太好修改，姑且认为是历史造成的吧。

在目前的版本里，我还没有实现跨机器通讯，我也不打算讲跨机通讯做到核心层中。而希望用附加服务的方式在将来实现出来。

这个系统是单进程多线程模型。

每个内部服务的实现，放在独立的动态库中。由动态库导出的三个接口 create init release 来创建出服务的实例。init 可以传递字符串参数来初始化实例。比如用 lua 实现的服务（这里叫 snlua ），可以在初始化时传递启动代码的 lua 文件名。

每个服务都是严格的被动的消息驱动的，以一个统一的 callback 函数的形式交给框架。框架从消息队列里取到消息，调度出接收的服务模块，找到 callback 函数入口，调用它。服务本身在没有被调度时，是不占用任何 CPU 的。框架做两个必要的保证。

一、一个服务的 callback 函数永远不会被并发。

二、一个服务向两一个服务发送的消息的次序是严格保证的。

我用多线程模型来实现它。底层有一个线程消息队列，消息由三部分构成：源地址、目的地址、以及数据块。框架启动固定的多条线程，每条工作线程不断的从消息队列取到消息。根据目的地址获得服务对象。当服务正在工作（被锁住）就把消息放到服务自己的私有队列中。否则调用服务的 callback 函数。当 callback 函数运行完后，检查私有队列，并处理完再解锁。

线程数应该略大于系统的 CPU 核数，以防止系统饥饿。（只要服务不直接给自己不断发新的消息，就不会有服务被饿死）

由于我们是在同一个进程内工作的。所以我对消息传递做了一点优化。对于目前的点对点消息，要求发送者调用 malloc 分配出消息携带数据用到的内存；由接受方处理完后调用 free 清理（由框架来做）。这样数据传递就不需要有额外的拷贝了。

除了核心功能，我们还需要提供一些基础功能才可以做点真正的事情。

一个是简单的黑洞（blackhole），当消息没有接收者时，它可以接受到消息，并服务清理消息占用的内存。

一个是简单的错误信息记录器（logger）。内部错误信息不应该用简单的 printf 输出，这是因为在多线程模型下，这样会造成混乱。用一个独立服务，讲 log 信息串行化要清晰的多。有必要的话，可以加工这些信息。

启动新的服务和杀掉服务我把它们做到了框架内，以 skynet command 的形式提供。按原本的项目，应该有一个额外的服务管理器的东西来做这些事情。但我发现，大多数情况下，我希望知道我启动的服务的地址，以方便做后续操作。如果用一个管理器服务的形式来工作，虽然可以简化核心，但必须建立一套 RPC 协议出来。这次我不打算在核心层约定 RPC 规范，所以就选择放在了核心指令内。

Timer 及时间服务是一项基础功能。所以我实现在了框架内。特别是 timeout 为 0 的特例，是不进入 timer 队列，而是直接进入消息队列。这次我提供了 1/1000 秒的时间精度，以及 1/100 精度的 timeout 回调，对于游戏服务感觉是够用了。

对于 MMO 的基础需要，我提供了 gate 的独立服务，用来处理大量的外部链接。这个是 前段时间用 epoll 实现的 。稍微做了些小修改就用上了。

这个只解决读外部链接的问题，暂时还没有实现发送的部分，接下来的时间我会完善它。

其工作方式是，启动 gate 服务后，根据启动参数， listen 一个端口。接受连接上来的所有外部连接。gate 会为每个连接赋予一个唯一 id 号。注意，这个 id 号是尽量不复用的。在 skynet 的生命期内是单调递增的。这是因为在这样的多服务并发的复杂系统内，短期复用 id 是很危险的一件事。我用了一个简单的方法（保证不冲突的 hash 表）来解决高效的映射关系。

gate 会默认将所有外部连接的相关消息（连入，退出，有数据到来）发送给一个叫 watchdog 的服务。

并且，它接受一些控制指令，可以主动断开外部连接；或是把这个特定外部连接的数据绑定到另一个不是 watchdog 的地址。

在目前的范例中，watchdog 用 lua 实现。当一个外部连接接入，它会启动一个类型为 agent 的服务（也是用 lua 编写），并通知 gate 绑定这个外部连接的数据到新启动的 agent 上。

关于外部连接的 client , 我简单的要求, 它必须是按一个个数据包发送数据过来, 每个数据包有一个两字节的大头数字表示包长. 我给出了一个 client.c 做简单的测试工作。

---

lua 服务是另一项基础设施，但不属于核心部分。如果你喜欢，也可以用 python 等其它动态语言替换掉。

这里叫 snlua ，以和系统内的 lua 区分。同时我提供了一个 skynet 的 lua 库，可以给 lua 程序实现。当然，如果不是在 snlua 环境中 require 这个 skynet 库的话，是不能正确工作的。

和之前 erlang 的版本比较，我在设计上做了一些修改。比如，并没有在核心层规定通讯协议，而之前默认一定用 protobuffer 来做消息通讯。

基础服务间的控制指令，目前基本都用简单文本协议。

还没有实现大量细节的配置表，以及组播、跨机通讯等等。这些繁琐的工作可能要花掉我接下来几周的时间，才可能无缝的接入现在已有的系统。

云风 提交于 August 1, 2012 12:28 PM | 固定链接

COMMENTS

建立一个固定 slot 数量的 hash 表,能不能用word写出来啊

Posted by: word | (22) August 28, 2012 02:44 PM

@David Xu

谢谢您的点评！您说得非常到位。BSD还是更重视响应啊。

Posted by: zelor | (21) August 15, 2012 09:59 PM

做游戏用得着这么复杂吗？

Posted by: leodo | (20) August 11, 2012 02:57 PM

@zelor
BSD可能会好一点，主要原因是BSD在一个线程阻塞于I/O后，再醒来时会暂时拔高线程的优先级，不是位于动态调整的用户态优先级范围，而是高于所有用户态级别，这样的设计主要是为了缩短响应时间，但是对于吞吐量会有所下降，主要原因是可能引起上下文切换的次数比较多。

Posted by: David Xu | (19) August 9, 2012 10:39 AM

@David Xu

居然大神也来了……

对于请求-应答模式，确实是性能杀手。尤其是遇到LINUX之类高吞吐量低响应型的OS，不知BSD如何？
我在几年前的项目里有比较郁闷的教训。

Posted by: zelor | (18) August 8, 2012 11:06 PM

你的恶源代码里面的代码风格感觉不是很好，比较古老的大妈风格吧。
还有那个harbor参数是什么意思（什么作用？）。

Posted by: XXX | (17) August 8, 2012 10:11 AM

一个是简单的错误信息记录器（logger）。内部错误信息不应该用简单的 printf 输出，这是因为在多线程模型下，这样会造成混乱。用一个独立服务，讲 log 信息串行化要清晰的多。有必要的话，可以加工这些信息。

不懂 ，log信息详细点，线程安全就可以了，为什么要单独线程服务呢，这样也未必能保证串行化。

Posted by: Anonymous | (16) August 5, 2012 09:45 AM

多线程也没那么可怕。OS内核那么复杂的多线程模型，都能写出来。他们也是人，为什么就可以呢？

Posted by: David Xu | (15) August 4, 2012 09:01 PM

前不久云风老大不是实现了一套服务器，用的是单线程多进程模型吗？现在开始使用单进程多线程了，不知是如何考虑的。

Posted by: winapp | (14) August 4, 2012 11:51 AM

@David Xu

我们的结构, 在底层是没有 rpc 的. 全部都是点对点的消息.

不过 erlang 做了些什么, 我也不是很清楚.

另外性能问题, 我觉得多半还是 lua 层的某些东西搞的. 我也不指望把底层重写能有数量级的提高.

这部分做完, 再折腾 luajit 的工作.

Posted by: Cloud | (13) August 4, 2012 01:06 AM

不知道是不是节点之间都用了请求应答方式，这种方式很容易有性能问题，性能会受到路径长途的影响，包括网路距离。但是节点之间可以有很高的带宽，如果程序采用流式数据处理，而不是等待应答再请求下一个，而是有请求就发送，有数据回来就处理，则可充分利用带宽优势。一个简单的例子，我们到月球的距离很遥远，但是无线电以光速的带宽前进，带宽做的很高，但是延迟解决不了，延迟很大，大概几分钟后才达到月球。这个例子可能不是最好的，但是足以说明问题。
以前cvs update很慢，但是有了cvsup这个软件就很快，因为后者不是请求应答的，而是边发送请求边接收处理结果，充分利用了流式处理。

Posted by: David Xu | (12) August 3, 2012 03:44 PM

@David Xu

目前的系统无法确定到底是不是框架的性能有问题.

我决定用有限时间 (一周) 重写这个框架. 至少可以有个对比.

这样,如果整体性能有显著提高,就说明原来的框架有问题. 如果没有,说明原来的没问题,继续用原来的就好了.

我觉得这样做,比研究 erlang 的实现要快的多.

btw,在此之前,我们已经花了不少人力研究性能问题到底出在哪里,一直没有明确的结论. 花掉的时间和精力已经远远超出我重写这个的时间了.

Posted by: Cloud | (11) August 3, 2012 12:26 PM

erlang 有些性能不理想吗?到底什么原因?

Posted by: David Xu | (10) August 3, 2012 10:08 AM

并发的hash可以参考ConcurrentHopscotchHashMap，这个实现考虑了cacheline对齐，可以达到很高的并发性能

Posted by: marble | (9) August 2, 2012 02:06 PM

請問是否有嘗試過gearman?目前我們新專案正以gearman為基礎，基本上效能還不錯，也有多種語言的binding。。
http://gearman.org/

Posted by: mars | (8) August 2, 2012 09:10 AM

看上去很像zmq干的事情：为各个服务搭建一个沟通的信道，也就是类似消息队列的感觉. 好处是很轻量

fork来看看，有没有可能和python集成到一起.

Posted by: c4pt0r | (7) August 1, 2012 08:24 PM

虽然只看懂一部分，但还是下来看看!

Posted by: baohuams | (6) August 1, 2012 04:18 PM

@guest

首先， 你可以看代码。

其次,这个方法很简单。 建立一个固定 slot 数量的 hash 表，新增 id 如果发现和原来的 id hash 冲突，就继续加一，直到在 hash 表中找到一个不冲突的位置。

Posted by: Cloud | (5) August 1, 2012 03:15 PM

大师，讲讲不冲突的hash表是怎么回事。google未果。谢谢

Posted by: guest | (4) August 1, 2012 02:00 PM

下来看看，支持开源精神~

Posted by: anders0913 | (3) August 1, 2012 01:31 PM

直觉上，单进程多线程模型的优点就是没有IPC，通信效率会很高；
但一个明显的缺点是，假如哪个地方的code有问题crash掉了，整个进程就挂了。

Posted by: Mine | (2) August 1, 2012 01:12 PM

不错，下下来研究看看

Posted by: yourihua | (1) August 1, 2012 12:41 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