云风的 BLOG

思绪来得快去得也快，偶尔会在这里停留

« 一个适用于腾讯开放平台的 tunnel | 返回首页 | Linode 服务真不错 »

重新设计并实现了 skynet 的 harbor 模块

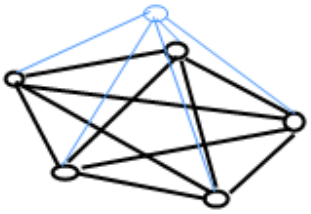
skynet 是可以启动多个节点，不同节点内的服务地址是相互唯一的。服务地址是一个 32bit 整数，同一进程内的地址的高 8bit 相同。这 8bit 区分了一个服务处于那个节点。

每个节点中有一个特殊的服务叫做 harbor (港口)，当一个消息的目的地址的高 8 位和本节点不同时，消息被投递到 harbor 服务中，它再通过 tcp 连接传输到目的节点的 harbor 服务中。

不同的 skynet 节点的 harbor 间是如何建立起网络的呢？这依赖一个叫做 master 的服务。这个 master 服务可以单独为一个进程，也可以附属在某一个 skynet 节点内部（默认配置）。

master 会监听一个端口（在 config 里配置为 standalone 项），每个 skynet 节点都会根据 config 中的 master 项去连接 master 。master 再安排不同的 harbor 服务间相互建立连接。

最终一个有 5 个节点的 skynet 网络大致是这样的：



上面蓝色的是 master 服务，下面 5 个 harbor 服务间是互连的。master 又和所有的 harbor 相连。

这就是早期的 skynet 分布式集群方案。有一篇 2 年前的 blog 记录了当时的想法，可以一窥历史。

由于历史变迁，从早期的手脚架不全，到如今的 skynet 的基础设置日臻完善。这部分代码也改写过很多次，每每想做大的改动，都不敢过于激进。

最近，我们的一个新项目要上线，由于运营方只能提供虚拟机，且网络状态不是很好，暴露了 skynet 在启动组网阶段的一些时序漏洞。所以这个周末我咬牙把这块东西全部重新设计实现了。

当初为了简化设计，每两台机器间的连接使用了两条 TCP 连接。数据流在每条连接上都是单向的，即谁发起连接，谁就在这个连接上单向推送数据。这样做的好处是，如果双方都是可信的机器的话，可以省去握手的协议。

如果采用一条连接，双工使用，势必需要在接受连接时询问对方是谁。组网代码的复杂度就高了许多。

但是，两条连接的问题也很明显。当我可以向对方发送数据成功后，对方未必反向连接成功。就需要做更复杂的状态管理。当然，一旦组网成功，就没有太大区别了。

这块代码一开始就是 C 语言编写的，最早期利用 zeromq 搭了初期的雏形，后来又利用独立的 gate 服务重新实现了早期协议（实现方法上还留有 zeromq 的痕迹）；再后来 gate 服务用后期完善的 socket 模块改写，一直演化到今天。在这段时间里，我充分意识到，让多台机器两两互连，组成一个通讯网络是一个及其复杂的事情。而组网本身又不要求特别高的效率，对效率有要求的只是网络组建好之后的消息转发而已。

所以把这快业务分离出来，用 lua 来编写组网模块要更合适一些。

这个周末，我按新的想法重新实现了这块代码。由于全部重新实现，就顺便把协议也重新设计了。

现在，节点间不再需要两条连接，而只用一条。每个节点加入网络（首先接入 master）后，由 master 通知它网络中已有几个节点，他会等待所有现存节点连接过来。所以连接建立后，就关闭监听端口。

如果再有新节点加入网络，老节点主动去连接新节点。这样做的好处是，已经在工作的节点不需要打开端口等待。

这套代码实现在 cmaster.lua 和 cslave.lua 中，取代原来的 `service_master.c` 和 `service_harbor.c` ，用 lua 编写有更大的弹性。这两个服务还负责同步 skynet 网络中的全局可见的服务名字，原本在 C 版本中，这部分实现的很繁琐，

改为 lua 后，清晰了许多。

原有的远程消息代理模块被剥离出来，还是放在 service_harbor.c 中，比之前的代码篇幅小了很多。它只负责管理 tcp 连接的 fd, 而不必操心连接连接的过程。这样，也不再依赖额外的 gate 服务，只需要做简单的拼包处理即可。

至此，C 版本的 gate 已经完全脱离了核心模块。在一般应用中，后来写的 lua 版 gate 已经足够用了。

此外，对于松散的集群结构，我推荐使用 skynet 的单结点模式，在上层用 tcp 连接互连，并只使用简单的 rpc 协议。在目前的 skynet 版本中，有封装好的 cluster 模块 可供使用。

这种做法要求明确本地服务的调用和远程调用的区别。虽然远程调用的性能可能略低，但由于不像底层 harbor 那样把本地、远程服务的区别透明化，反倒不容易出问题。且 tcp 连接使用了更健壮的 socketchannel ，一旦连接断开，发起 rpc 的一方会收到异常，也可以重试（自动重连）。

而底层的 harbor 假设机器间是可靠连接，不会断开。而一旦内部网络不健康，很可能会导致整个系统无法正常工作。它的设计目的并不是为了提供弹性扩展的分布式方案，而是为了突破单机性能上限的问题。

两个跨机方案各有利弊，所以还请设计系统的时候权衡。只使用其中一个方案或是两个同时用，应该都有适用的场合。

这次大的修改，将作为 0.4.0 版的核心特性发布。不过我想把它们在 dev 分支 放一段时间，希望有能力的同学可以帮忙 review 一下代码，或做一些测试。

ps. 最近 skynet 社区比以前活跃了许多。看的到不少利用 skynet 开发的游戏项目了。为 skynet 减少了不少 bug。感谢大家。

云风 提交于 June 21, 2014 08:02 PM | 固定链接

COMMENTS

请问一下这里的master是单点的吗？

Posted by: OWenT | (5) July 24, 2014 01:30 PM

如果skynet中的一个节点网络不稳定，与其它节点断开了，会自动重连吗？

Posted by: ai | (4) July 5, 2014 05:45 PM

@David Xu

数量刚凑够.
今天说是明天去印花.

Posted by: Cloud | (3) June 23, 2014 08:21 PM

我订购的T恤呢？发货了吗？

Posted by: David Xu | (2) June 23, 2014 06:01 PM

mark !

Posted by: rvrv | (1) June 21, 2014 08:42 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