云风的 BLOG

思绪来得快去得也快，偶尔会在这里停留

« 一个 Bump Pointer Allocator | 返回首页 | skynet 服务启动优化 »

Skynet 的服务监控及远程调用

基于 Actor 模式的框架，比较难解决的问题是当一个 actor 异常退出后如何善后的问题。

Erlang 的做法简单粗暴，它提供了 spawn_link 方法，当一个 process (Erlang 的 Actor) 退出后，可以把和它关联的 process 也同时退出。

在 skynet 中，一开始我并不想在底层解决这个问题。我希望所有的 service 都是稳定的。如果一个 service 可能中途退出，那么在上层协调好这个关系。

而且 skynet 借助 lua 的 coroutine 机制，事实上在同一个 lua service 里跑着多个 actor 。一个 lua coroutine 才是一个 actor 。粗暴的将几个 service 在底层绑定生命期不太合适。

但是，如果有 service 有中途退出的可能，那么利用 skynet.call 调用上面的远程方法就变得不可靠。简单的用 timeout 来解决我认为只是回避了问题，而且会带来更多的复杂性。这是我在设计 skynet 时就想避免的。所以我在处理 service 生命期监控的问题上，做的比较谨慎。

前段时间我在 skynet 底层加上了 monitor 机制，可以把服务退出的消息从框架层抛出来，让上层逻辑可以感知到。但我并不想固定处理服务退出事件的逻辑，所以加了一个 skynet.monitor 方法把一个特殊服务注册入框架，让它全权负责处理这类消息。

btw, 这里我刻意回避了使用命名服务的机制，没有把 monitor 起一个特定名字。（例如：早期我就给 launcher 起了名字。）这是因为我觉得具名服务不应该在框架底层实现，而应该放在上层机制中。目前 skynet 支持的名字服务仅仅是出于历史兼容原因存在。新增加的代码我也不想再依赖这个特性了。

同时，我写了一个简单的 monitor 做为示范。在我们自己的项目中，对应的服务逻辑要复杂的多。

显然，光有 monitor 机制仅仅提供了解决问题的可能，把 skynet.call 做的更完备是不够的。今天的 skynet 更新 patch 中，我尝试完善了一下。

首先，增加了 skynet.watch 这个 api ，用于显式关注一些可能不稳定的服务。我不想给原有的 skynet.call 加太多的负担，毕竟大多数服务都是稳定不会退出的。（一旦意外退出，整个系统都会不可用，基于保持运行也意义不大）

skynet.watch 会向 monitor 汇报，当 monitor 发现服务消失的那一刻，将发消息通知。

skynet.call 在调用一个被 watch 的服务上的方法时，把 session/service 记录在一张表里。当收到 monitor 的反馈消息时，将从表中找到需要恢复的 coroutine ，把它唤醒并抛出异常。这样，skynet.call 就不会因为调用服务在处理远程请求过程中异常退出而无法返回的情况了。

目前还没有实现异常传递的机制。也就是当一个 coroutine 发生异常，而它又是被远程调用的方法，那么最好是能把这个异常传递回去。这套机制在我们的项目中有实现，但我还没有想好怎样合并到 github 的 skynet 核心代码树上。

12 月 10 日补充:

想了一下, 发现异常传播可以利用上面的通道进行,只需要增加几行代码即可. 所以今天的 patch 把这个功能加上了。这样，一但 skynet.call 调用的远程方法发生了异常后，调用者这边也会同样抛出一个异常，而不是挂起。

云风 提交于 December 9, 2013 05:53 PM | 固定链接

COMMENTS

想问问 云风大哥

```
static void *
_timer(void *p) {
```

```
struct monitor * m = p;
for (;;) {
skynet_updatetime();
#warning 不明白这里一直不会执行到后边去
CHECK_ABORT
wakeup(m,m->count-1);
usleep(2500);
}
```

Posted by: sky | (8) March 7, 2014 04:58 PM

那么在上层协调后这个关系。

“后” 改为 “好”

Posted by: wardenlym | (7) December 12, 2013 05:01 PM

@lpk

monitor 现在并不限制在同一个进程内.

@asking

keep alive 只是一个手段而已. 无所谓怎样实现,在哪里实现. 它解决的问题就是在超时没有信号反馈时,判定连接中断.

RPC 就不需要额外再做一次 timeout 的处理. 因为 RPC 是工作在连接层之上的层次的. 它不需要关心超时.

Posted by: Cloud | (6) December 11, 2013 04:14 PM

tcp keep alive也不可靠。http://250bpm.com/blog:22

Posted by: asking | (5) December 11, 2013 03:08 PM

我指的是远程harbor后面的服务，这类服务退出后，harbor间的连接并没有断开

Posted by: lpk | (4) December 11, 2013 02:33 PM

超时是不需要的.

如果远程服务是通过 tcp 连接, 而又设置了 keepalive 的话.

那么连接断开一定可以捕捉到.

Posted by: Cloud | (3) December 11, 2013 10:57 AM

如果是远程服务的话，那只能超时机制了吧？

Posted by: lpk | (2) December 10, 2013 08:49 PM

不错，感谢分享，期待后续更多哈。http://www.haobitou.com/#2

Posted by: 好笔头业务云笔记 | (1) December 10, 2013 02:10 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