零声学院 Mark 老师 QQ:2548898954

主题

- 多核并发编程
  - 多线程
  - 多进程
  - CSP模型
  - actor模型
- actor模型详解
  - 定义
  - 原理图
  - 构成
  - actor调度
  - actor与网络
- 游戏实现
  - 游戏简介
  - 设计原则
  - actor模型开发思路
  - 游戏展示

多核并发编程 N:M

多线程

> 并发实体为线程；开发难点：锁的使用以及粒度控制；
>
> 线程，是程序执行流的最小单元，是进程中的一个实体，是被系统独立调度和分派的基本单位，线程自己不拥有系统资源，只拥有一些在运行中必不可少的资源，但它可与同属一个进程的其它线程**共享进程资源**，线程的切换一般也**由操作系统调度**。
>
> （运行环境）隔离性差，（运行数据）统一性强；

多进程

> 并发实体为进程；开发难点：服务粒度控制，服务间数据一致性问题；最终一致性 强一致性
>
> 进程的缺点是创建、销毁以及切换的开销比较大。
>
> （运行环境）隔离性强，（运行数据）统一性差；

csp模型

> 以go为例，并发实体为goroutine，也就是协程（轻量级线程）；相较于线程，优点在于可以轻松创建上百万个而不会导致系统资源衰竭；channel
>
> 协程（创建、销毁以及切换）交由用户来调度；

actor模型

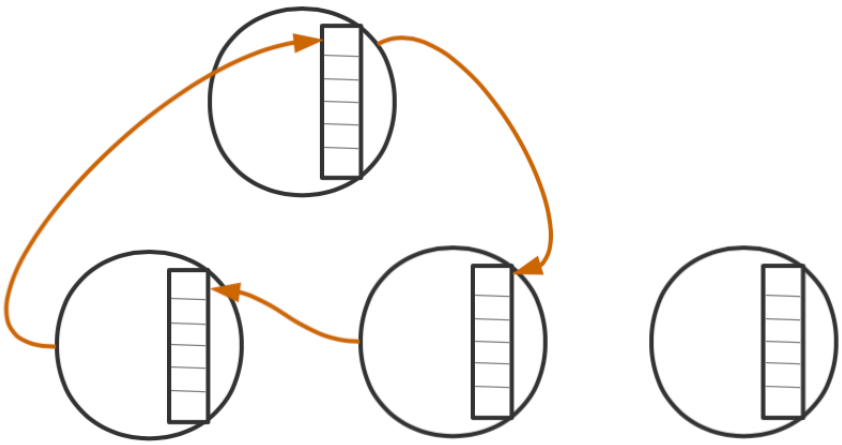
> 并发实体为actor，也就是轻量级进程；相较于进程，优点在于可以轻松创建上百万个而不会导致系统资源衰竭；
>
> actor交由用户来调度；
>
> erlang从**语言层面**支持actor，skynet从**框架层面**支持actor；

Actor模型详解

定义

- 用于并行计算-充分利用多核
- actor是最基本的计算单元
- 基于消息计算
- actor之间通过消息沟通并且相互隔离

原理图



> 如图，有三个actor之间互相发送消息，另外有一个非活跃的actor；

构成

- 隔离的环境

  > 在skynet中，隔离的环境可以是lua虚拟机，也可以一个c内存块；

- 消息队列

  > 将消息按照到达顺序依次存储，消息的执行也将按照到达顺序执行；

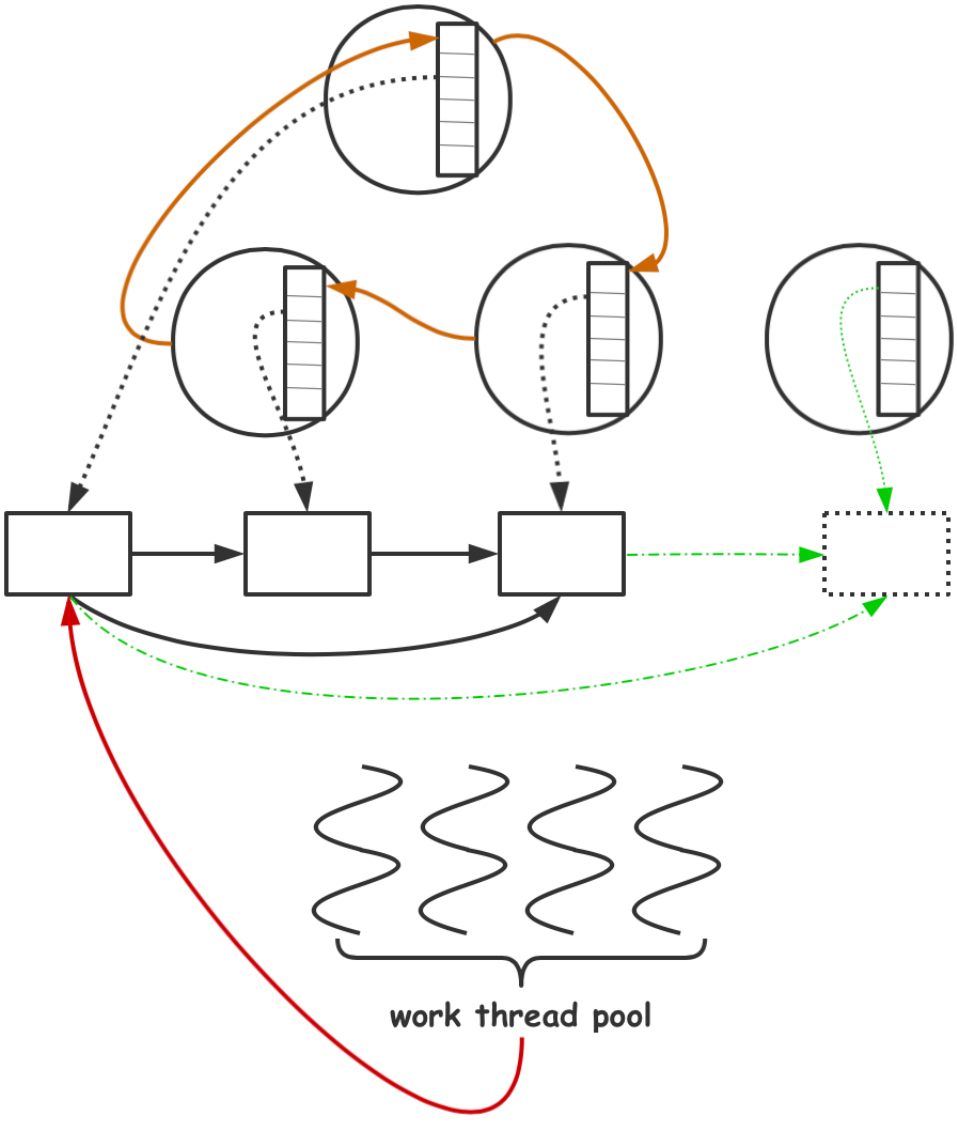
- 运行函数

> actor需要提供函数来消费消息；该函数的运行实体是lua协程；
>
> 在线程中，创建线程需要提供函数指针，在协程当中也是，也需要提供函数指针；
>
> PS：在一个lua虚拟机中，同一个时刻只有一个协程在运行；

```c
// 在c中创建一个线程
int pthread_create(pthread_t *restrict tidp,
    const pthread_attr_t *restrict attr,
    void *(*start_rtn)(void),
    void *restrict arg);
```

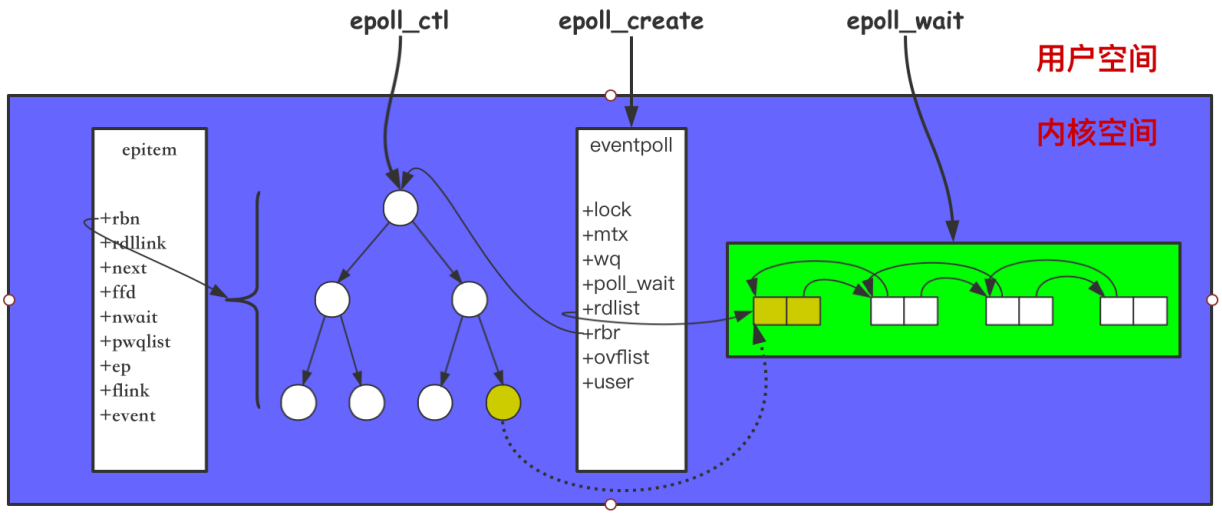
```lua
-- 在lua中，创建一个协程
coroutine.create(func)
coroutine.wrap(func)
```

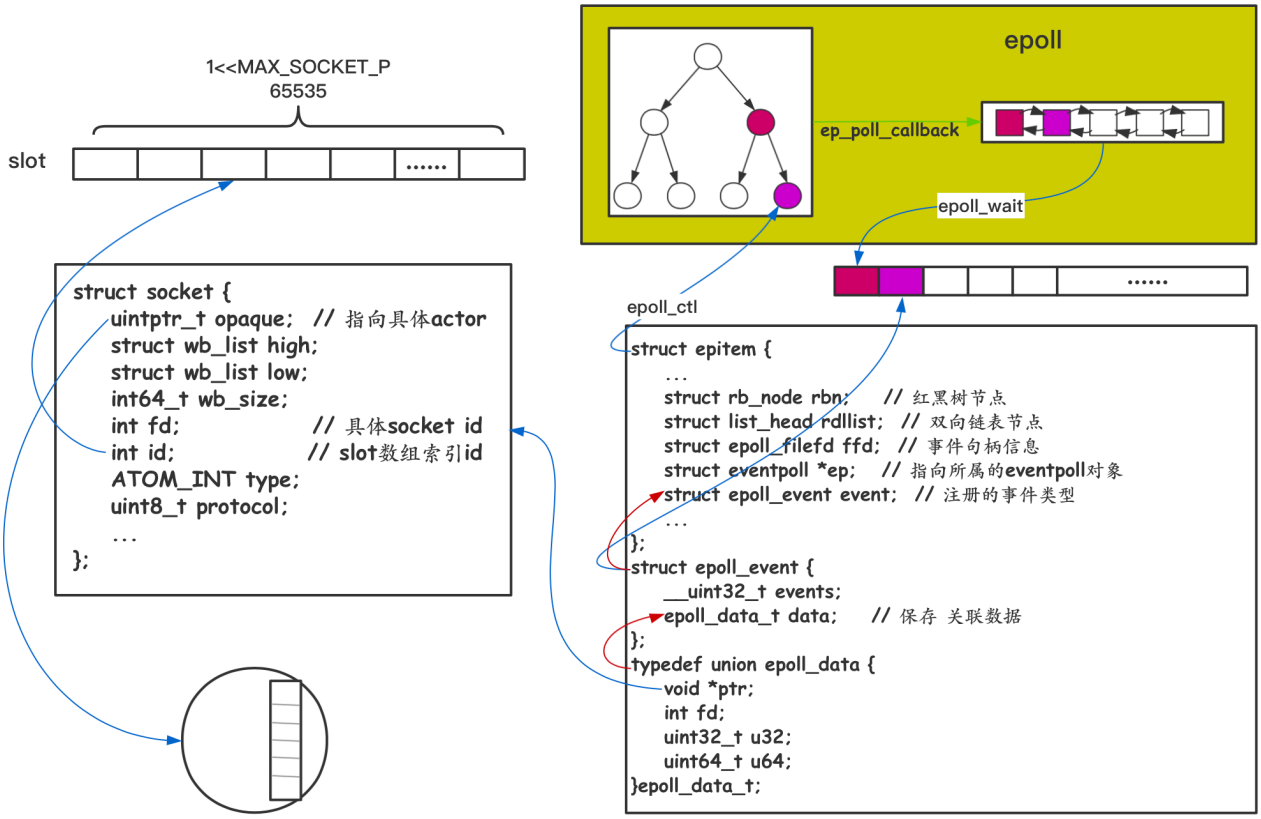
actor调度



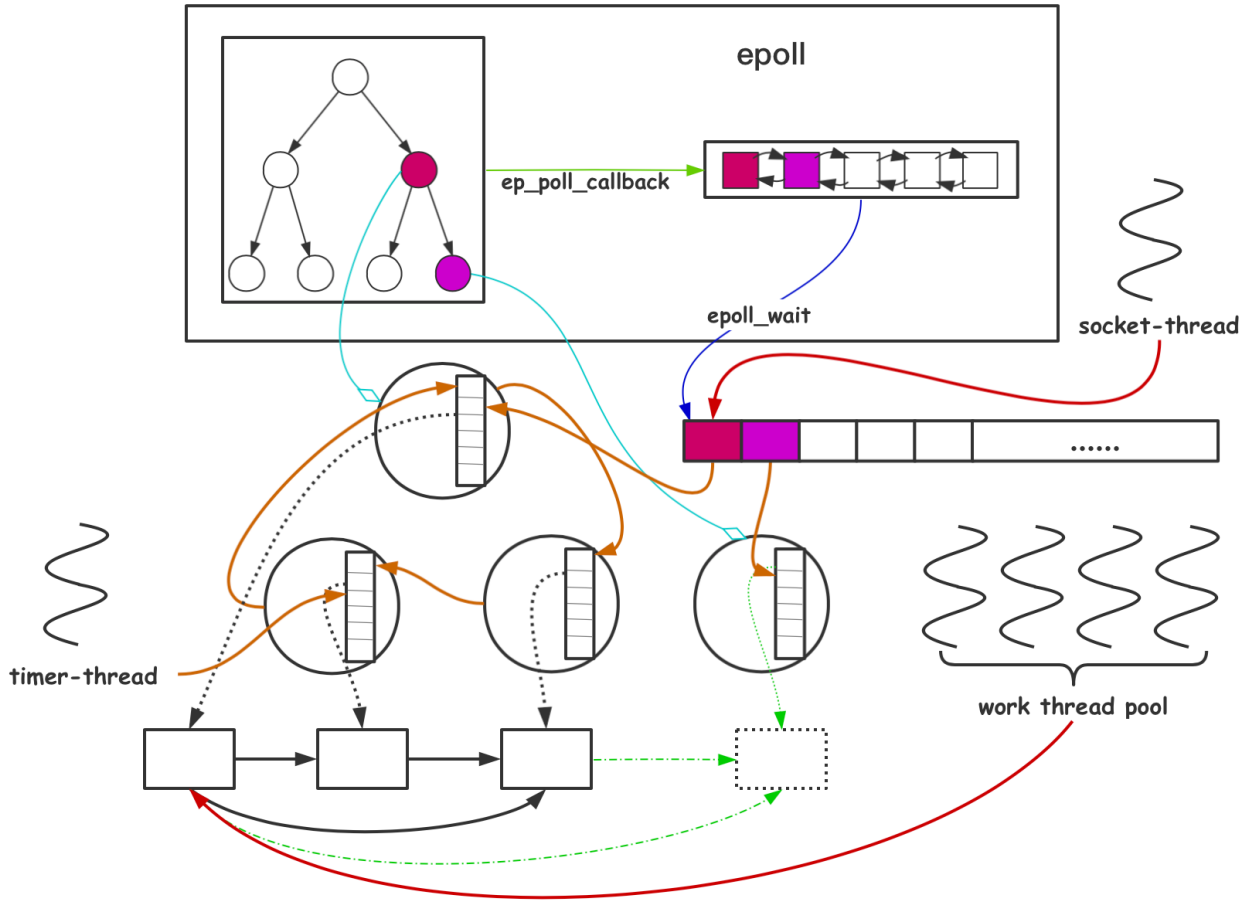
> 根据是否有消息来定义该actor是否活跃，将活跃的actor中消息队列串联起来，组成一个全局消息队列；然后通过线程池来消费全局消息队列中的消息；

actor与网络 epoll





总体图



游戏实现

游戏简介

> 目的：掌握actor模型开发思路；
>
> 游戏：猜数字的游戏；
>
> 条件：满3人开始游戏，游戏开始后不能退出，直到这个游戏结束；
>
> 规则：系统当中会随机1-100之间的数字，参与游戏的玩家依次猜测规定范围内的数字；如果猜测正确那么该玩家就输了，如果猜测错误，游戏继续；直到有玩家猜测成功；

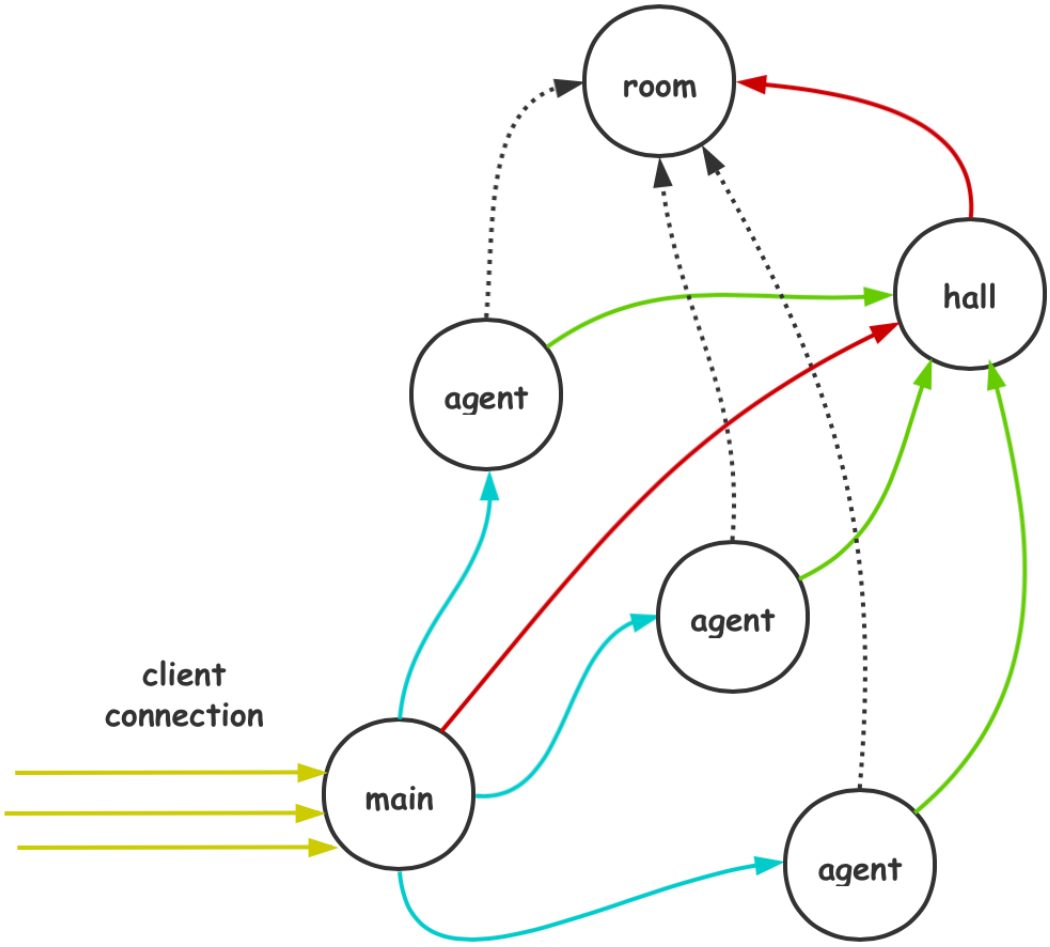
设计原则

> 简单可用，持续优化，而不是一开始就过度优化；

actor模型开发思路

服务的划分

> 单一职责原则：把因相同原因而变化的东西聚合在一起，而把因不同原因而变化的东西分离开来；
>
> 根据业务功能的边界来确定服务的边界；
>
> 根据功能划分服务，根据业务计算密集程序扩展服务；



接口的设计

> skynet中，从actor底层看是通过消息进行通信；从actor应用层看是通过api进行通信；这里有点类似restful api；
>
> 接口隔离原则：不应该强迫客户依赖于他们不用的方法；从安全封装的角度出发，只暴露客户需要的接口；服务间不依赖彼此的实现；

- agent

```lua
local CMD = {}

function CMD.login()
end
function CMD.ready()
end
function CMD.guess()
end
function CMD.help()
end
function CMD.quit()
end
```

- hall

```lua
local CMD = {}

function CMD.ready()
end

function CMD.offline()
end
```

- room

```lua
local CMD = {}

function CMD.start()
end

function CMD.online()
end

function CMD.offline()
end

function CMD.guess()
end
```

游戏演示

- 客户端

> telnet 127.0.0.1 8888

- 服务端

> 启动redis
>
> 启动skynet

- 如何优化

> 1. agent 一条连接 如果伪造多条连接 消耗大量的内存；
> 2. 创建gate服务：登陆验证 流程验证 心跳检测 验证成功之后 再创建一个agent；还进一步优化；
> 3. 如果agent（lua虚拟机）功能比较简单，那么可以创建固定数量的agent
> 4. 如果room（lua虚拟机）功能比较简单，那么可以创建固定数量的room
> 5. 如果是万人同时在线，agent room 需要提前创建 长时间运行 gc 会让内存膨胀
> 6. 定时重启服务，创建同样数量的 agent服务组 后面进来的玩家 分配到新的服务组 原来旧的服务组 就慢慢淘汰

关于零声学院

- 构建**完整的后端开发**的知识体系，以此提升自身竞争力；
    - 框架、工具、中间件以及解决方案
    - 理论知识
    - 软实力
        - 安全能力
        - 代码能力
        - 工程素养
        - 架构能力
        - 运营能力
- 你可以自己学习，但也可考虑跟着我们的步伐来学习；

    > 零声学院第9代 linux C/C++ 后台架构开发成长体系

    https://www.0voice.com/uiwebsite/html/courses/v9.2.html
- 课程地址

    https://ke.qq.com/course/420945?tuin=97ed1c8a
- 不看广告，看疗效；

    https://www.yuque.com/lingshengxueyuan/0voice/rw8cgz
- 腾讯知识体系分享