Actor简介&实战

张波
bo.zhang86@gmail.com

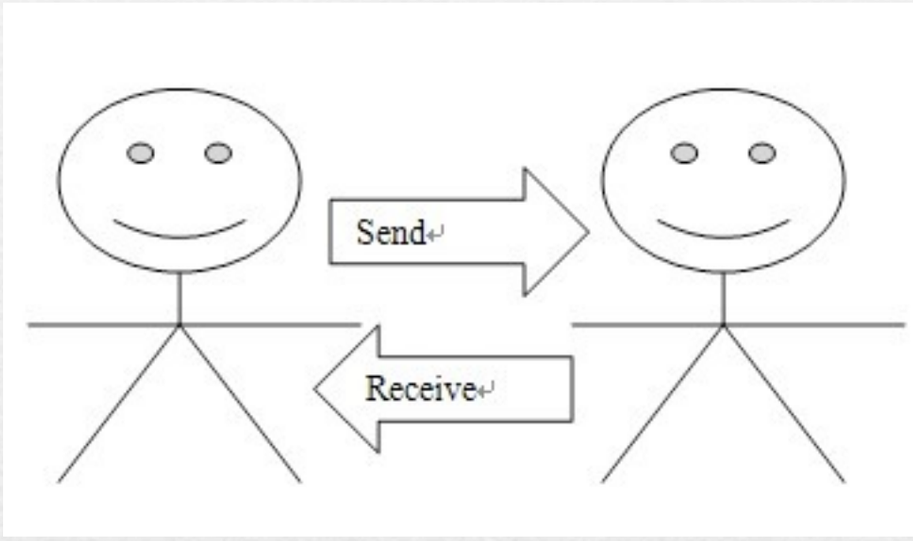
Actor是什么

- The **Actor model** in computer science is a mathematical model of concurrent computation that treats "actors" as the universal primitives of concurrent digital computation: in response to a message that it receives, an actor can make local decisions, create more actors, send more messages, and determine how to respond to the next message received

Actor是什么

- Actor模型在并发编程中是比较常见的一种模型。很多开发语言都提供了原生的Actor模型。例如erlang,scala等

- Actor，可以看作是一个个独立的实体，他们之间是毫无关联的。但是，他们可以通过消息来通信。一个Actor收到其他Actor的信息后，它可以根据需要作出各种相应。消息的类型可以是任意的，消息的内容也可以是任意的。



- 一个Actor如何处理多个Actor的请求呢？它先建立一个消息队列，每次收到消息后，就放入队列，而它每次也从队列中取出消息体来处理。通常我们都使得这个过程是循环的。让Actor可以时刻处理发送来的消息。

- Actor在很多语言实现中（scala，erlang等），都以轻量级进程的形式存在，简单的说，一个actor就是一个执行上下文，绑定了actor的相关数据，实际运行时被调度到真正vm线程中运行。

Actor是什么

- Erlang 例子
  - Pid = spawn(Mod,func,Args) %起一个进程
  - func()->
    receive
    {From,Msg}-> %收到一个消息
    %% do something
  - Pid ! {From,Msg}
- Scala 例子
  - val pong = new Pong
  - class Pong extends Actor {
    def act() {
    case Ping // loop & react
    // do something
  - pong ! Ping
- Akka 例子
  - val helloActor = system.actorOf(Props[HelloActor])
  - class HelloActor extends Actor {
    def receive = {
    case “hello” =>
    // do something
  - helloActor ! “hello”

Actor特性

- 轻量级进程（非操作系统线程），占用资源非常少，创建速度快
  - erlang: 309字，其中233为堆空间，耗时1~3微秒
  - akka: 1GB 2.5million actors，50million msg/sec
- 分布式计算抽象，把任务切分到细粒度，并发完成任务
- 沟通基于消息，避免锁的问题
- 进程完全隔离
- ……

Actor 组件

- actor (msg处理，容错)
- 调度器 (Scheduler/Dispatcher)
- 执行线程
- 邮箱 (Mailbox)
- 路由 (Routing)

Actor实现

- Scala： 在java基础上增加了一些语法糖
  - Akka： 重写actor实现，替换scala actor，类似erlang OTP
- Erlang： 比java更早，语言级别支持actor
  - Erlang OTP： 支持分布式，容错，并发等的一套框架
- Go： 发展迅猛，目前以协程模拟actor
- Clojure： 跑在jvm上，提倡函数式编程

Akka

- 基于 Scala
- actor library & framework
- 很容易编写支持并发和分布式的代码
- 性能非常棒（重写Scala actor和调度器，以及所有操作基于事件）
- 监控树体系
- 扩展不错，并有不少扩展包
- 去中心化架构
  - 一般通过配置来完成负载，路由，远程actor访问等，类似于注册表
  - erlang也基本这样，封装更全，网络程序通常基于dns

Akka

- Actor 例子

```
case class Greeting(who: String)

class GreetingActor extends Actor with ActorLogging {
  def receive = {
    case Greeting(who) ⇒ log.info("Hello " + who)
  }
}

val system = ActorSystem("MySystem")
val greeter = system.actorOf(Props[GreetingActor], name = "greeter")
greeter ! Greeting("Charlie Parker")
```

```
// config on all machines
akka {
 actor {
   provider = akka.remote.RemoteActorRefProvider
   deployment {
     /greeter {
       remote = akka.tcp://MySystem@machine1:2552
     }
   }
 }
}

// ----------------------------
// define the greeting actor and the greeting message
case class Greeting(who: String) extends Serializable

class GreetingActor extends Actor with ActorLogging {
  def receive = {
    case Greeting(who) ⇒ log.info("Hello " + who)
  }
}

// ----------------------------
// on machine 1: empty system, target for deployment from machine 2
val system = ActorSystem("MySystem")

// ----------------------------
// on machine 2: Remote Deployment - deploying on machine1
val system = ActorSystem("MySystem")
val greeter = system.actorOf(Props[GreetingActor], name = "greeter")

// ----------------------------
// on machine 3: Remote Lookup (logical home of “greeter” is machine2, remote deployment is transparent)
val system = ActorSystem("MySystem")
val greeter = system.actorSelection("akka.tcp://MySystem@machine2:2552/user/greeter")
greeter ! Greeting("Sonny Rollins")
```

Akka

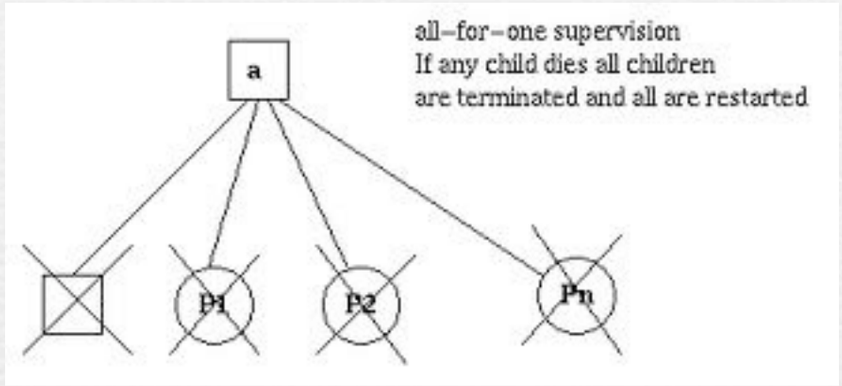
- Supervisor 例子
  - Akka中每个parent actor都是子actor的supervisor

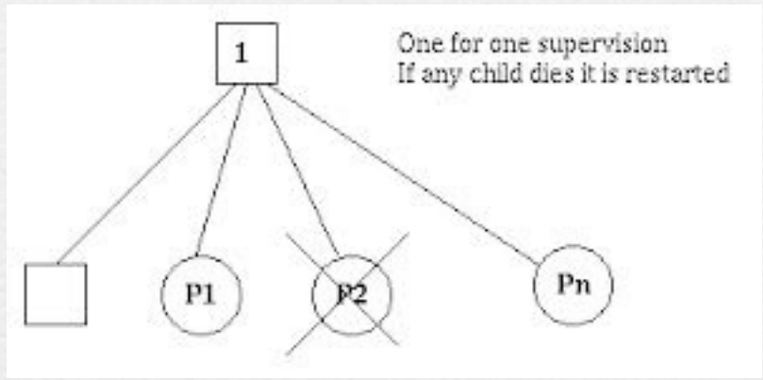
```
class Supervisor extends Actor {
  override val supervisorStrategy =
    OneForOneStrategy(maxNrOfRetries = 10, withinTimeRange = 1 minute) {
      case _: ArithmeticException      ⇒ Resume
      case _: NullPointerException     ⇒ Restart
      case _: Exception                ⇒ Escalate
    }

  val worker = context.actorOf(Props[Worker])

  def receive = {
    case n: Int => worker forward n
  }
}
```

应用

- Scala 应用案例：
  - Linked in
  - Facebook
  - twitter
- 国外比较火热，国内使用较少

专题： 监控树

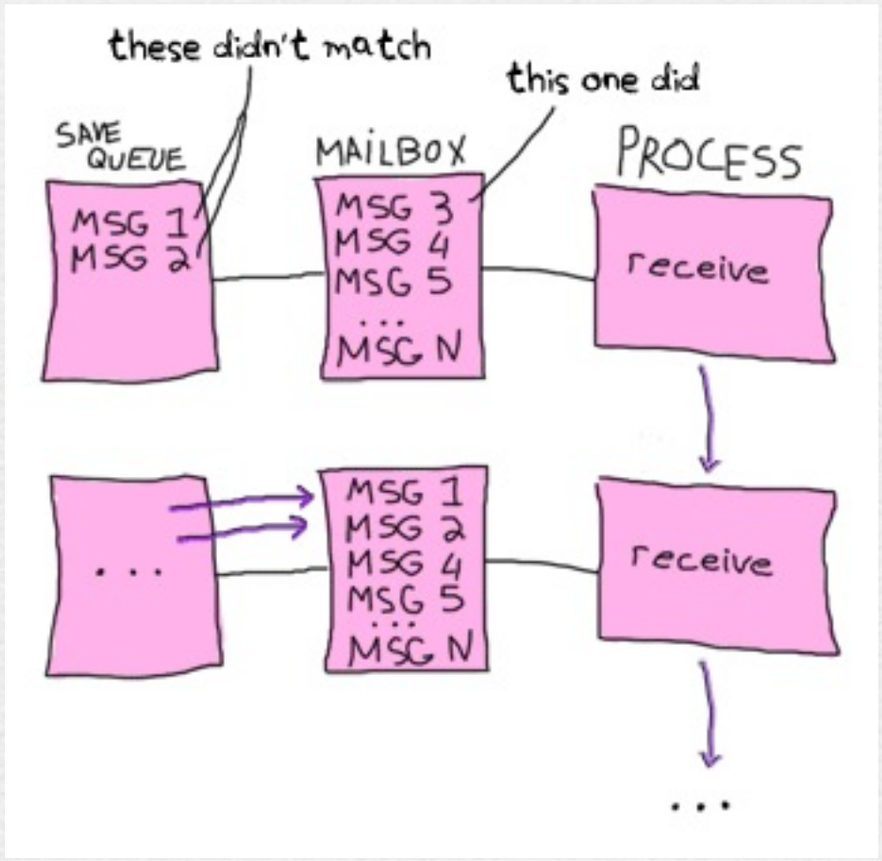




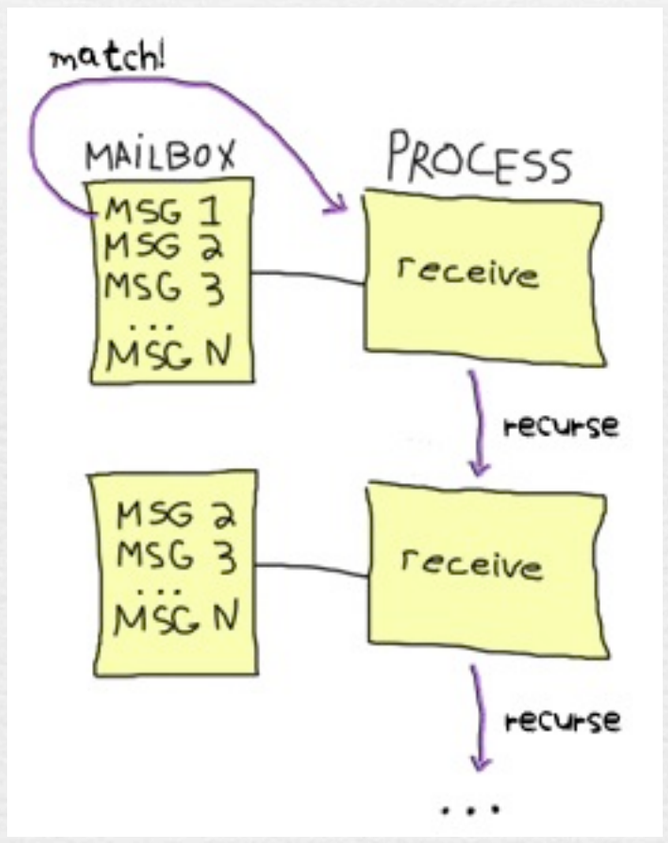
- 相关actor，以树形结构组织，supervisor监控actor，当发生“异常”时，supervisor对事件进行处理
- supervisor有对事件处理的策略，常用有all for one & one for one
- supervisor可以有子supervisor
- 容错
- 分布式

专题：邮箱消息处理

- Erlang & Scala



- Akka



if ! msg match

failure(默认会catch，记录到日志中)

扩展篇： erlang

- 函数式语言
- 原生支持actor模型
- 原生支持分布式
- 容错性好
- hot code swap
- real scheduler（用户进程调度，基于时间片）
- 历史很悠久，系统很稳定
- riak & rabbitmq & ejabberd 很棒
- web framework : yaws, mochiweb, cowboy, CouchDB

erlang otp

- supervisor
- behaviour(通用服务器，fsm，event handler)
- event & log
- 标准化应用结构
- hot code swap
- mnesia

erlang 案例

- 游戏行业（页游中erlang如日中天）
- 广告行业
  - 互联网 & 移动互联网 广告平台商
- 互联网行业
  - taobao
- 非常成功
  - whatsapp (2012年已超过200w tcp连接)

Thanks !