NJSD
中国（南京）软件开发者大会
China（Nanjing）Software Developers Conference
2016

Akka与微服务实践

1号店-搜索与精准化部架构师 王富平

SOA治理

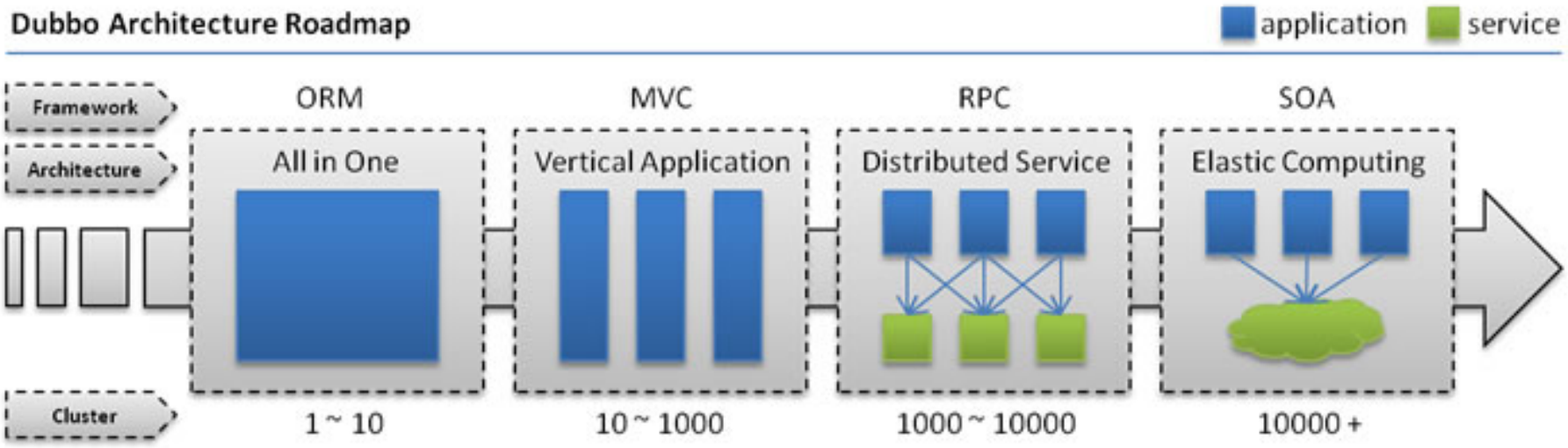
- 微服务的官感：独立、开放。

  1、契合人类的历史：部落-族群（各种歧视）-家庭-人

  2、契合人的本性：开放促进了融合、消弭误解

SOA演变历史

图片引用自dubbo官网



SOA演变历史

- 单一应用架构
  - 当网站流量很小时，只需一个应用，将所有功能都部署在一起，以减少部署节点和成本。
  - 此时，用于简化增删改查工作量的 **数据访问框架(ORM)** 是关键。
- 垂直应用架构
  - 当访问量逐渐增大，单一应用增加机器带来的加速度越来越小，将应用拆成互不相干的几个应用，以提升效率。
  - 此时，用于加速前端页面开发的 **Web框架(MVC)** 是关键。
- 分布式服务架构
  - 当垂直应用越来越多，应用之间交互不可避免，将核心业务抽取出来，作为独立的服务，逐渐形成稳定的服务中心，使前端应用能更快速的响应多变的市场需求。
  - 此时，用于提高业务复用及整合的 **分布式服务框架(RPC)** 是关键。
- 流动计算架构
  - 当服务越来越多，容量的评估，小服务资源的浪费等问题逐渐显现，此时需增加一个调度中心基于访问压力实时管理集群容量，提高集群利用率。
  - 此时，用于提高机器利用率的 **资源调度和治理中心(SOA)** 是关键。

需要给微服务一个定义

- 服务化　!= 微服务
- 微服务的重点

　　1、微型化：进一步分解传统领域服务，粒度更小

　　2、标准化：统一部署、管理

　　3、平台化：在标准化前提下，接入统一管理平台，提供监控、负载均衡、动态扩展等功能

Akka是什么鬼？

- 可扩展的实时事务处理框架

  1. 并发模型： 基于actor模型，对并发、并行、事务处理做了高级抽象。

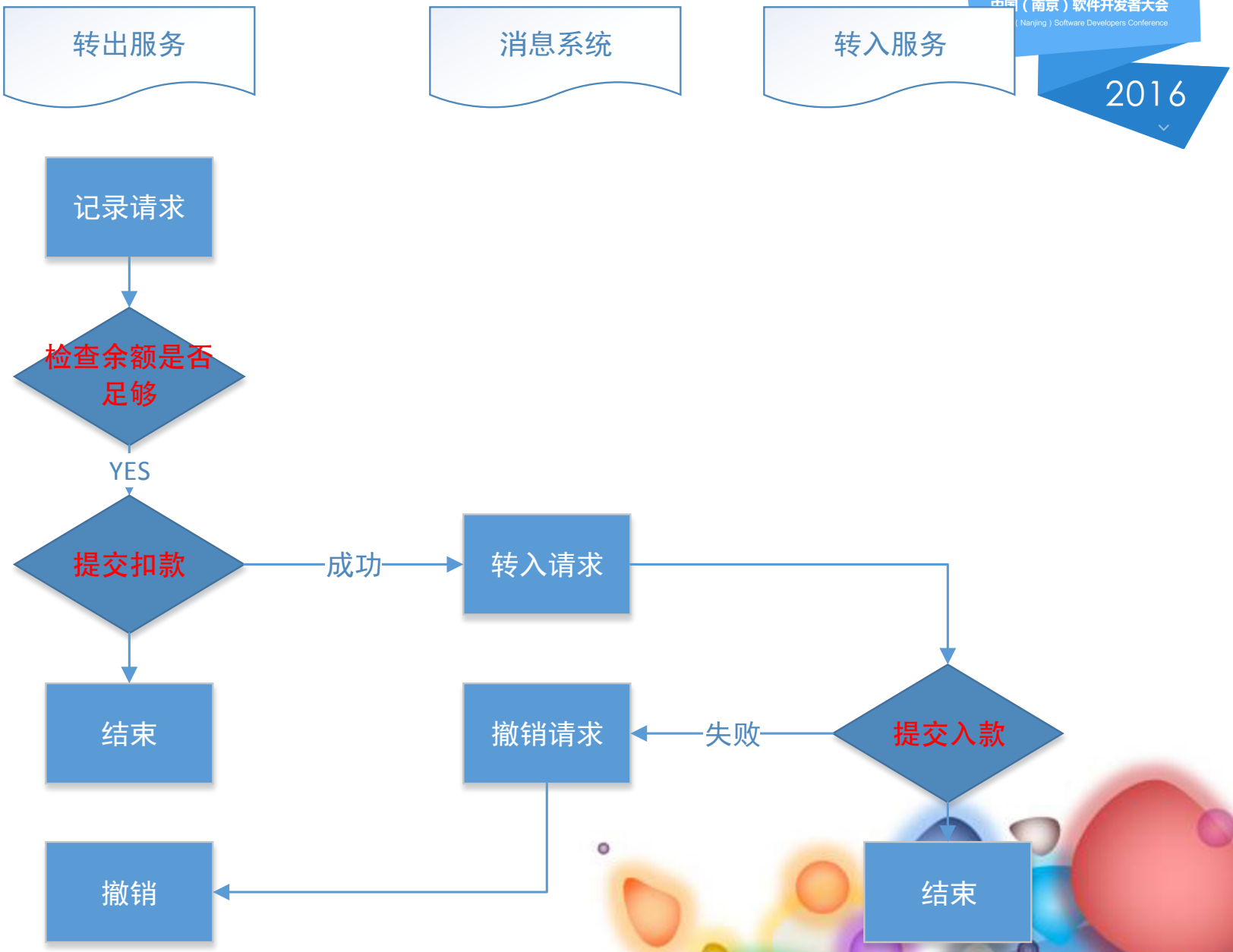
  2. 高性能、容错： actor提供了异步非阻塞的事件驱动编程模型, 提供了明确的错误处理机制

  3. 可扩展：

  3.1. cluster模式下，可以动态增减处理节点

  3.2. 自动监控节点，实现负载均衡

举个栗子：A转账给B

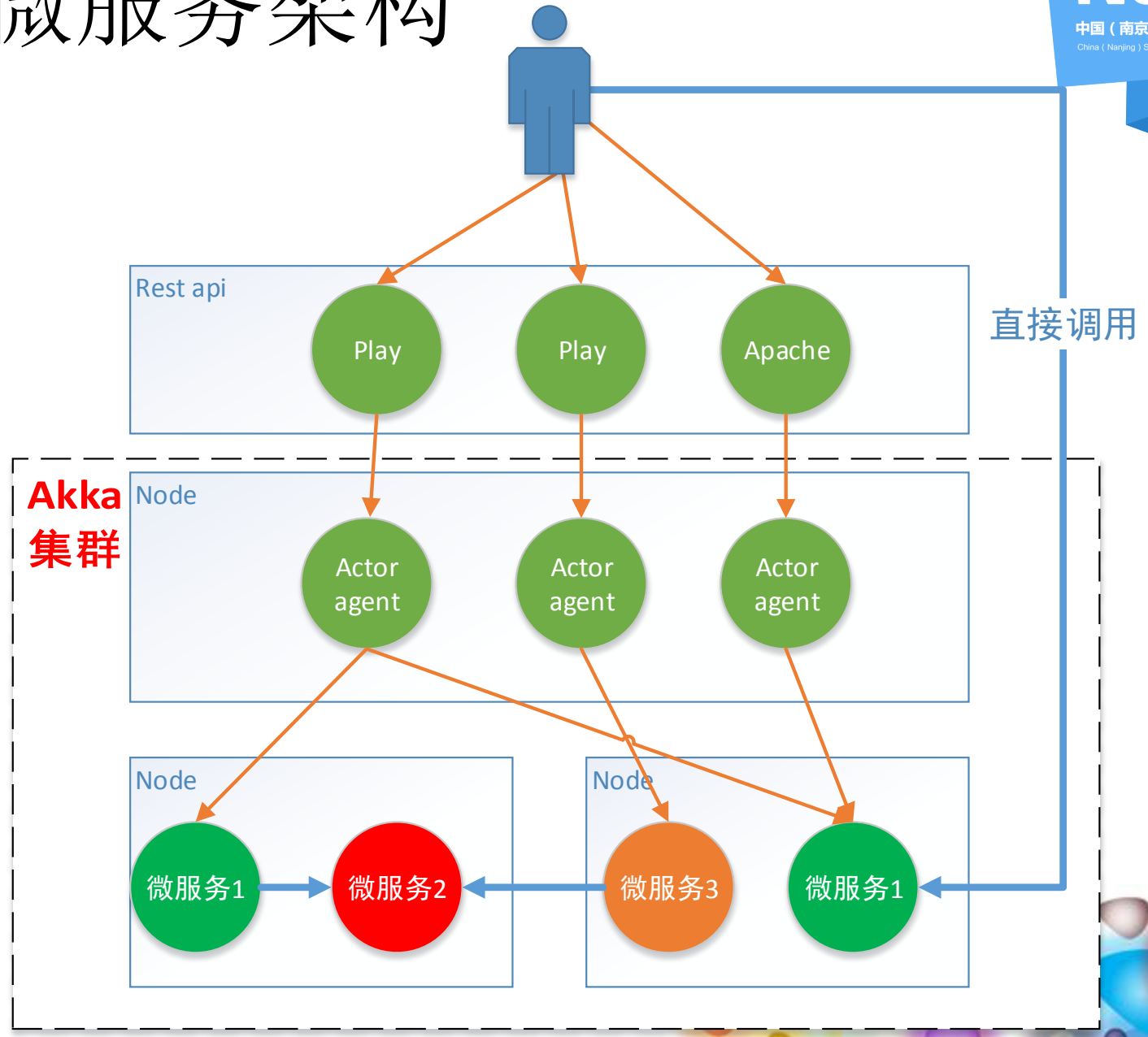


Akka实现


```
//转账请求
case class TransferMoneyRequest(txid: String, fromUserId: Int, money: Double, toUserId: Int)
//转入请求
case class AddMoneyRequest(txid: String, fromUserId: Int, money: Double, toUserId: Int)
//取消转账请求
case class CancleTransferMoneyRequest(txid: String, fromUserId: Int, money: Double, toUserId: Int)

class UserStat(val userid: Int, var account: Double)
abstract class AccountService extends Actor{
    val router = context.actorSelection("/user/router")
    def getUserStat(userid: Int): UserStat
    def addMoney(userStat: UserStat, money: Double): Boolean
    def lockUserid(userid: Int): Boolean
    def unlockUserid(userid: Int): Boolean
    def recieve = {
        case req:TransferMoneyRequest if req.money > 0 => ...
        case req:AddMoneyRequest => ...
        case cancle: CancleTransferMoneyRequest => {}
        case _ => {}
    }
}
```

Akka微服务架构



从Actor开始

- Akka是actor模型的scala语言实现
- Actor是akka提供的一个轻量级线程：百万级别消耗1G左右内存
- Actor是一个有状态的轻量级线程

  1、只有actor才能向actor发送消息

  2、每个actor有一个mailbox来存储消息

  3、重载actor的receive函数实现处理逻辑

  4、actor不会并行处理数据，即一个actor是顺序处理消息的

  5、每隔actor都有一个dispacher，来帮助actor发送消息

正确打开actor的方式

- Akka可以运行在多个模式：本地模式、集群模式

  1、本地模式下，actor之间传递消息直接使用对象引用

  2、集群模式下，同一jvm上的actor传递消息也会直接使用对象引用

- 正确使用actor

  1、actor消息不可变：可变的消息在会造成数据共享问题

  2、保持actor状态封闭：actor的状态可变只由消息驱动，而不是调用actor的某个方法

Actor =？　微服务

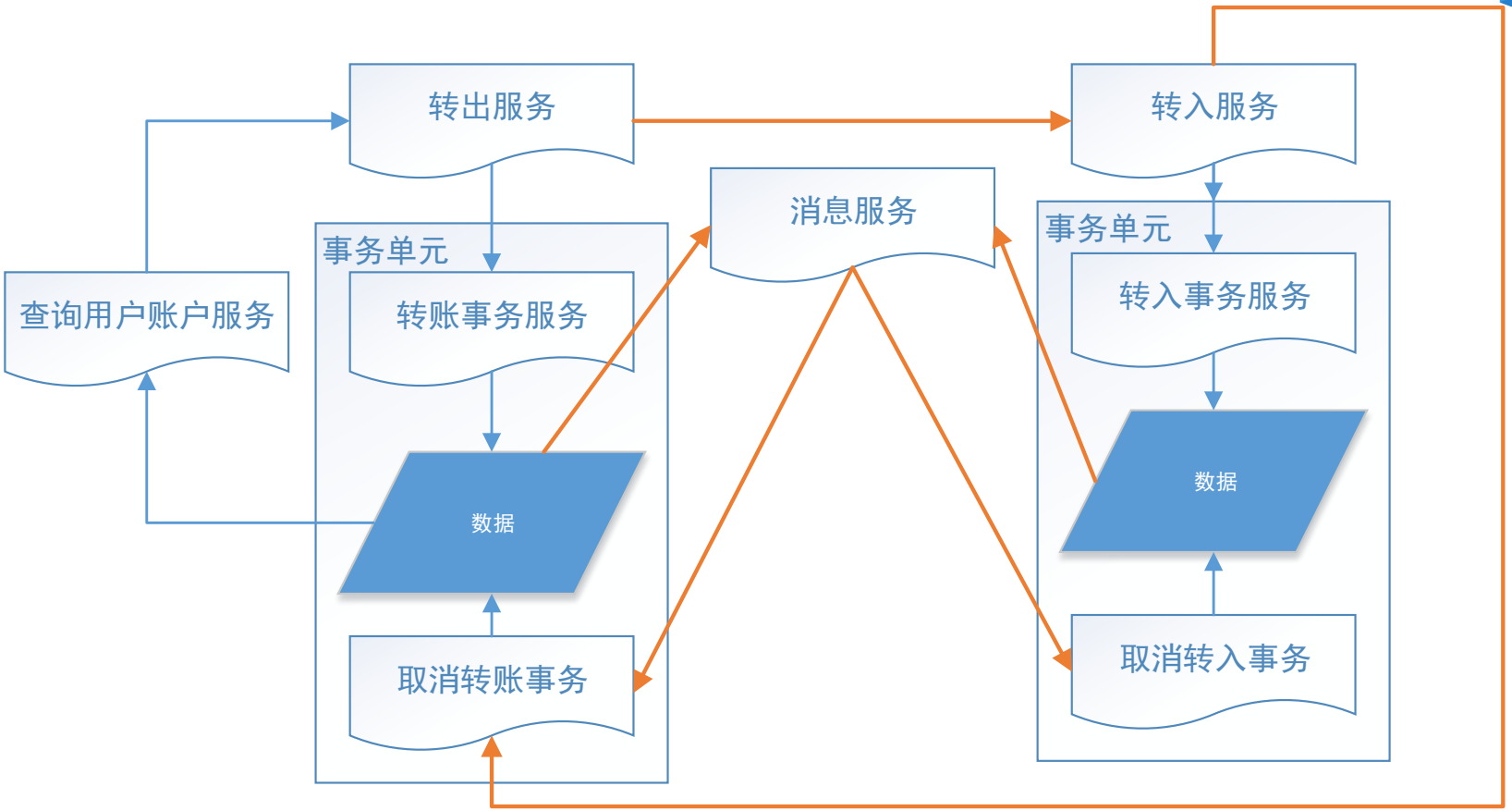
- Actor的是不是粒度太小了？

  重点在于actor是一个正确的封装：独立的服务单元

- Actor的接口是什么？

  1. 可以通过apache、play框架封装rest接口
  2. 直接使用actor接口，直接发送消息

分解转账服务



微服务要解决的问题

- 模块解耦：细粒度的划分模块，理清依赖关系。
- 服务重用：提取公共服务
- 缩短构建流程：

1. 微服务减小了模块边界，bug修复、版本发布，更易测试、上线。

2. 再也不需要一个超级发布包，再也不用害怕错综复杂的依赖关系，会产生什么样的问题

Akka服务管理

1. 服务发现
2. 服务监控
3. 负载均衡

服务发现

1. 使用接口查询Akka集群上线服务
2. 订阅Akka集群事件：服务上线、下线等

服务监控

- Jmx监控管理：*akka.<protocol>://<actor-system-name>@<hostname>:<port>*
  1. 查看当前集群节点、运行状态
  2. 节点管理：增加、下线节点
- Cluster metrics：akka集群节点运行过程中会监控节点运行状态：内存、cpu、处理性能等

负载均衡与fail detector

- **Adaptive Load Balancing**

  根据cluster metrics，进行负载均衡

- **Fail detector**

Akka集群通过心跳方式来检测node是否可用

1. 每隔节点都有fail detector
2. 一个fail detecor发现节点A不可用，则进行广播告知不可用，节点A设置为unreadable
3. 当所有节点都发现节点A可用后，设置A为reachable
4. 超过一定阈值后，A还未回复，则下线

Akka集群通信协议

为什么通信协议重要？

1. 选取leader

2. 广播消息

Akka使用的是gossip协议，该协议最大的特点：leader随时可以变，可以是任何节点

自动化部署

- Docker

  容器化akka

  结合kubernetes进行容器管理

- Mesos

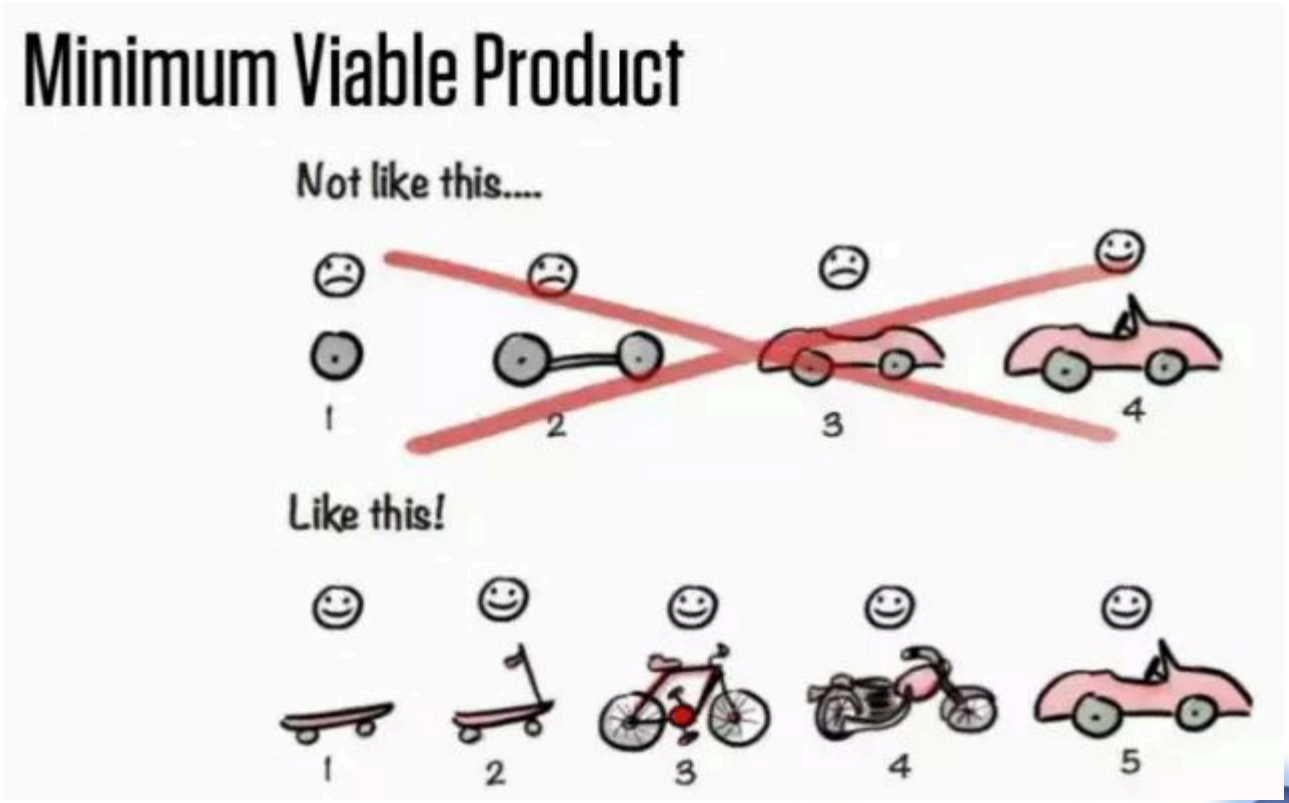
  自动化管理、部署akka集群

为什么是Akka？

	Dubbo	Akka
服务注册	√ 依赖zookeeper	√ 无需借助外部组件
服务监控	√	√
负载均衡	√	√
接口统一	不统一	√ 1、接口用rest还是thrift，akka说直接传对象不行吗？ 2、protobuf还是kyro做序列化，akka说其实我已经提供解决方案

一次编写，无限扩展

其实人家只是开发比较快



NJSD
中国（南京）软件开发者大会
China（Nanjing）Software Developers Conference
2016

回到南京很开心
谢谢大家！