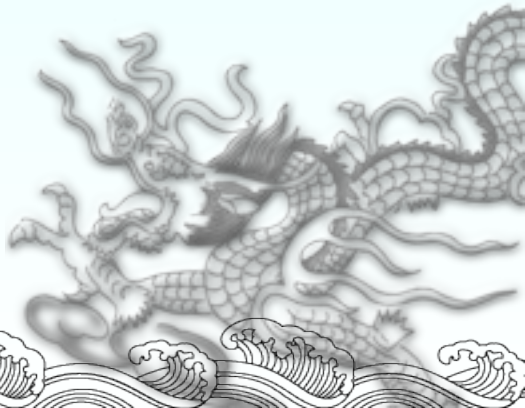
基于 Erlang 的 MMO 链接管理服务器

侯明园

御风行数码科技有限公司

目录

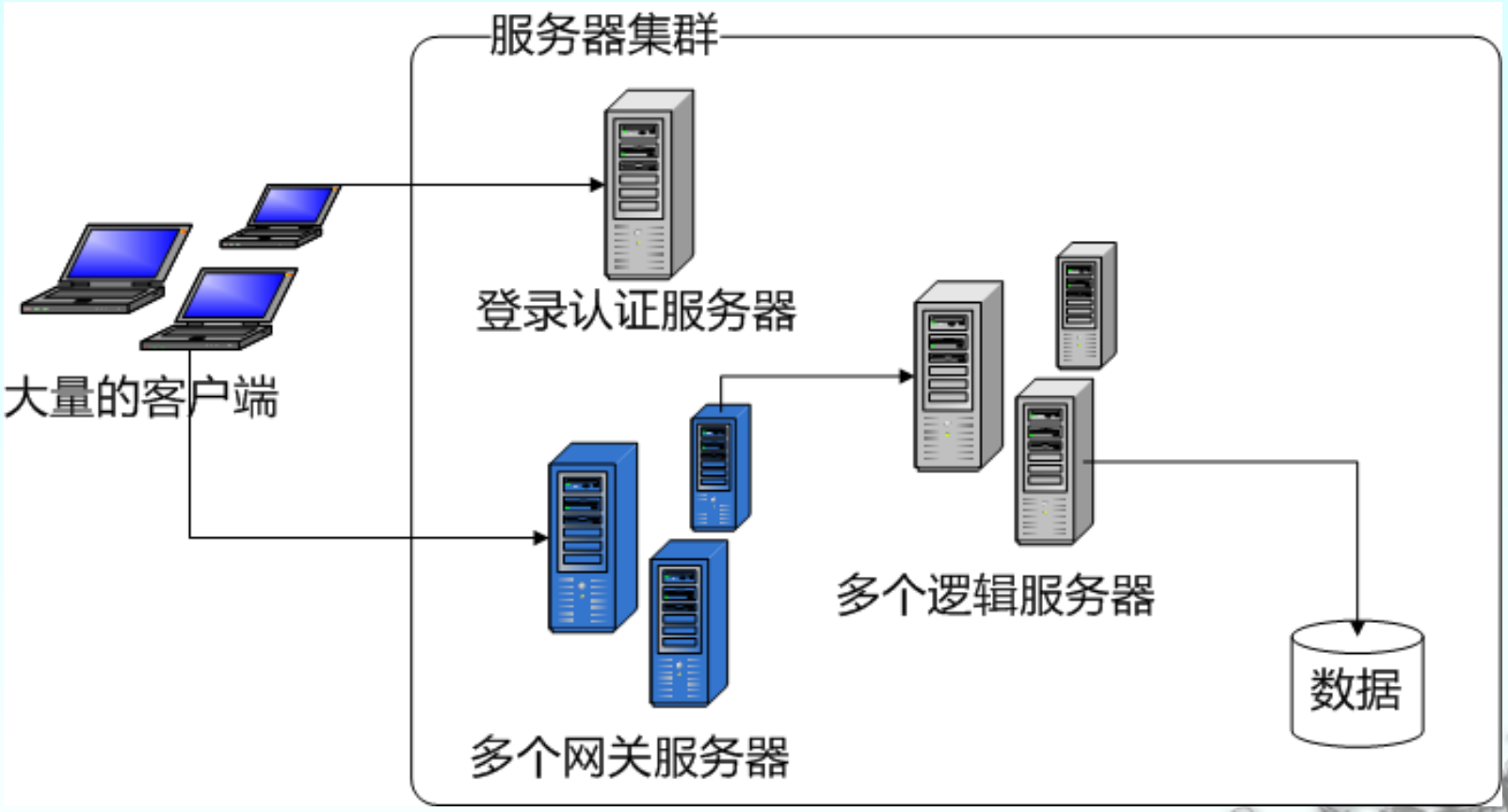
- 问题
  - MMO 链接管理的特点与需求
- 整合
  - Erlang 如何整合入到现有集群系统？
- 测量
  - 建立基于 erlang 的压力测试环境
- 实现
  - Erlang/OTP 如何简化编程？
- 展望
  - erlang 在网络游戏开发中的潜力点



问题

MMO 链接管理的特点

MMO 集群结构图



MMO 链接管理的特点

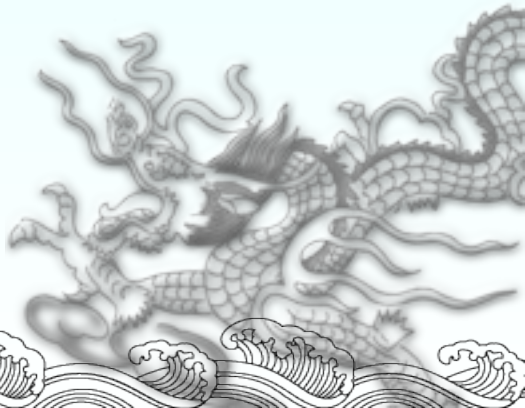
- 链接管理的应用场景
  - MMORPG 游戏服务端结构中包含是那个部分：登录认证服务，链接管理服务，游戏逻辑服务。游戏逻辑服务还可以继续进行细分。
  - 通常只有链接服务与客户端直接进行通信，可以认为集群内是安全的可以预测的计算环境，外部是实际的不安全的计算环境，这个服务是逻辑服务器与客户端之间的桥梁和管道。
  - 逻辑服务切换时，用户拥有一个不变的访问点

MMO 链接管理的特点（续）

- 管理集群中所有的来自 client 的链接，将从客户端收到的消息转发到对应的逻辑服务器， 将逻辑服务器收到的消息转发到相应的 client， 发出去的包大部分的包是广播，链接服务器负责处理广播。
  - 路由
  - 广播
  - 安全
- 功能明确，逻辑比较简单

MMO 链接管理的需求

- 高性能
  - 路由和广播的要尽可能快，保障游戏的流畅感觉
  - 服务端频率 10HZ，响应时间 <100ms
- 高并发
  - 连接多，能处理的链接越高越好， MMO 服务器承载的人数对游戏有重要的意义
  - 大于 5000+
- 高吞吐量
  - 输出远远大于输入
    - 收到的数量少，需要广播的数据多
- 高可用性
  - 出故障时间要少
  - 恢复时间要短
  - 95%



整合

ERLANG 如何整合入到现有集群系统？

整合

- 整合进入原有的监控系统
  - 兼容原有管理工具
  - 状态报告
  - 维护操作，重启，关闭，查询
- 数据库访问　mysql 数据库支持
- 兼容原有网络协议
  - 原有协议是简单二进制协议
  - 引入协议定义语言，自动生成 actionscript， python， C++, erlang 协议编码解码代码

关于整合的归纳与总结

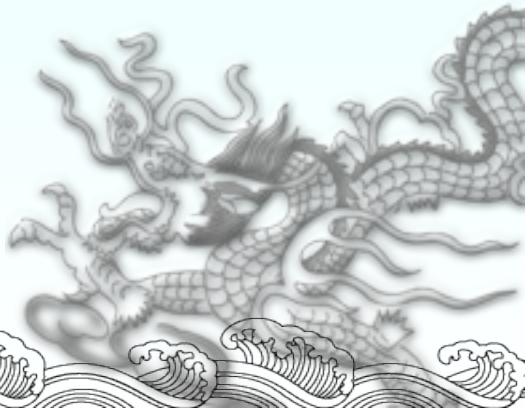
- 定义通信标准
  - 如果公司的应用程序想要扎根网络应用领域，及时你是第一款产品，建立的一个网络协议的规范和标准，定义语言，也是非常有必要的。
  - Protobuf， asn， CERL SDL
  - 未来可以进一步建立一种行业标准

测量

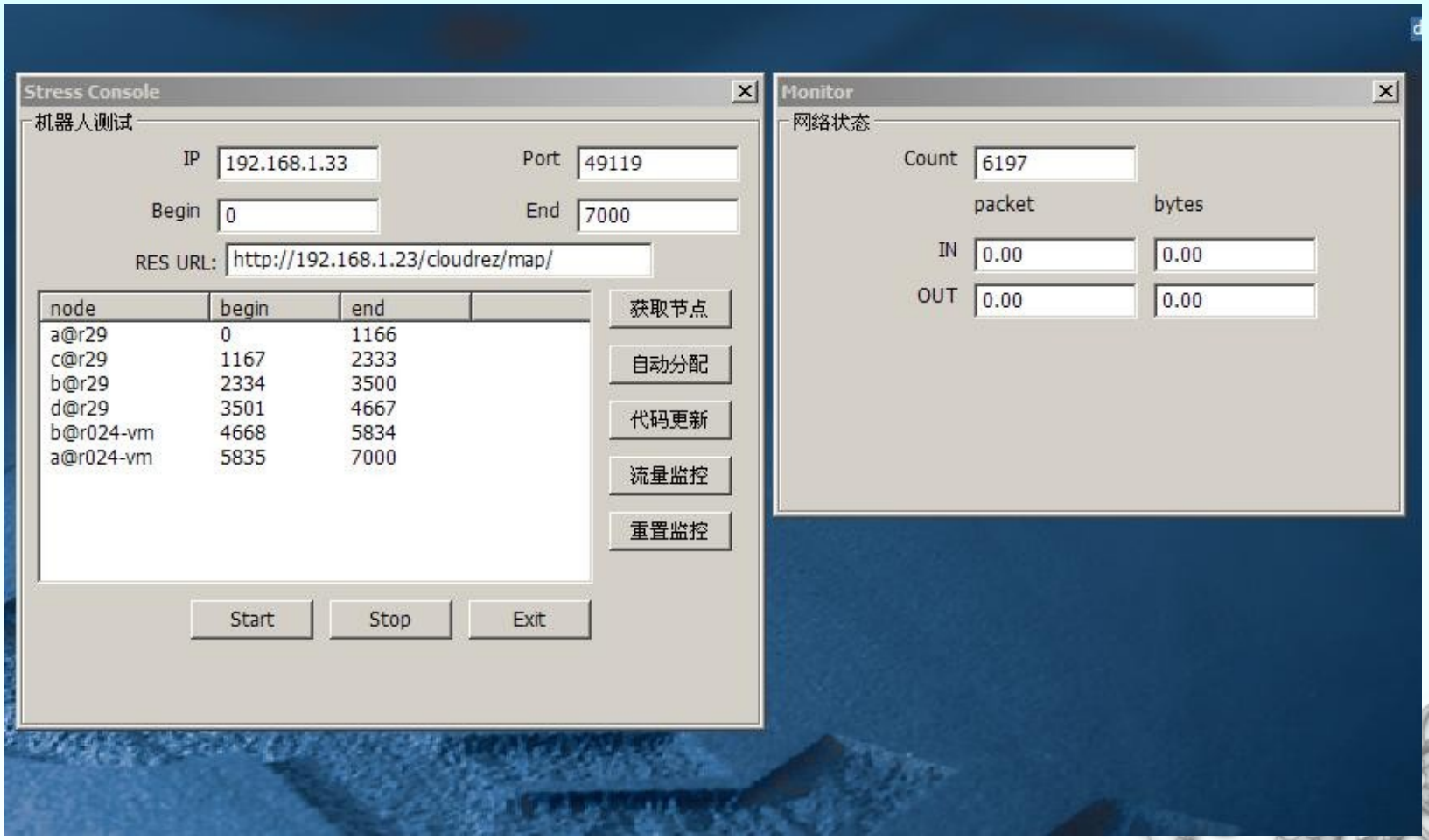
压力测试环境的建立

Erlang 之前

- C++&python Client
- 缺点
  - 远程连接 10+ 机器
  - 管理困难
  - 状态监测困难
  - 代码发布困难

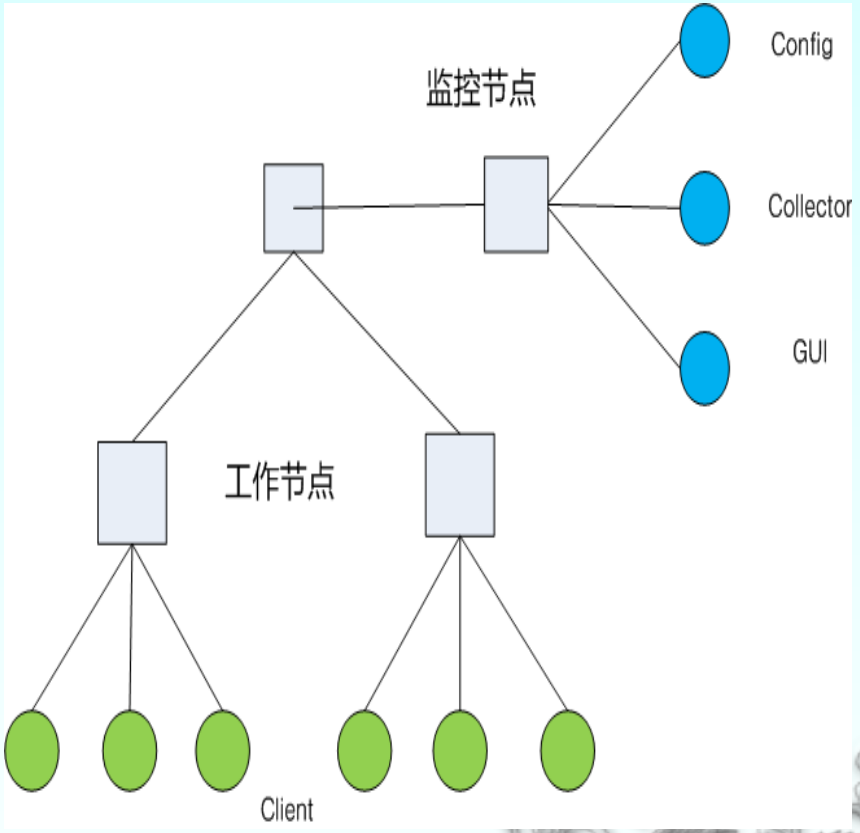


Power by Erlang/OTP



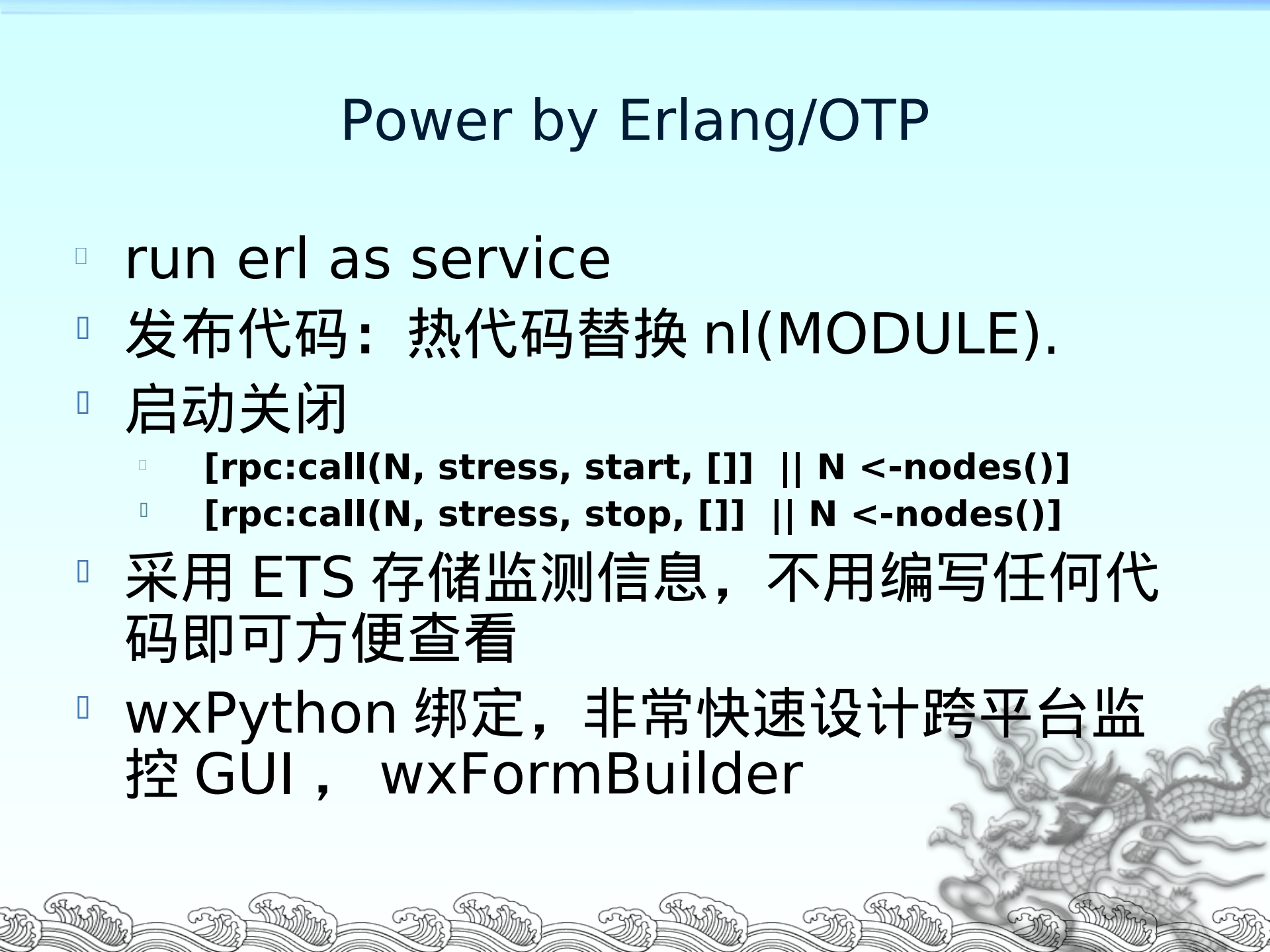
Power by Erlang/OTP

- 监控节点（唯一）
  - 测试配置状态进程（ global)
  - 测试数据收集进程(global)
  - GUI 控制进程
- 工作节点（多个）
  - 监督进程
  - 多个工作进程



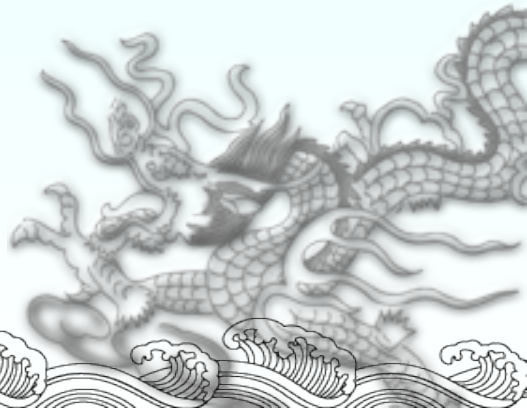
Power by Erlang/OTP

- run erl as service
- 发布代码：热代码替换 nl(MODULE).
- 启动关闭
  - **[rpc:call(N, stress, start, []) || N <-nodes()]**
  - **[rpc:call(N, stress, stop, []) || N <-nodes()]**
- 采用 ETS 存储监测信息，不用编写任何代码即可方便查看
- wxPython 绑定，非常快速设计跨平台监控 GUI， wxFormBuilder



Power by Erlang/OTP

- 代码规模
  - 总共 erlang 代码：2200
  - 注释： 600
- 所有的进程都是用 behavior,:
  - gen_server
  - gen_fsm
  - gen_supervisor
  - wx_object



关于测量的归纳与总结

- 测试先行
- Erlang 非常适合搭建分布式测试系统
  - 开发快速
  - 平台丰富
  - 工具完善
  - 管理简单

实现

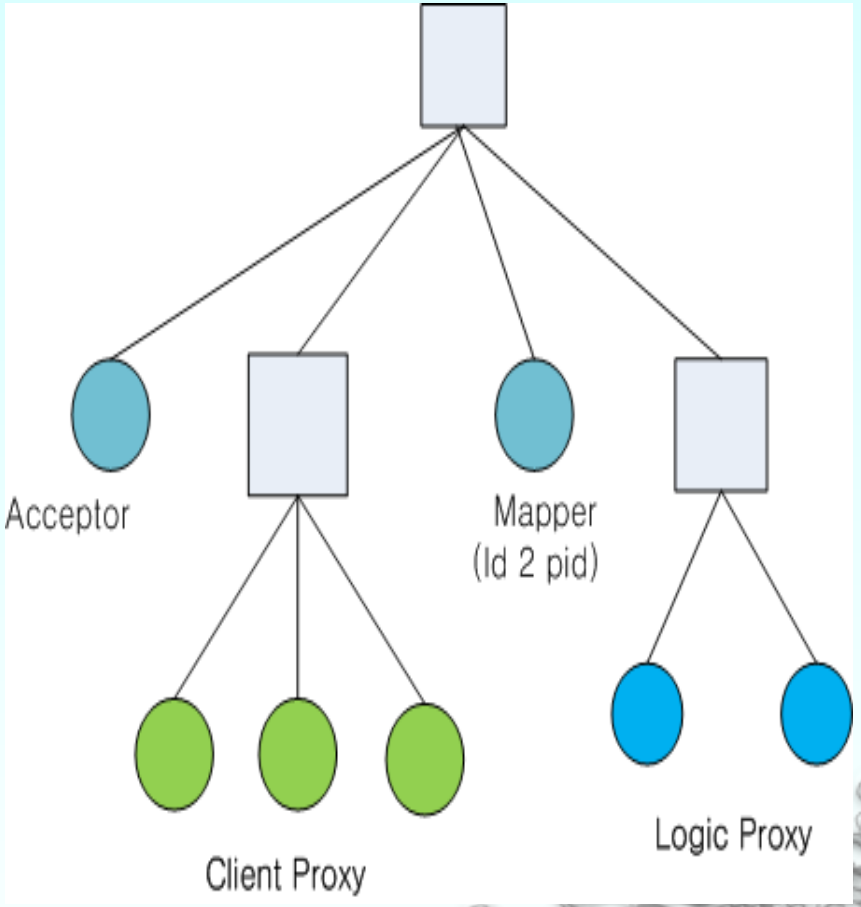
ERLANG 如何简化编程？

原有 C++ 版本

- 非阻塞 socket
- 状态机 (20 个状态， 26 个事件)
- 同一个集群中可以启动多个连接管理进程

实现 V1

- 监听进程（ 1 个）
- 客户端链接管理
  - 所有 Client 进程由一个监督进程管理
  - 一个进程对应一个进程对应一个客户端 TCP 链接
  - 一个进程对应一个到逻辑服务器的链接
  - 进程实现采用 OTP 状态机
- 服务端链接管理
  - 所有 Service 进程由一个监督进程管理
- 链接 ID 到 PID 映射管理



简单的实现比较

C++

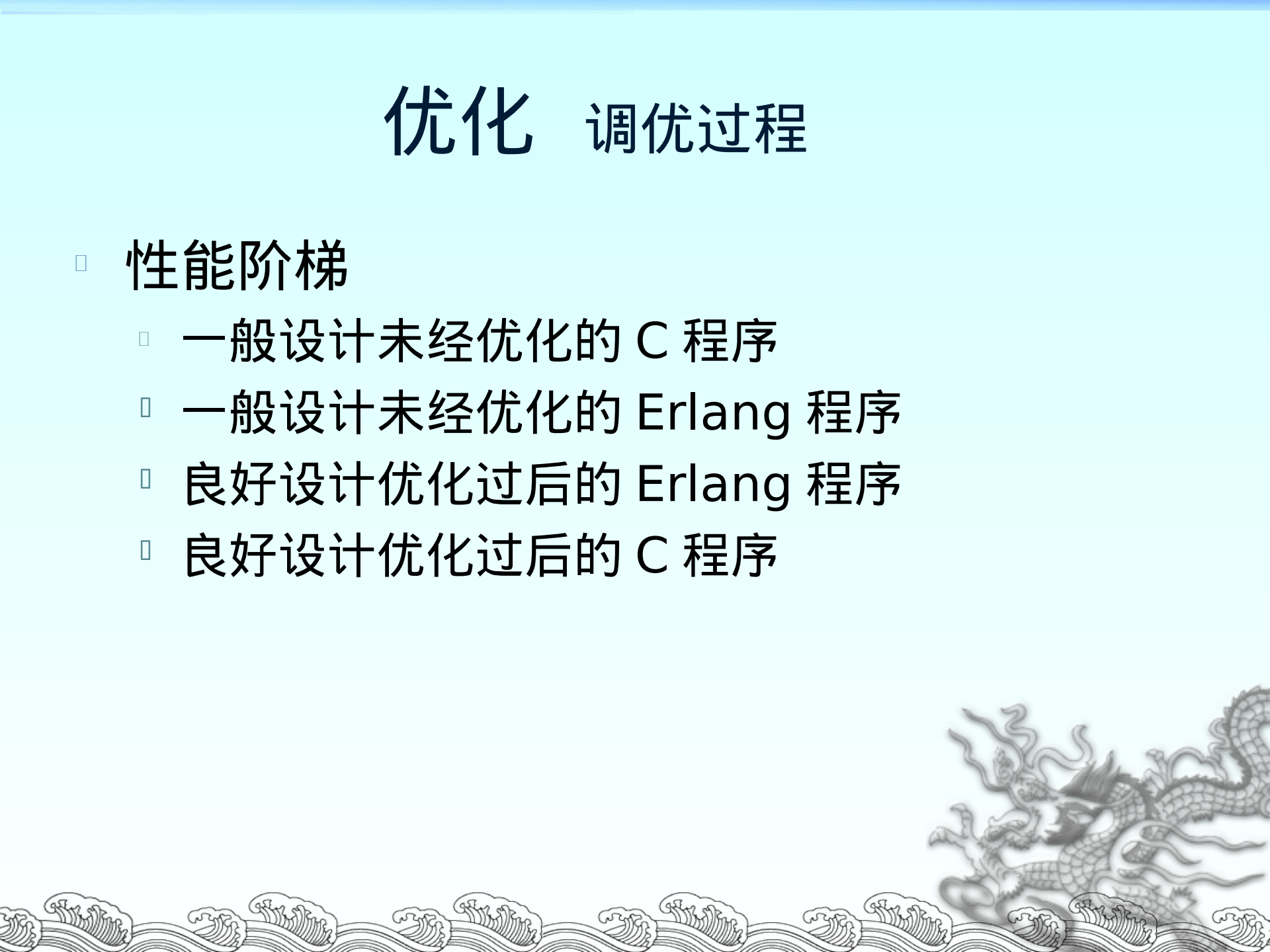
- 代码行数 :7924
- 逻辑结构方面比较
  - 单进程
  - 状态机
  - （ 20 状态， 26 事件 ）
- 性能
  - 6000

Erlang

- 代码行数： 2000 线程
- 逻辑结构
  - 表示连接
  - 状态机状态减少了：采用进程来模拟客户端之后
  - (10 状态， 12 事件)
  - 模式匹配使代码更加清晰
  - otp 行为模式框架简化程序
- 性能
  - 6000

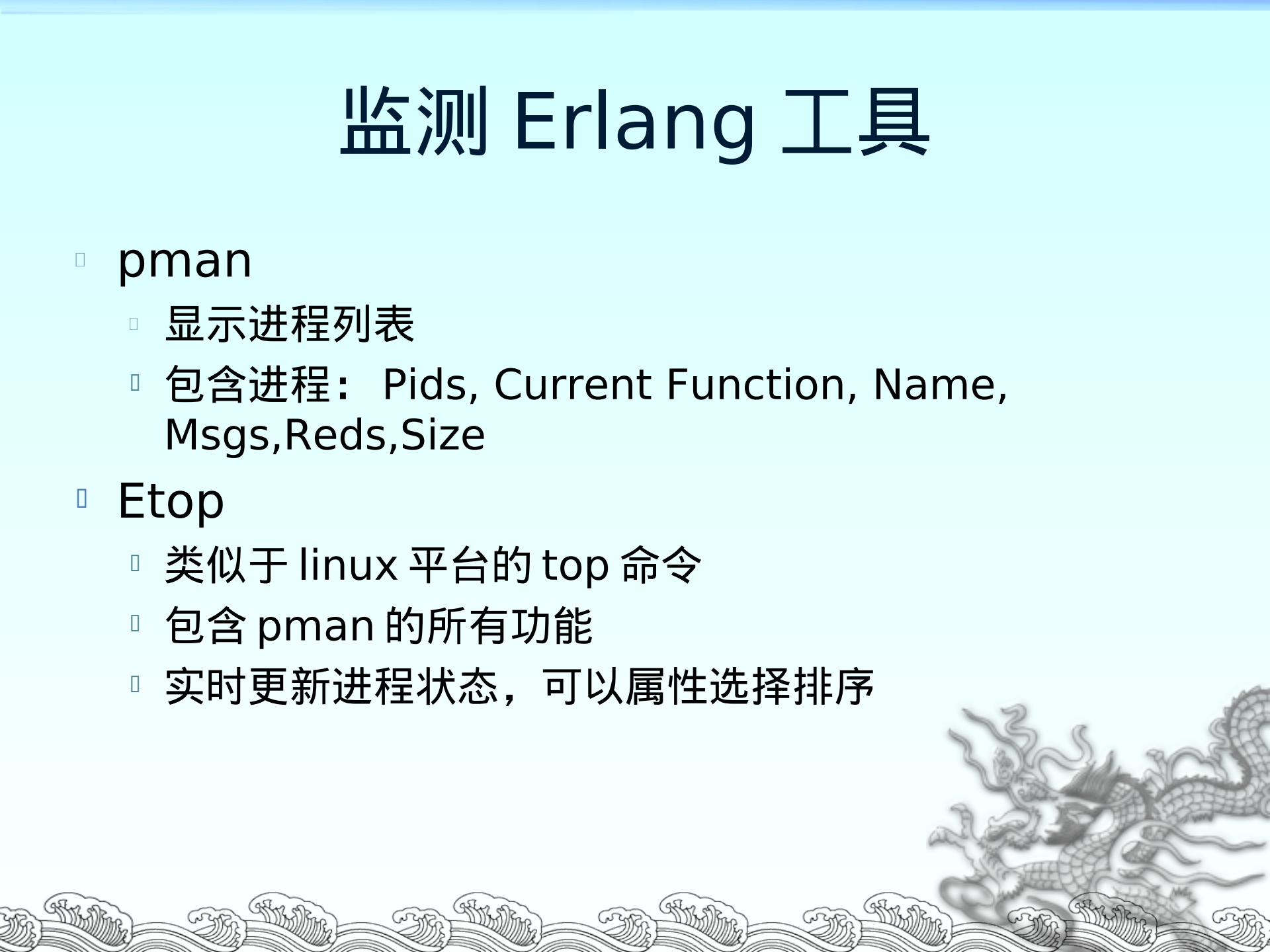
优化 调优过程

- 性能阶梯
  - 一般设计未经优化的 C 程序
  - 一般设计未经优化的 Erlang 程序
  - 良好设计优化过后的 Erlang 程序
  - 良好设计优化过后的 C 程序



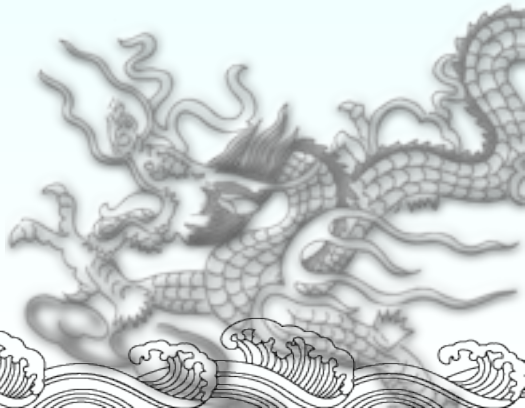
监测 Erlang 工具

- pman
  - 显示进程列表
  - 包含进程： Pids, Current Function, Name, Msgs,Reds,Size
- Etop
  - 类似于 linux 平台的 top 命令
  - 包含 pman 的所有功能
  - 实时更新进程状态，可以属性选择排序



优化 case 1

- 多链接情况下 **+K** 非常重要
  - **4000, CPU: 80%~100%**
    - **Top** 看到 **60-70%sy**
  - **+K true, CPU 13% ~ 20%**
  - **+K true && –smp disable, CPU 9%~20%**
  - 原因：使用 **epoll**



优化 case 2： 数字 ID 查找 PID

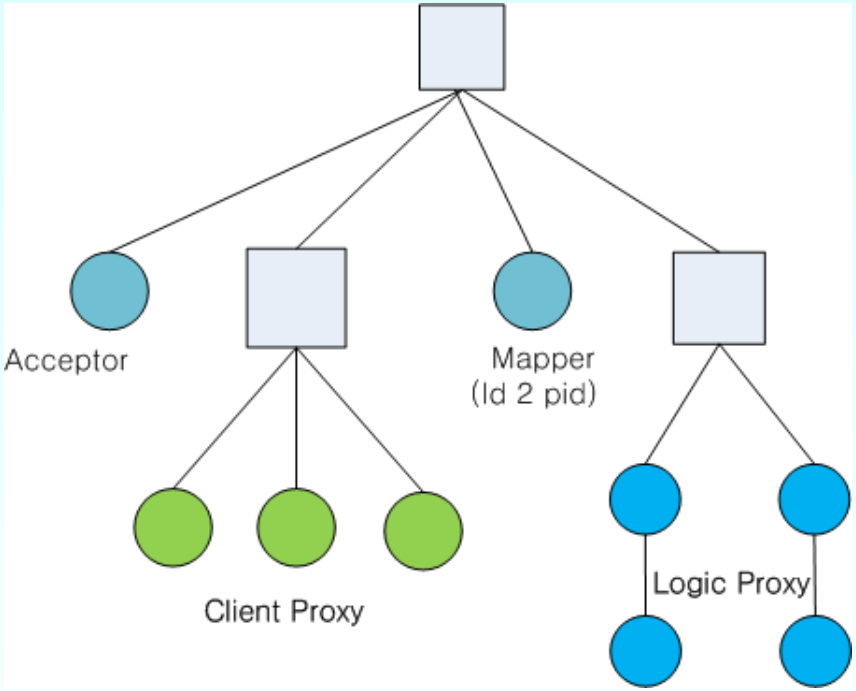
问题

- 每个连接有一个数字 ID，逻辑服务器的广播包里面包含一个数字 ID 类别和数据，网关服务器通过 ID 查找到对应 PID, 这个查找过程一度成为瓶颈

方案性能对比

查找方案	Insert	find
ETS	1.11	8.98
DICT	5.12	9.65
gbtree	1	28.25
Register	80	1

优化 case3



- LoginProxy 负责从客户端到 login 服务器转发，服务端到客户端的转发，这有可能造成性能瓶颈，收到服务端的消息之后，一般会进行循环处理，可能会延迟转发客户端消息，将转发任务分两部分，一个只负责从客户端到 logic 服务器，一个负责从逻辑服务器到 client

结果

- OS: CentOS release 5 (Final)
- Intel(R) Xeon(TM) CPU 2.80GHz X 2
- 6500 链接
- 30000 ~ 40000 packet/s

展望

- erlang 在网络游戏开发中的潜力点
- 热部署
  - 网络游戏服务器维护通常会导致玩家流失如果 Erlang 的热部署能够有效应用那么有可能极大减少服务器维护的频率和时间
- AI
  - 有 AI 实现采用 Lua 协程， Stackless Python 。Erlang 进程也有类似的特点
- 运营接口
- 整合工具