用 Erlang 构建容错系统

- 连城 (rhythm.mail@gmail.com)
- Twitter @liancheng
- 微博 @ 连城 404

Erlang 是一套什么样的系统？

- 涉及语言、库、操作系统多个层次的全方位分布式容错解决方案
- 从高难度工程问题中催生而来
  - 与 C 的历史背景惊人地相似
- 具有显著的函数式特征
  - 但从来都不是一个学院派语言
  - 没有学院派函数式语言中的各种花哨概念

历史

- 起源于 1980 年代 Joe Armstrong 在爱立信计算机科学实验室所做的电信系统研究
  - Smalltalk 太慢
  - Prolog 缺乏并发支持
  - 作为 Prolog 的并发扩展， Erlang 诞生了
- 逐渐发展成为语言、库、操作系统多层次全方位的分布式容错系统解决方案

可怕的电信系统

- 高并发
- 软实时（毫秒级）
- 分布式
- 直接操作硬件
- 大型软件系统
- 功能复杂
- 不间断运行（5个9的SLA）
- 超高质量要求
- 软硬件容错

成功案例

- 早年
  - 爱立信 AXD301 交换机（ loc > 1M ）
  - 爱立信 GPRS 系统
- 当前
  - Facebook IM
  - WhatsApp
  - 云诺
  - 国内页游界

互联网环境下容错系统的特点

- 系统的一部分受损不影响整体功能
- 尝试自行修复受损的部分
- 支持分布式
- 无缝升级


BUU
BY FUBUAZNBOY

如影随形的终极错误修复手段

A problem has been detected and Windows (lol) has been shut down to prevent damage
to your computer.

THE FOLLOWING FILES ARE MISSING/CORRUPT:
REASON_TO_USE_WINDOWS.SYS
LOL_WINDOWS.SYS
LMAO_WINDOWS_VISTA.SYS

THE FOLLOWING FILES HAVE PERFORMED AN ILLEGAL OPERATION:
WINDOWSXP.EXE

This is definitely not the first time you've seen this error screen, but what can you do?
Might as well restart your computer. If this problem persists (it probaly will), follow these steps:

Walk/ride/fly/swim to your nearest Apple Store
Talk to a clerk about the Mac OS X Operating System.
Pay him one-hundred twenty nine (129$) dollars U.S.
Bring the box home, and pop the Compact Disk into your Optical Drive.

Or, if you think Apple sucks, try checking your piece of shit BIOS, to see if you have not already
disabled everything, and fix it. Once again, Windows will provide you with no support. If by chance
you have no idea what the hell you are doing (yea, right) press F8, and just boot into Safe Mode,
even though half the time it is unavailable due to the *incredible reliability* of Windows Operating
Systems.

Technical Information:

*** STOP: 0xUSING00x33 (x00xWIND0Wx5, 0PxER4T1NG, x5Y5Z0T3Ms0x, 0xDu0xMBASSx)

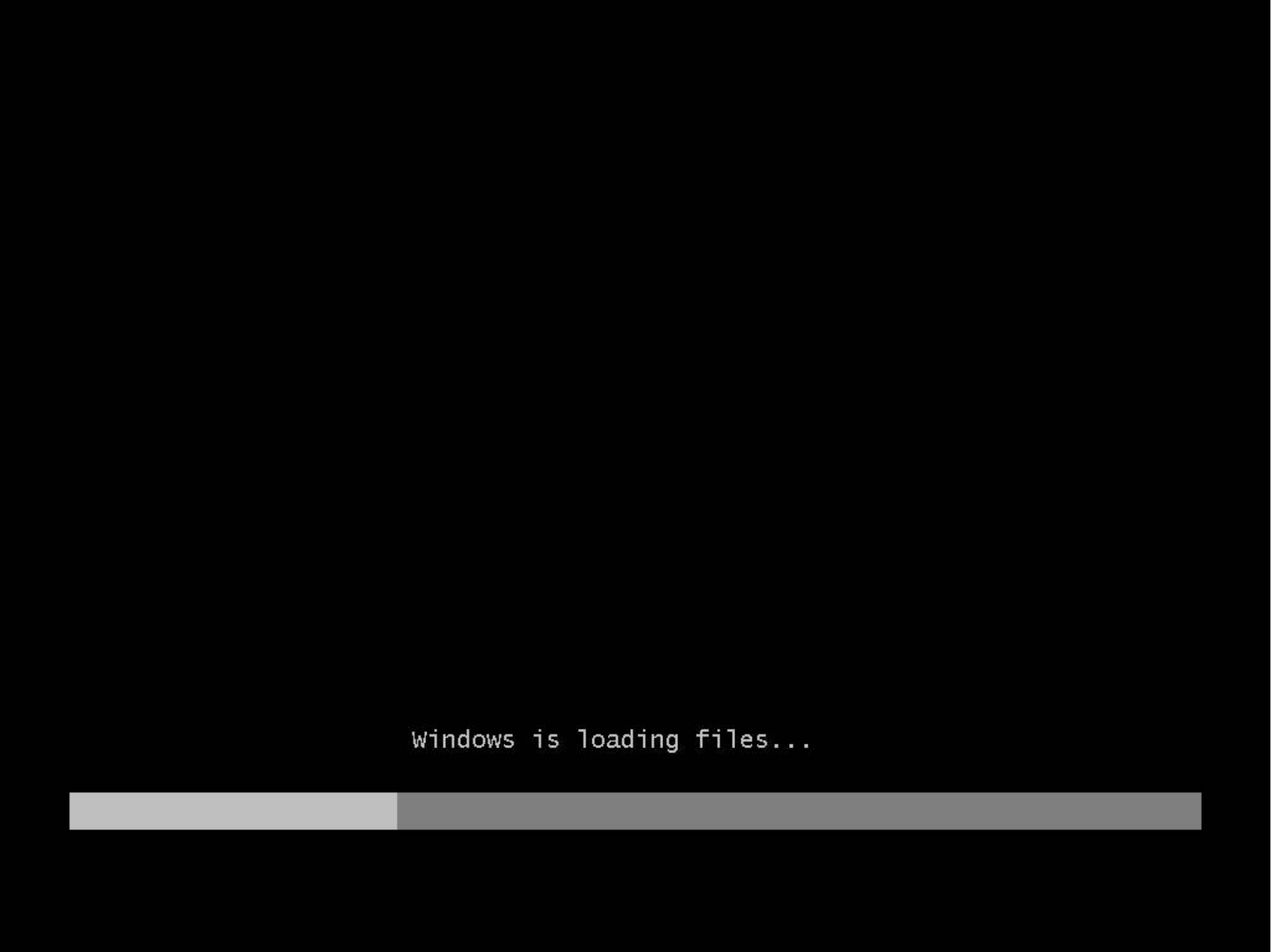
*** REASON_TO_USE_WINDOWS - Bad Idea, xU382xk,as00
            \system32\systemfilezz\REASON_TO_USE_WINDOWS.SYS

beginning to dump physical memory in order to hide evidence
physical dump complete - lol. dump.

Contact one of your many IT Technicians or your Administrator (like he really knows anything) for further
gri - i mean assistance.

THX 4 US1N M1CR0SOFT W1NDOWZ, YO..


Starting Windows
© Microsoft Corporation


Windows is loading files...

重启、重装、复位，何以凑效？

状态重置的威力

- 系统错误往往体现为系统进入了未知的、难以观测、难以控制、难以预知的错误状态
- 重启、重装、复位这些操作，通过将系统重置为我们熟悉的、稳定的、可控的、可预知的状态，来解决问题

状态重置的问题

- 状态重置意味着状态丢失
- 大粒度的状态重置往往会带来高昂的状态重建成本
  - 重启系统后，要重新打开浏览器、 IM 等常用程序，耐心等待它们启动完毕
  - 重装系统后，要重新安装常用程序、恢复备份数据
  - 游戏服务器重启服务会导致用户掉线重连

针对患部做细粒度状态重置？

细粒度的状态重置

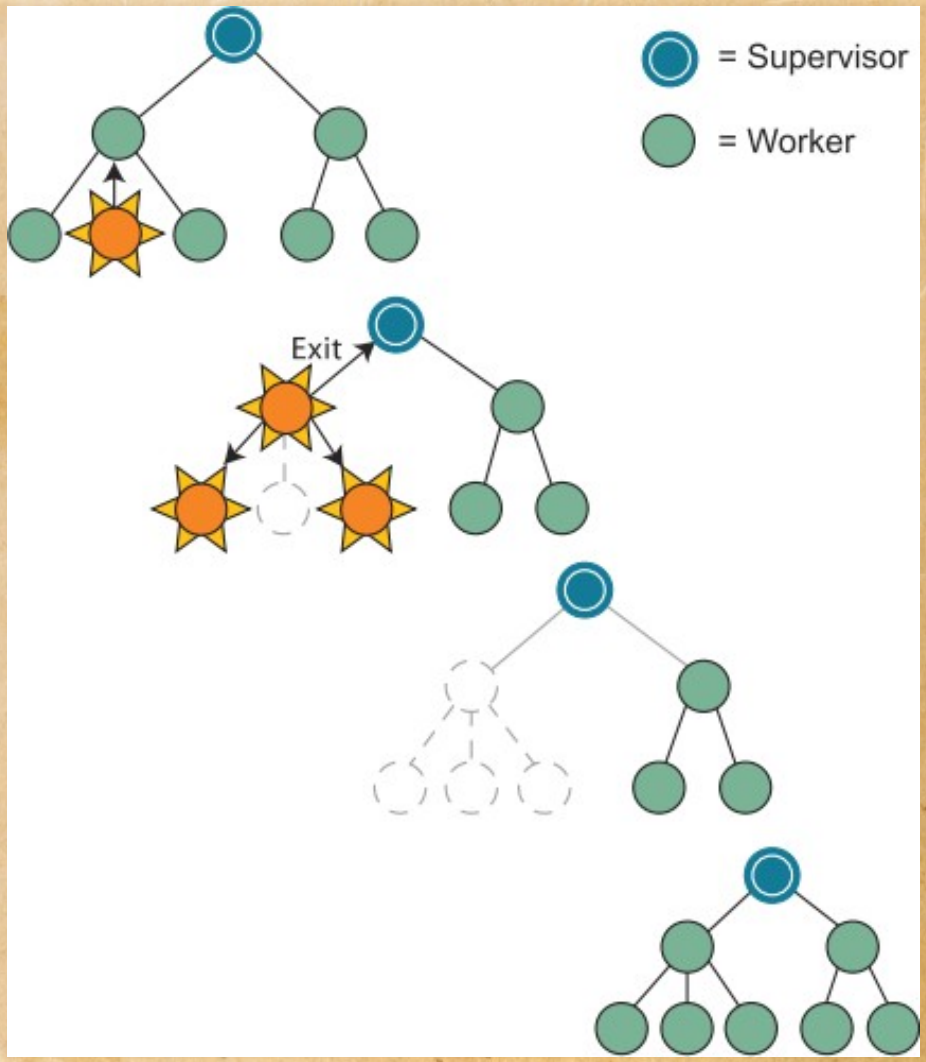
- 将系统任务按自顶向下分出层级
  - 顶层任务最复杂、底层任务最简单
  - 底层子任务故障只会导致系统降级，不应导致整个系统不可用
- 故障隔离
  - 故障任务不应影响正常任务
- 故障恢复
  - 通过重启故障任务尝试修复故障，如无法恢复则立即退出并逐级上报

Erlang 的容错之道

- 海量轻量级进程
  - 进程语义隔离故障
  - 进程间通过消息传递通讯
- 监督进程和工作进程共同构成树状结构
  - 利用监督进程和工作进程间的链接自底向上定向传播故障
  - 监督进程负责发现和重启故障进程
  - 重启失败则监督进程关闭辖区内的所有进程，上报故障

Fail Fast, Let It Crash!

- 利用 Erlang 轻量级进程将任务粒度切细
- 进程间无共享数据保证故障隔离
- 利用监督进程重启故障任务实现细粒度的状态重置



Concurrency Oriented Programming

定义

- By Joe Armstrong:
  - 在语言层面支持进程语义
  - 每个进程都具有全局唯一标识
  - 进程之间高度隔离，不共享任何状态，仅靠异步消息传递进行通讯
  - 不保证消息传递的可靠性
  - 进程可以监测到其他进程的故障

Erlang COP 的威力

- 海量轻量级进程
  - 每个进程启动时默认占用 309 个字（非 SMP VM ）
  - 16G 内存的 64 位服务器可支持近 700 万轻量级进程
- 并发逻辑得以简化
  - 原始的阻塞多进程模型得以再次粉墨登场
  - 无锁，靠消息传递同步并发逻辑
- 位置透明，简化分布式应用的设计实现
- 对现实世界的建模更加直观

代码示例

```
-module(area_server0).
-export([loop/0]).

loop() ->
    receive
        {rectangle, Width, Ht} ->
            io:format("Area of rectangle is ~p~n",[Width * Ht]),
            loop();
        {circle, R} ->
            io:format("Area of circle is ~p~n", [3.14159 * R * R]),
            loop();
        Other ->
            io:format("I don't know what the area of a ~p is ~n",[Other]),
            loop()
    end.
```

示例

```
1> Pid = spawn(fun area_server0:loop/0).
<0.36.0>
2> Pid ! {rectangle, 6, 10}.
Area of rectangle is 60
{rectangle,6,10}
3> Pid ! {circle, 23}.
Area of circle is 1661.90
{circle,23}
4> Pid ! {triangle,2,4,5}.
I don't know what the area of a {triangle,2,4,5} is
{triangle,2,4,5}
```

COP v.s. OOP

- COP
  - 模块
  - 进程
  - 协议
  - 消息传递
- OOP
  - 类
  - 对象
  - 接口
  - 方法调用

Functional Programming

Erlang 不是纯函数式语言

- Erlang 只是一门具有函数式特征的语言
  - 变量单次赋值
  - 尾递归优化
  - 模式匹配
- Erlang 大量依赖副作用
  - 消息传递（包括文件、网络等 IO ）
  - 进程的创建和销毁

为什么选择函数式？

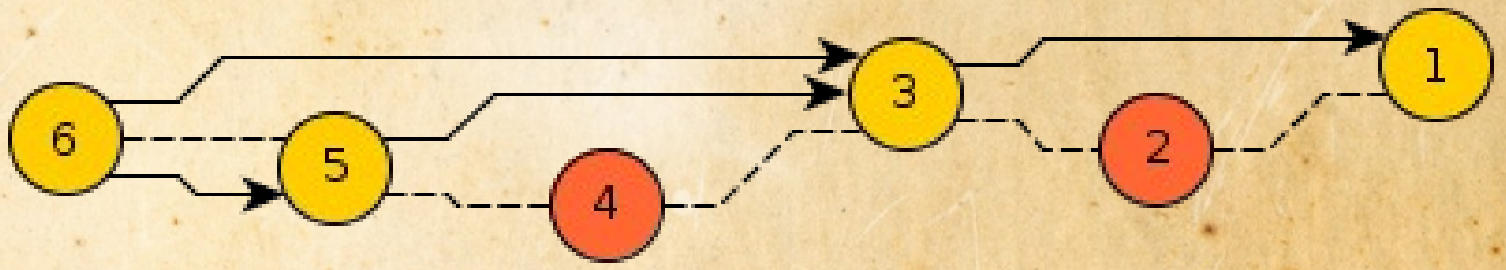
- 绝不是出于学院派喜好
- 函数式编程天生适合并发系统

Single Assignment

- 优势
  - 变量只读，因此状态只读
  - 并发状态访问无需加锁
  - 便于实现代码热替换
  - 简化 GC 实现
- 劣势
  - 加重了 GC 的负担（但被 COP 很好地解决了）

Single Assignment

- 简化 GC
  - 由于变量不可修改，老旧对象不会持有年轻对象的引用，内存中的对象引用关系必定是一张从年轻对象指向老旧对象的 DAG
  - Armstrong, J., & Virding, R. (1995). One pass real-time generational mark-sweep garbage collection.



Single Assignment & COP

- Erlang VM 的 GC 以 Erlang 进程为单位实施
- Erlang 进程的内存堆通常很小
- 整个 VM 的 GC 均摊到各个进程，天然减小了 GC 停顿时间
- 生存期短的进程甚至在触发 GC 前便会终止

Pattern Matching

```
-module(area_server0).
-export([loop/0]).

loop() ->
    receive
        {rectangle, Width, Ht} ->
            io:format("Area of rectangle is ~p~n",[Width * Ht]),
            loop();
        {circle, R} ->
            io:format("Area of circle is ~p~n", [3.14159 * R * R]),
            loop();
        Other ->
            io:format("I don't know what the area of a ~p is ~n",[Other]),
            loop()
    end.
```

Pattern Matching

```
ipv4(<<Version:4, IHL:4, ToS:8, TotalLength:16,
       Identification:16, Flags:3, FragOffset:13,
       TimeToLive:8, Protocol:8, Checksum:16,
       SourceAddress:32, DestinationAddress:32,
       OptionsAndPadding:((IHL-5)*32)/bits,
       RemainingData/bytes >>) when Version =:= 4 ->
    ...
```

Erlang/OTP

使用 OTP 的理由

- 效率
- 稳定性
- 监督机制
- 代码热升级支持
- 可靠的代码库

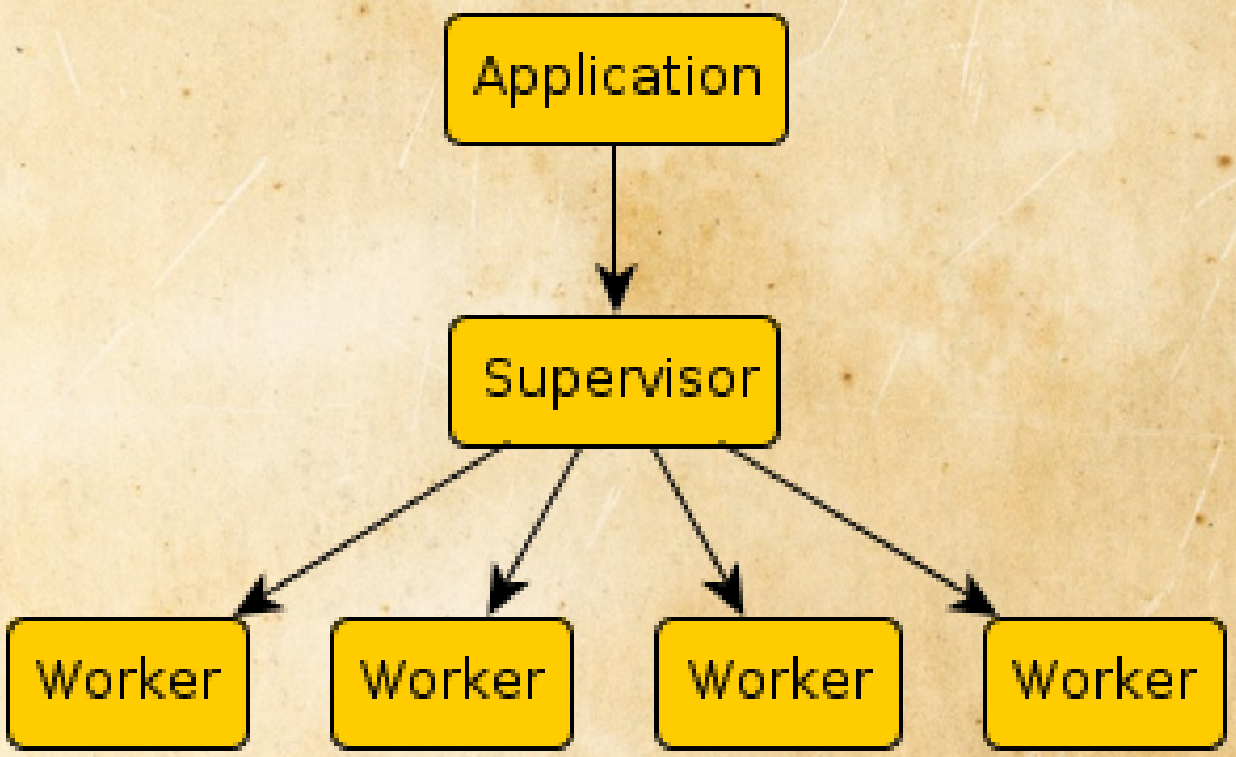
设计模式的先驱

- 从高强度电信系统中抽象出了若干通用模式
  - gen_server
  - gen_event
  - supervisor
  - application
  - ...

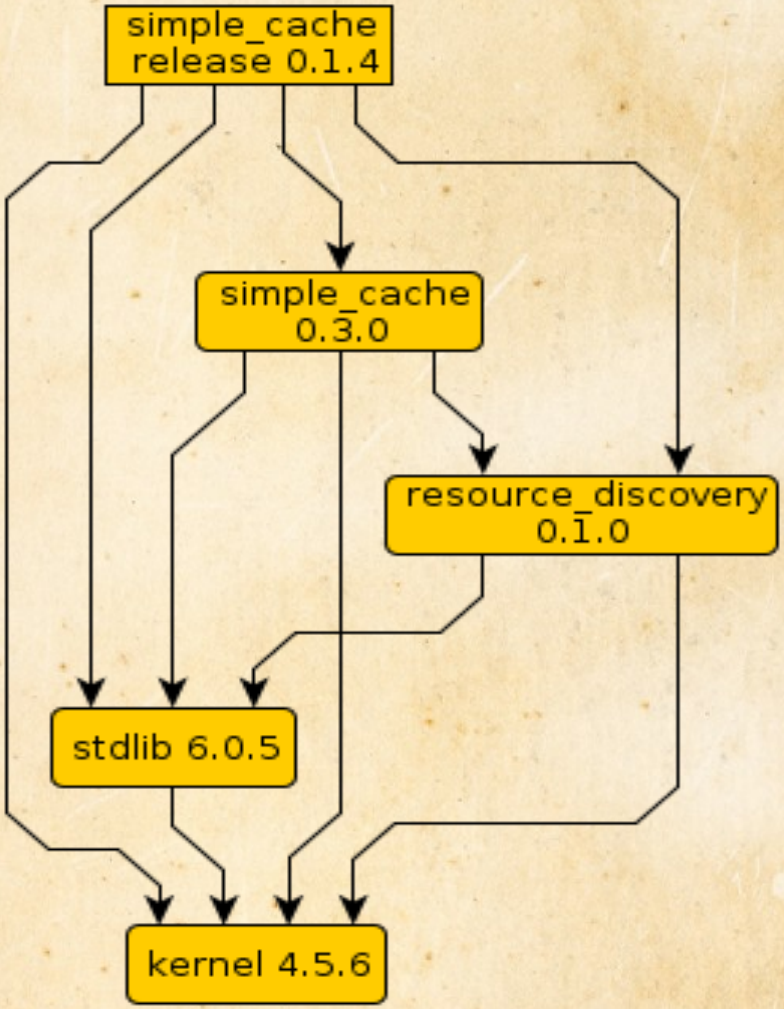
层级化的工业级应用解决方案

- 模块（ Module ）
- 应用（ Application ）
- 发布镜像（ Release ）
- 目标系统（ Target system ）

模块与应用



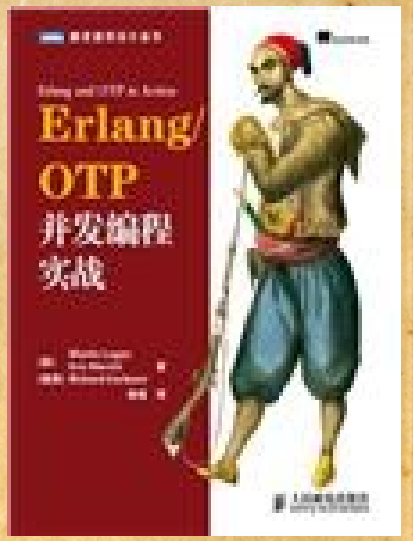
应用与发布镜像



- 发布镜像
  - 多个应用的集合
  - 统一管理应用间的依赖关系
  - 处理热升级和回滚

参考文献

- Armstrong, J. (2007). A History of Erlang
- Armstrong, J. (2003). Making reliable distributed system in the presence of software errors
- Logan, M., Merrtt, E., Carlsson, R. (2011). Erlang and OTP in Action



用 Erlang 构建容错系统

- 连城 (rhythm.mail@gmail.com)
- Twitter @liancheng
- 微博 @ 连城 404

大型电信公司（ AT&T 和爱立信）门下的研究机构（贝尔实验室和爱立信计算机科学实验室）；一小撮才华横溢、自由自在的工程师和研究者；面对在各自时代最为复杂的工程问题（ UNIX 和电信系统）；在内部产品中大量应用、快速迭代；十年磨一剑。

Joe Armstrong 原本将赌注压在 Smalltalk 上，并发明了一套描述电信系统的代数方法。由于 Smalltalk，太慢， Joe 订购了一台 Smalltalk 机，在送货期间，同事告诉他 Prolog 可以很好地描述他的代数方法。等到 Smalltalk 机到货时， Joe 已经对 Smalltalk 失去兴趣了。

《七龙珠》中的魔人布欧：具有超强的再生能力，可分裂为多个独立个体，可通过吞噬其他个体获得对方的能力

如影随形的终极错误修复手段

蓝屏怎么办？

重启……

重启还解决不了问题怎么办？

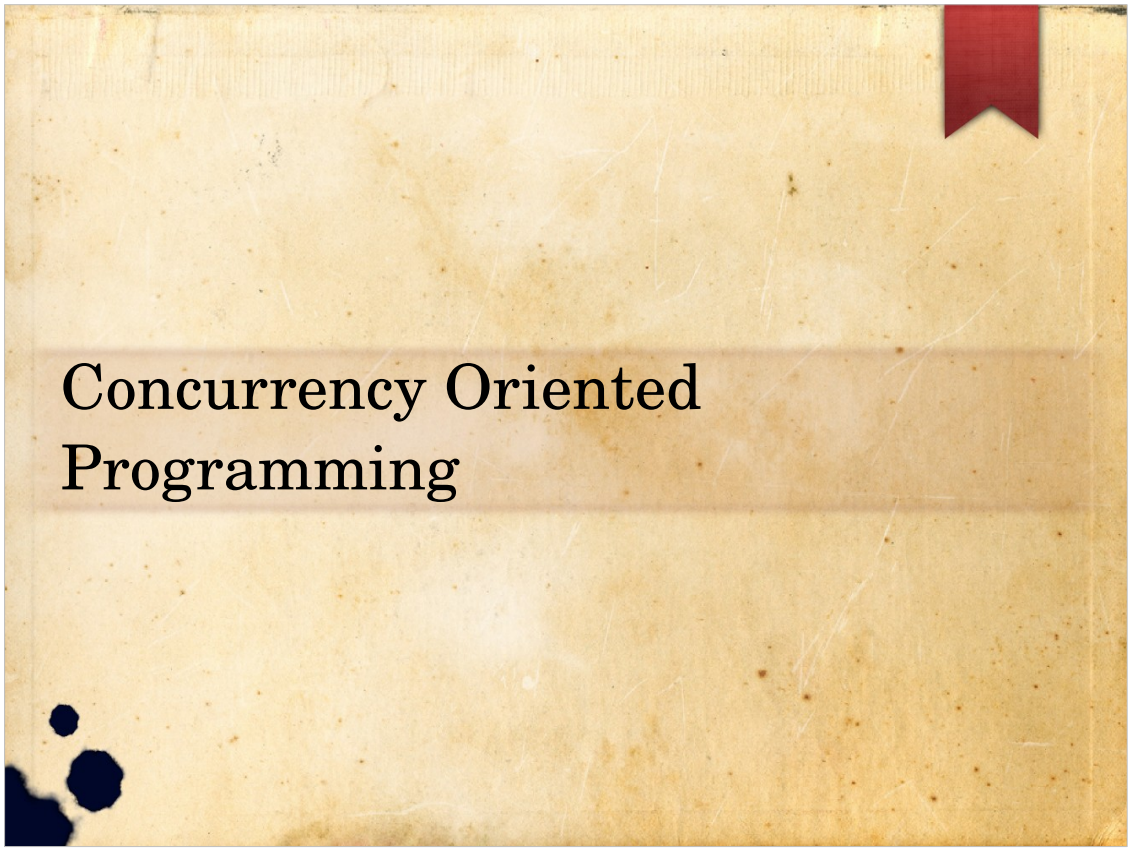
重装……

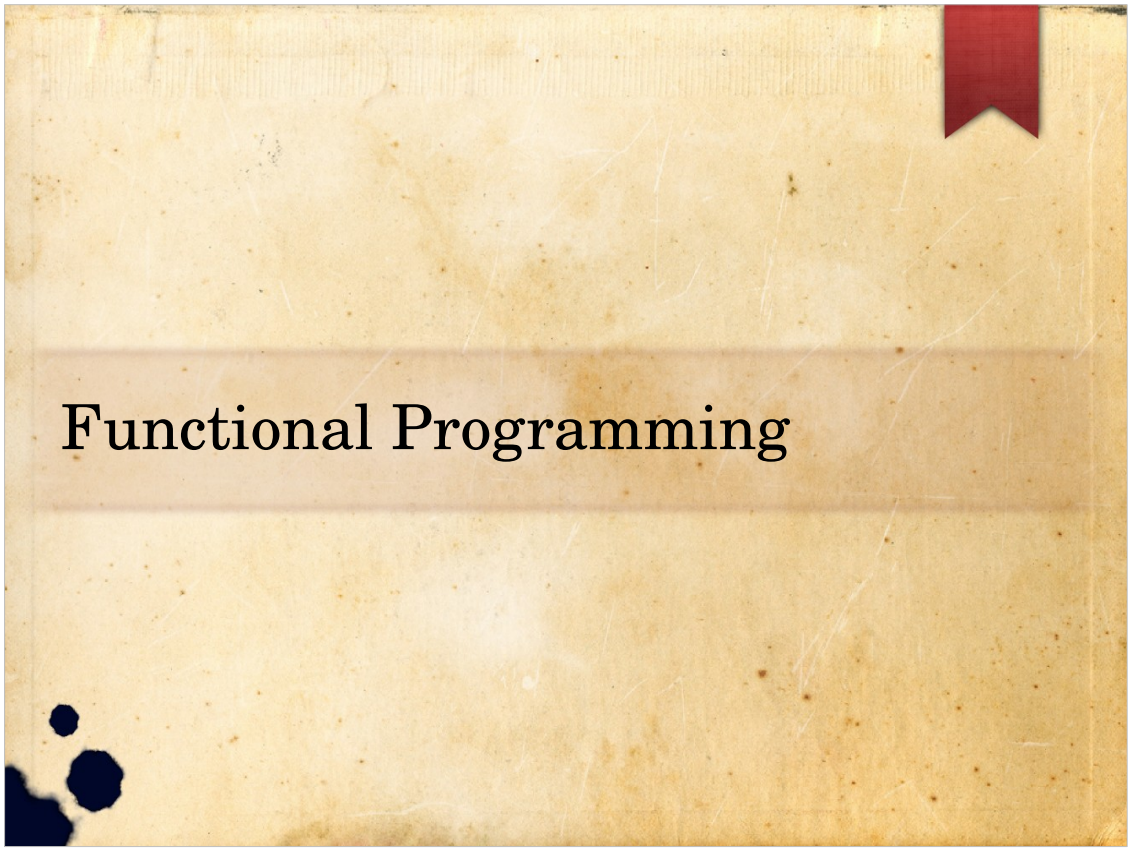
如果说用重启机器和重装系统来对付蓝屏是不得已而为之，那么下面这个例子绝对是精心设计而成的

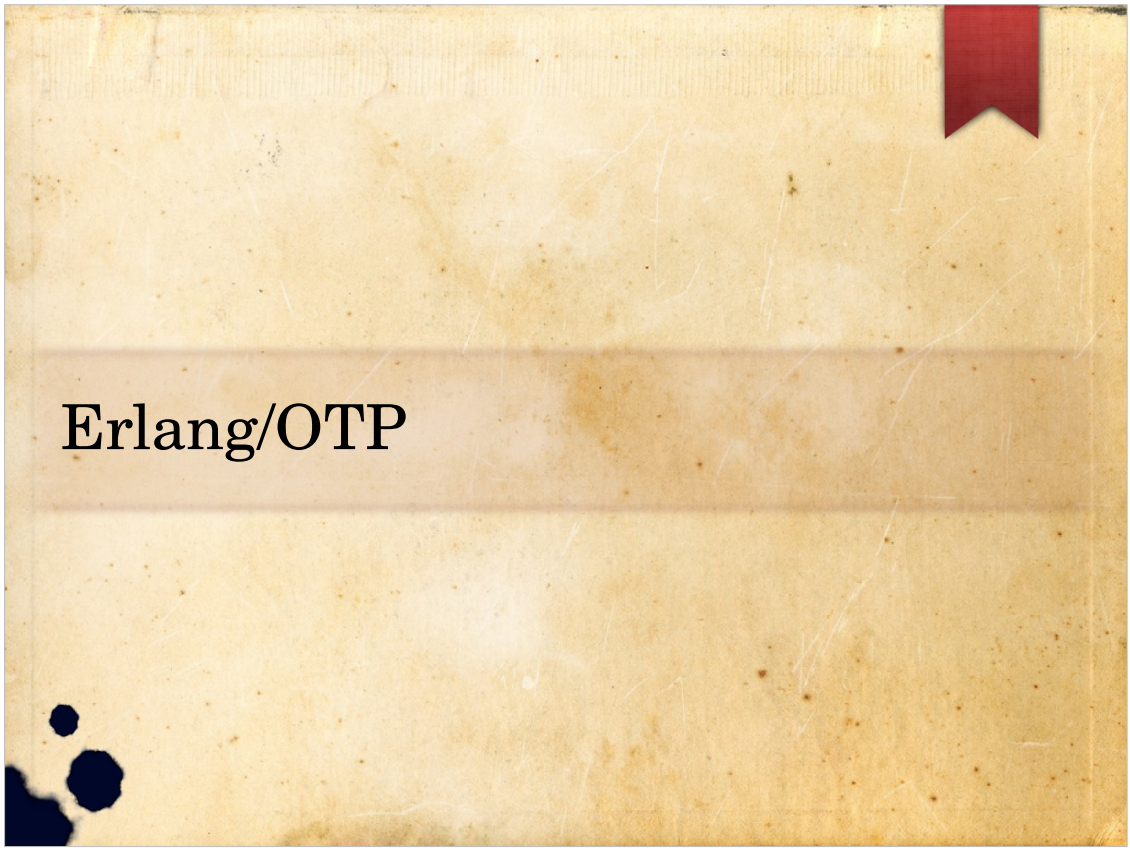
有调查显示相对于 Android 设备，中老年人对 iPhone 、 iPad 上手更快。其中很重要的一个原因就是在使用过程中无论触发了什么诡异的问题，只要按一下 Home 键，就可以立即回到熟悉的主界面。

重启、重装、复位，何以凑效？

针对患部做细粒度状态重置？


Concurrency Oriented
Programming


Functional Programming


Erlang/OTP

機