❖优化案例

Erlang应用优化案例

- Ehttpd测试
  - 输出“Hello world”
  - 可超过20000并发短链接
- Hotwheel 40000广播
  - Google hotwheel
- 简单Key/Value查询系统

HTTP echo每秒20000短连接单个CPU

- Taskset -c 1　erl +K true +h 99999 +P 99999　-s ehttpd
- 优化前后对比
  - 11203
  - **20090**
- 硬件普通桌面双核CPU，2G内存
- 微调Linux VM和协议栈，32位操作系统
- 优化和patch了Erlang VM，采用beam.plain
- 优化了ehttpd程序，采用系统高级网络选项

Hotwheel广播服务

- Joel悬赏$2000，挑战20K
- 成功挑战通过每CPU 40K/s
- 这个应用代表了大部分网络服务程序的模型，对于整个业界水平的提高很有借鉴意义

简单Key/Value查询系统（身份证查询系统？）

- 测试硬件
  - 8核心
  - 16G内存
- 测试结果
  - 并发长链接数1000000
  - 并发查询100000/s

❖预备知识

优化的层次

- 选型
- 操作系统
- Erlang VM
- 语言
- 集群
- 业务

Erlang适合做什么

- IO 密集型
  - 高度优化完备的IO, 顶尖的C高手20年的耕耘
- 高性能网络服务器
  - 多年的开发
  - 非常完善
  - 类似于一个操作系统
  - 很好的处理掉了[高性能服务器Seven Sins]
  - 轻松达到C10K
- CPU利用
  - 先进的SMP调度器更好的利用多核心CPU

Erlang和操作系统的类比

- *nix操作系统，用C++做例子
  - 函数 (void fun() {})
  - 类 (class mod{};)
  - 模块(mod.cpp)
  - 可执行文件（编译器，机器指令）
  - 应用程序包括数据文件
  - OS启动，系统进程（抢占式调度 ）
  - IPC通讯
  - 监控工具(Top)

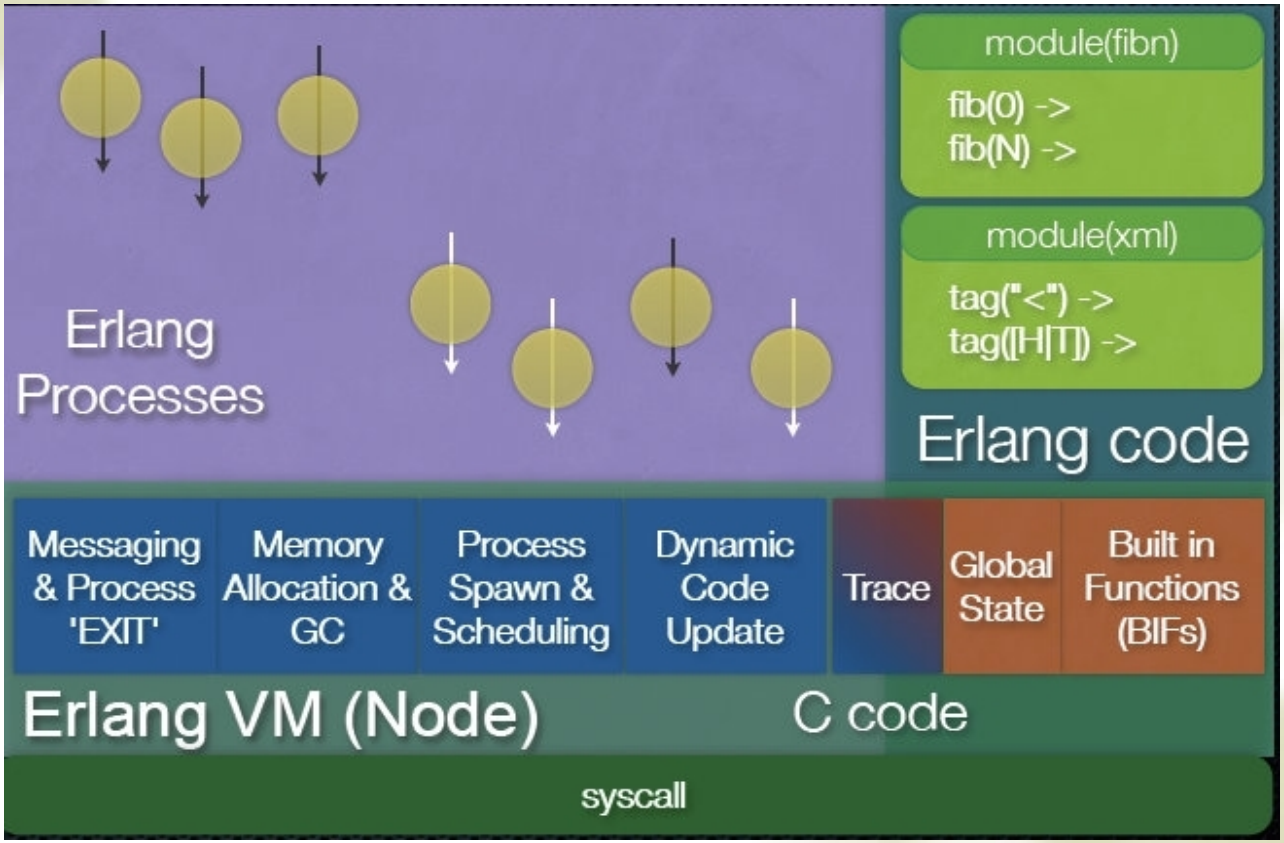
Erlang和操作系统的类比 (cont'd)

- 函数
  - fun () -> ok end
- 模块
  - module mod. mod.erl
- Beam文件
  - 编译器opcode
- Application
  - beam+数据文件

Erlang和操作系统的类比 (cont'd)

- VM bootstrap
  - Erlang进程 (抢占调度)
- 消息
  - Port
  - IPC
- 工具集
  - etop

ERTS内部结构


module(fibn)
fib(0) ->
fib(N) ->
module(xml)
tag("<") ->
tag([H|T]) ->
Erlang Processes
Erlang code
Messaging & Process 'EXIT'
Memory Allocation & GC
Process Spawn & Scheduling
Dynamic Code Update
Trace
Global State
Built in Functions (BIFs)
Erlang VM (Node)
C code
syscall

Erlang进程调度原理

- 调度原则
  - 尽量让一个CPU忙
  - Logic CPU从低到高
- 上下文切换
  - context_switch开销
- 消息传递的开销
  - 拷贝
  - malloc/free
  - 垃圾收集

Port调度原理

- Port独立调度
- 和宿主进程同一个调度器
- 调度的单位是该Port触发的一串IO事件
- 调度延迟
- busy_port
- 水位线buffer
- 锁

- 如何优化

工具方法

- 理解了Erlang和*nix的相同点
- 借鉴*nix 的工具和方法
- 创造工具和方法

操作系统层面的优化

- 操作系统的选择
  - 32位系统 vs. 64位系统
    - 没有内存空间限制
    - 64位比较慢
  - RHEL上游厂商致力于高性能操作系统
    - Vdso
    - RhelRt
  - 重新用ICC编译内核和glibc
  - VM和TCP协议栈的优化

操作系统层面的优化

- 降低系统的swapness, 避免内存颠簸
- 资源倾斜，全力服务应用
- 给我尽可能多的物理内存，越多越好

Erlang运行期层面优化

- 新版本的OTP致力于SMP方面的改进
  - 更细粒度的锁
  - 更好的内存分配器
- Hipe（erlang的jit）
  - 全面启用preloaded otp库
- Crack系统
  - 减少无必须的系统调用
- 参数微调
  - Effective guide
- 未公开的特性
  - 调度器绑定

语言层面优化

- 减少VM GC开销
- 进程字典
- 加大min_heap_size
- Hibernate
- Cache
- Lazy eval
- Record或者tuple变化部分和惰性部分分开