Erlang在云上数据链路的应用

By 武藏

http://weibo.com/u/1968605513

个人介绍

- 2012年加入阿里云
- 参与RDS多个组件开发
- Erlang应用与调优
- Rabbitmq
- 实时计算

ApsaraDB

- SQL，NOSQL， HTAP
- 强大的数据研发团队
- 全方位的数据库服务
- 行业领先的技术

数据链路主要产品

ApsaraDB Proxy

- DB高可用
- 安全审计
- 负载均衡
- 读写分离

HTAP DB

- 分库分表
- 分布式事务
- PB级存储
- 在线分析

云上数据链路的挑战

- 数据库服务极其严苛的可用性要求
- 云上业务多样性带来的困难
- 超高并发带来的问题
- 运维实体从几十到几十万带来的问题
- 资源倾斜，造成的服务不稳定
- 低成本与高效率的矛盾

云上数据链路的挑战

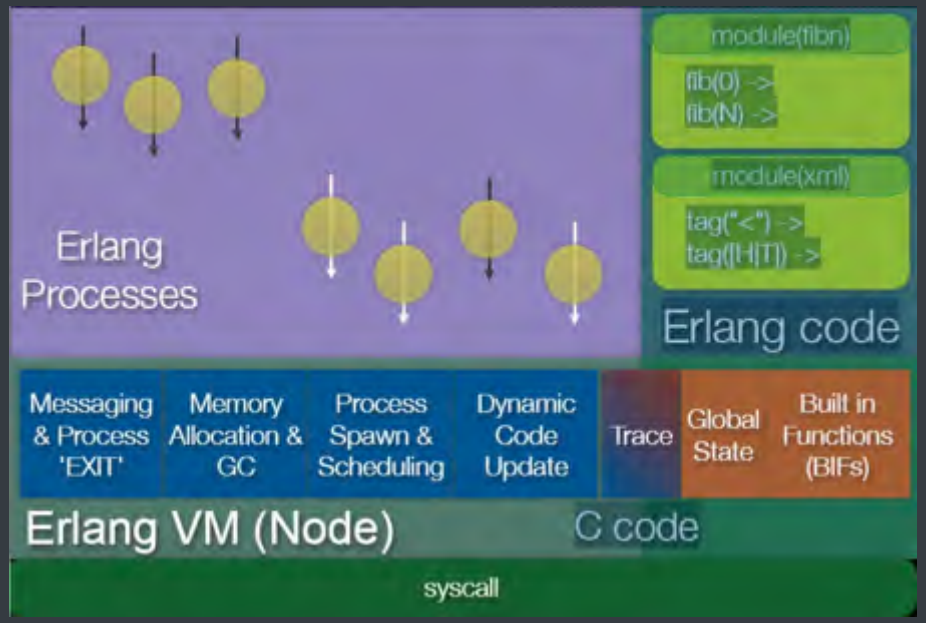
解法：

Start From Erlang

Erlang提供的基础设施

Erlang是高性能，高可用的平台，提供相关基础设施。负责：

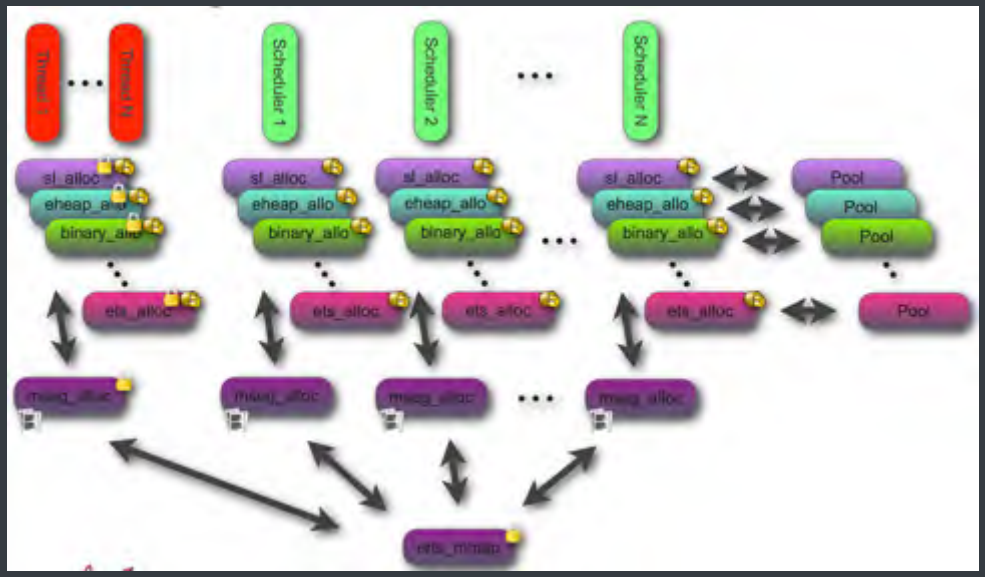
1. 任务调度
2. 内存管理
3. 应用隔离
4. 网络框架
5. 高可用框架
6. 代码热替换



Erlang 的高性能

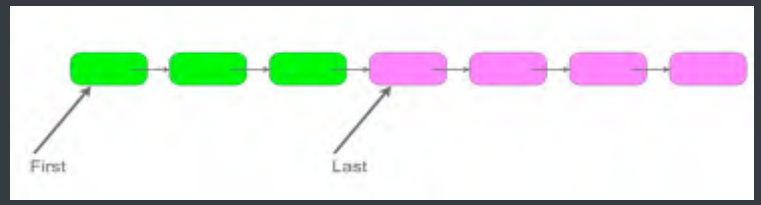
高效的内存分配体系：

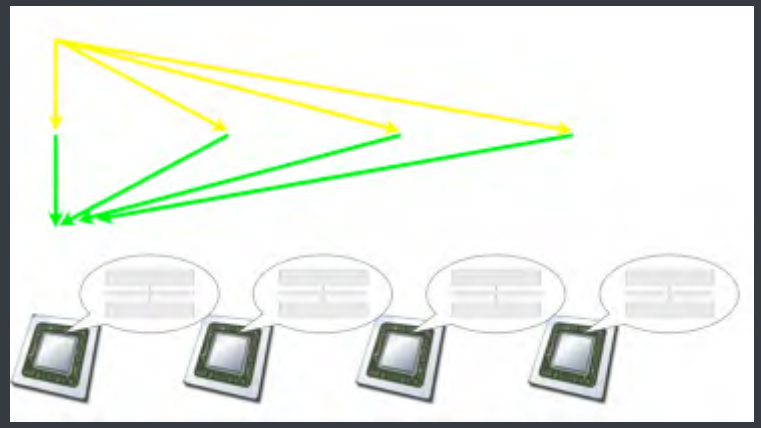
1. 多层次的内存管理
2. 根据不同对象的特点定制分配算法
3. 无锁算法
4. 高度可配置
5. 不会stop the world



Erlang 的高性能

高效的消息通讯机制：
1. lock-free queue
2. Thread Progress同步机制



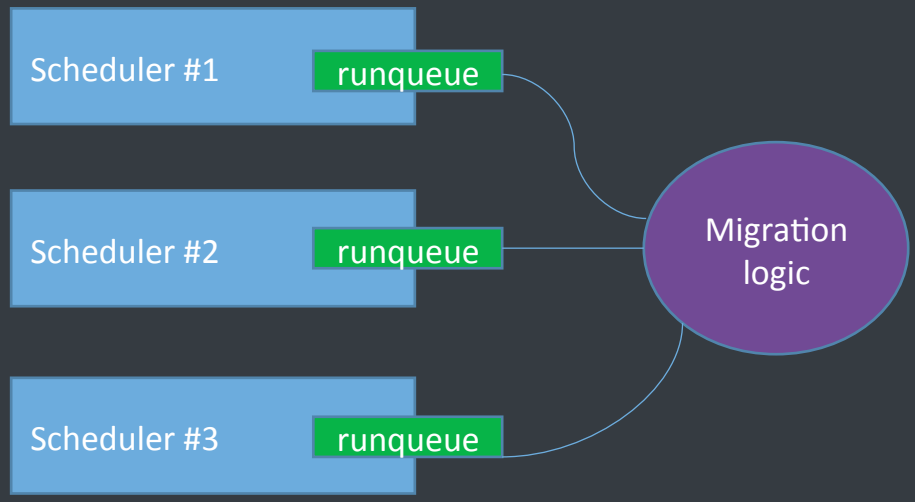


Erlang的高性能

高效的调度器：

1. 软实时的公平调度
2. 超轻量级的进程，轻松应对c1000k
3. 基于计数的调度,综合考虑了IO，CPU和网络的开销，为实现更好的Qos奠定了基础

4. 调度开销低



Erlang的高可用

进程隔离:

1. 应用以超轻量级的进程方式组织，故障隔离
2. 进程间不共享数据
3. 消息通讯，相互解耦

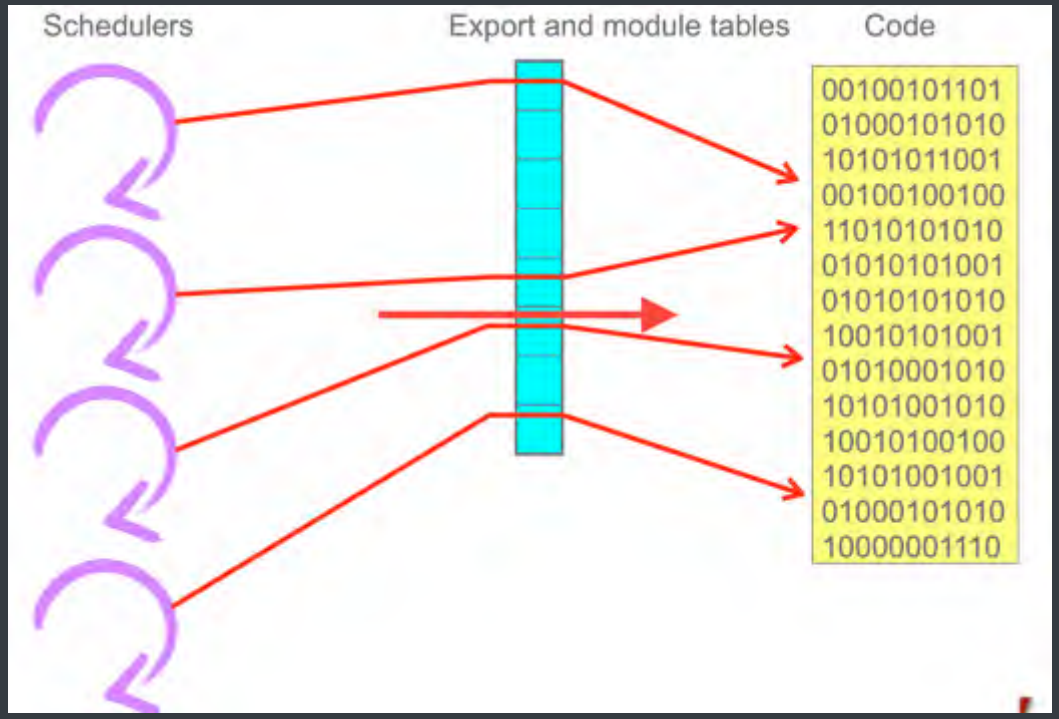
监督树模型:

1. 树形层次化结构
2. 进程挂断后，由supervisor处理
3. 重启解决大部分问题，Let it crash !

Erlang的高可用

热代码替换：

1. 每个方法有两个槽位。
2. 运行同时运行两个版本的代码。
3. 代码替换不中断服务，不需要全局同步
4. 完善的上层配套机制，便于实现从模块到应用再到集群的热升级



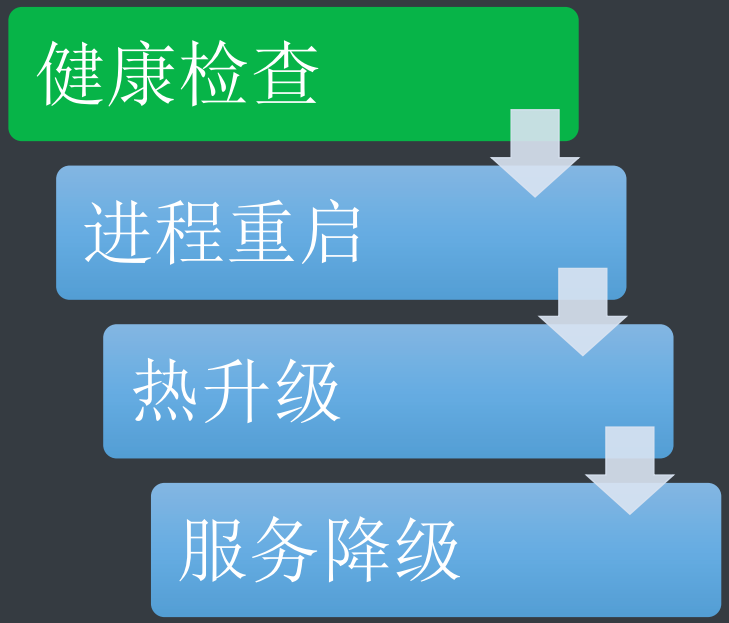
Erlang自省机制

- 强大的trace机制和配套工具
- 系统层面，方方面面的指标信息：调度器，内存分配器，GC，IO，网络等
- 自带多种profile工具
- 信息获取规范，每个模块都有info函数
- 大量dtrace埋点

Erlang开了个好头

高可用与服务质量保证

高可用



场景：
1. 硬件故障
2. 操作系统故障
3. 内部逻辑bug造成的假死

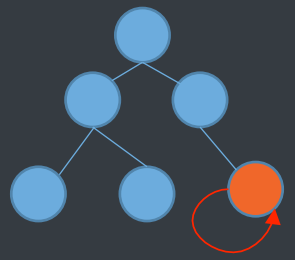
措施：
检查到异常后，将SLB路由摘掉，流量导走

高可用

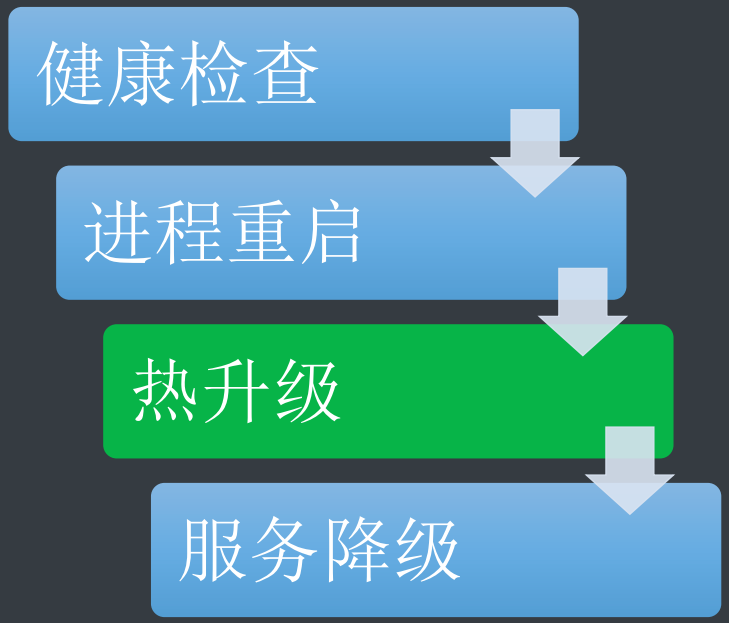
- 健康检查
- 进程重启
- 热升级
- 服务降级

故障时，利用OTP的监督树机制：

1. 控制故障范围，首先尝试重启
2. 关键有状态进程，状态保存到ETS，重启后恢复状态



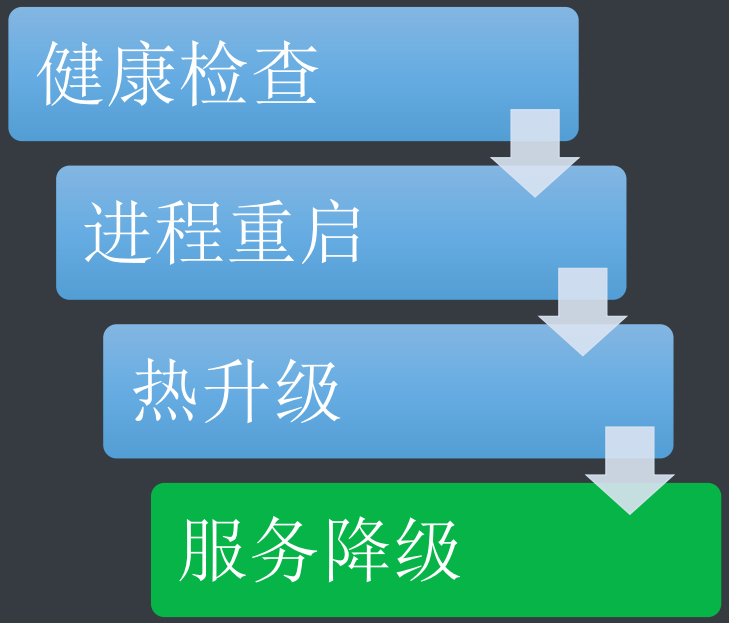
高可用



热代码替换：

1. 减少计划内的服务中断
2. 小步快跑，便于灰度
3. 逻辑变更与数据变更紧密配合
4. 相邻版本的兼容
5. 老代码监控

高可用



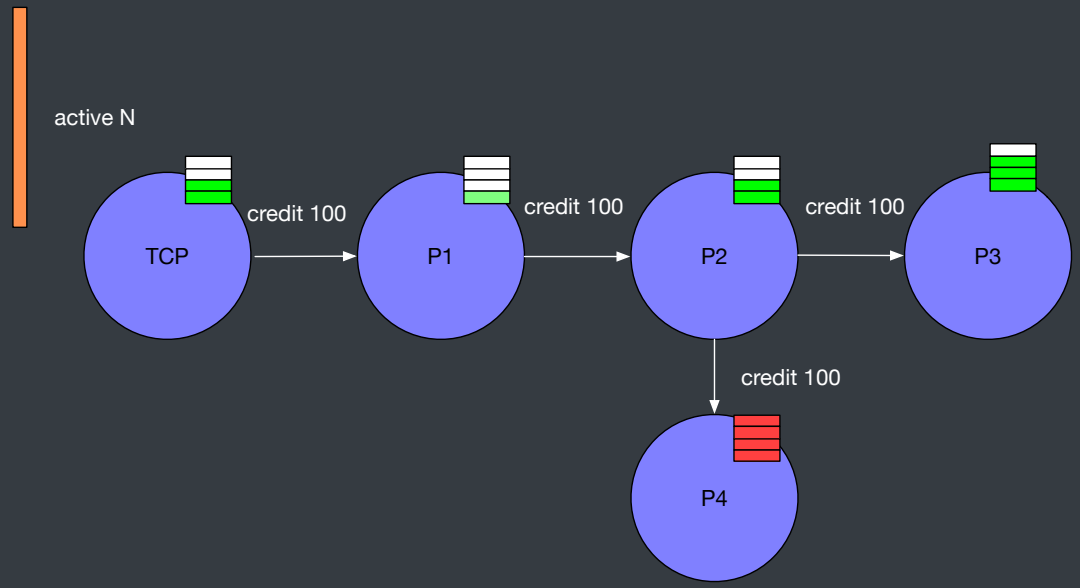
多种措施：

1. 降quota
2. 停止接收新连接
3. 安全审计变为抽查
4. 流量透传
5. 停止支持压缩

服务质量保障

反压机制：

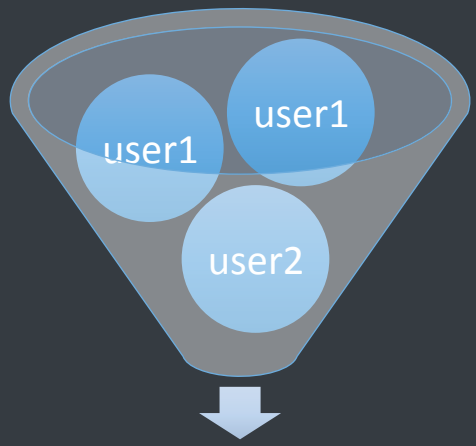
1. 逐级授信
2. 最终阀门： active N



服务质量保障

Quota控制：

1. 连接数控制
2. Token buckets的流控算法
3. 用户级流控
4. 核心基于ETS的计数器



$$LimitQPS_i = max(\frac{MaxQPS}{TotalQPS} * QPS_i, \frac{MaxQPS}{Connections}) \quad (2)$$

性能优化

Profiling工具

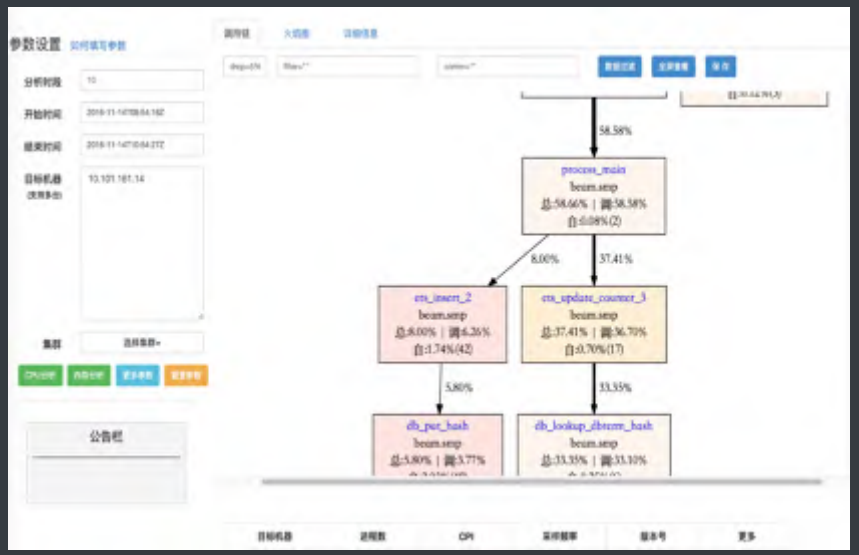
- etop
- fprof/eprof/cprof
- systemtap
- perf
- warden
- 扁鹊

工欲善其事必先利其器！

扁鹊

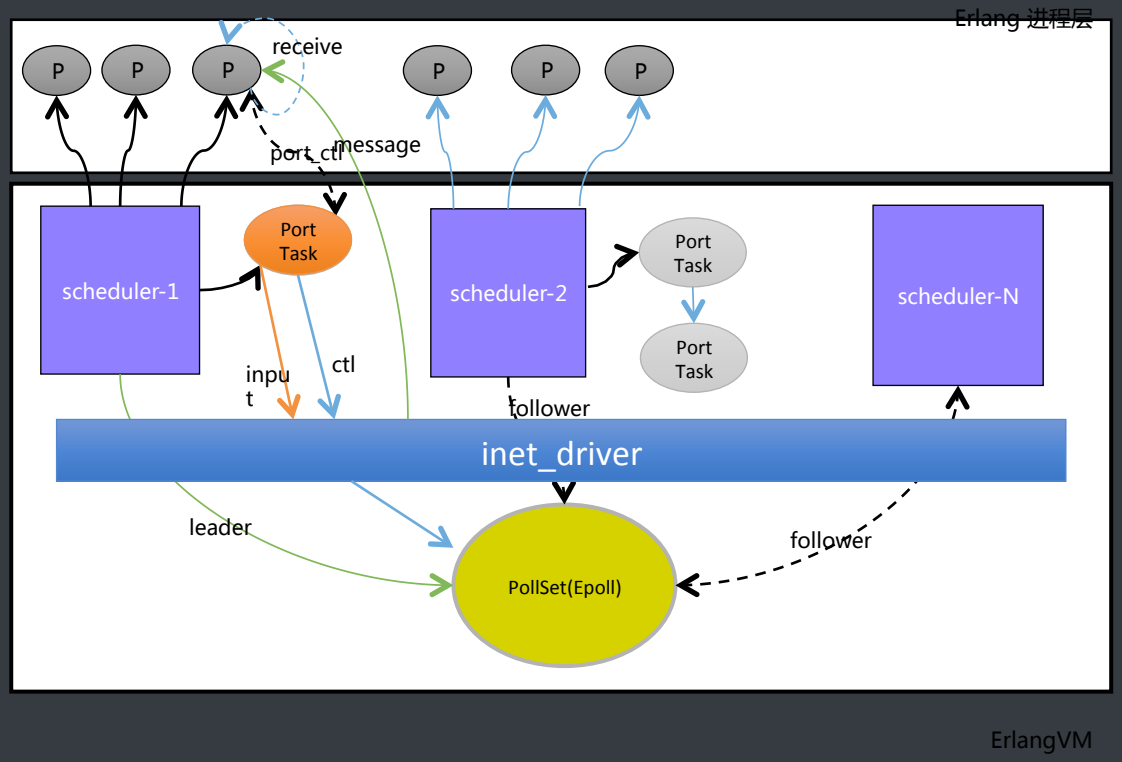
功能：

1. 收集内核态和用户态stack，函数调用频率信息
2. patch erts收集erlang层面调用链
3. 开销极小



Multi pollset

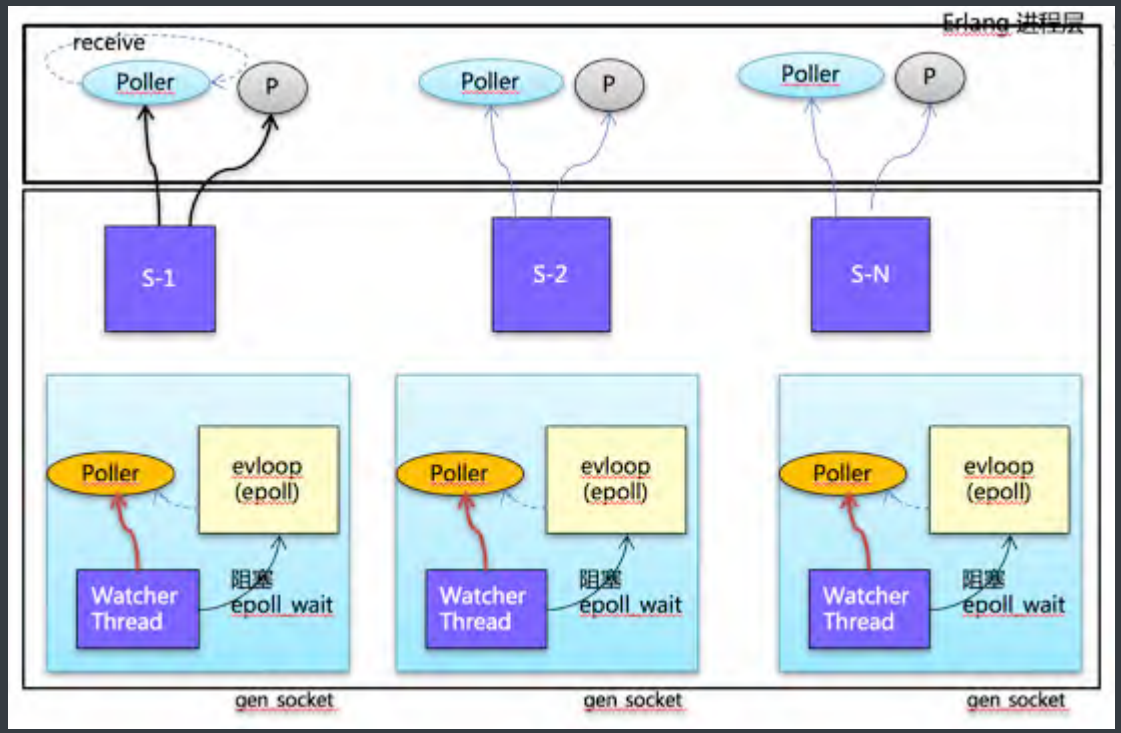
gen_tcp问题：
1. 单epoll set



Multi pollset

gen_socket:

1. N个OS线程收割网络事件
2. 对调度器,用户进程透明
3. 接口和gen_tcp完全兼容。
4. 细粒度锁优化
5. 性能提升110%
6. 已开源： https://github.com/alibaba/erlang_multi_pollset



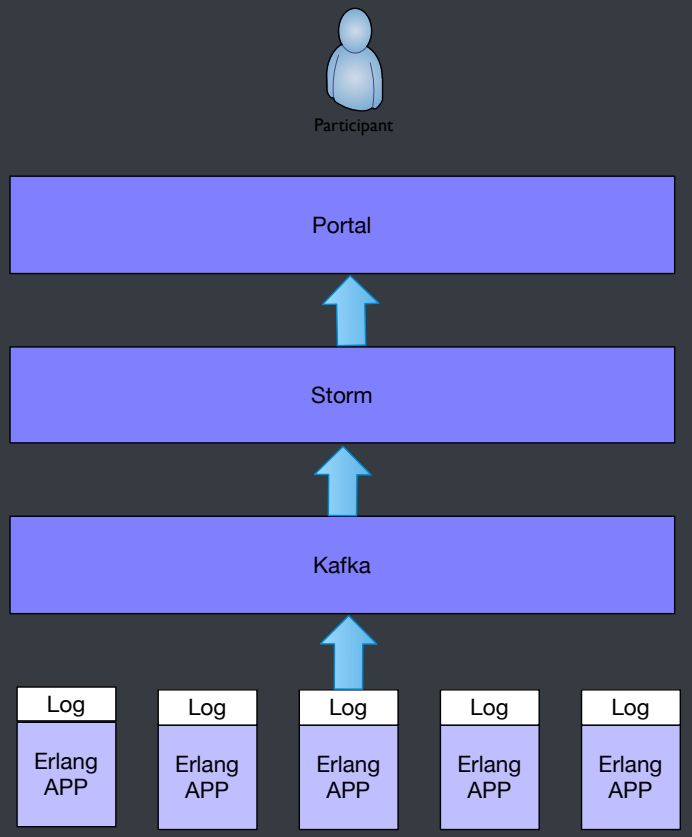
监控体系

采集与处理

数据源:
- recon
- 应用计数器
- system_info/1,memory/1
- etop

传输通道:
1. 落地到带缓存的日志
2. Logagent收集到kafka

处理分析:
1. storm进行预处理
2. PetaDB做汇总分析



指标构成

大类	子类	指标
系统指标	调度器	进程数，runq，util等
	内存	Gc次数，gc量，各分配器容量
	网络	延迟，重传数，吞吐等
业务指标		连接数，qps，rt，各类异常数
概况与日志		虚拟机配置信息，硬件和操作系统信息

统计维度

指标的应用

辅助运维

- 容量评估
- 自动化迁移

异常报警

- 异常模式识别
- 故障范围评估

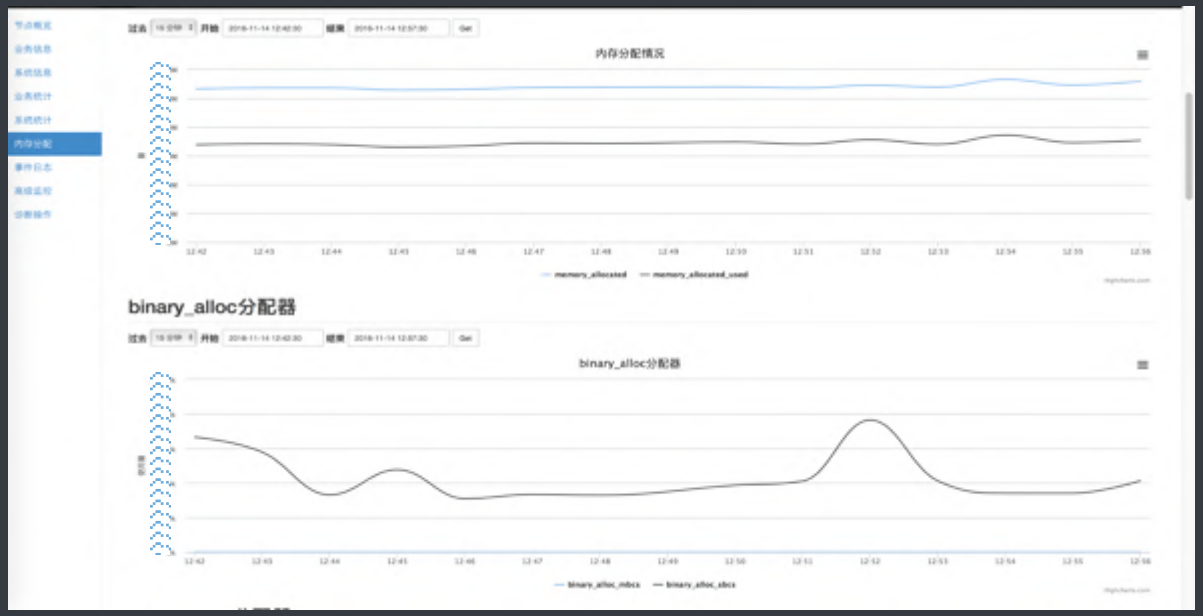
性能优化

- abtest
- 饱和度分析
- USE方法

例子一

监控基础应用数据，掌控性能和余量

列子二



监控各内存分配器的运行状况，可以作为性能调优依据，如内存申请量和网络吞吐的比例

问题与解决之道

内存泄漏

Atom爆炸

- 慎用list_to_atom, binary_to_atom

GC不及时

- 尽量避免产生长期不活跃的进程
- 定期所有进程gc一次

加强监控！

ETS数据未释放

- 业务上避免
- 进程终结时注意清理

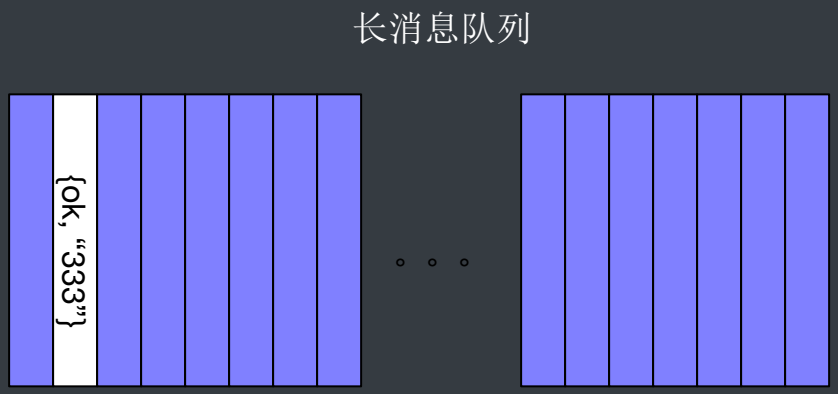
进程泄漏

- 避免只创建不销毁

调度不均衡

NIF引起的	大Io引起的
拆解nif函数为小操作，定期调用 **enif_consume_timeslice**	启用dirty scheduler，配置适当的数量

接收队列堆积



1. 尽量不要阻塞receive特定消息
2. 适当增加处理进程

```
receive
  {ok, Content} -> do_something(Content)
end.
```

热升级注意事项

1. 尽量有soft_purge，purge不成功不要强行替换
2. 数据与应用逻辑的兼容，写好code_change
3. 避免模块间循环依赖
4. 动态启动的进程可能不会执行code_change

其它问题

参考：《Stuff Goes Bad ERLANG IN ANGER》

我们在招聘：

阿里云ApsaraDB相应的团队配套就涉及到OS内核、存储、引擎（TP、AP）、数据库内核、管控、监控、数据流动、计算、搜索、服务等，是个复杂和精美的协作团队, 已经有10几个数据库相关产品，在市场地位和收入上都有不错的表现，团队有业内非常有经验的人，欢迎大家加入，团队主力主要在北京和杭州！