# 贴吧服务端性能优化实践

夏绪宏 xiaxuhong@baidu.com

Velocity Web性能大会 2014

Baidu 百度

# About Me

- 夏绪宏
- 百度贴吧：LAMP技术负责人
- http://github.com/reeze
- Weibo: @reeze
- Email: reeze@php.net

# Agenda

- 性能优化简介
- 贴吧服务端性能优化实践
- 总结展望

# Web Performance Matters ™

“ Performance is a Feature ”

— Jeff Atwood

optimize v. ['ɑptɪ.maɪz]

to make sth as good as it can be.

# Web  performance optimization

Achieve better performance with limited resources
by doing less, do it faster, correctly.

Better: faster, less resource consumption.

# Web  performance optimization

All about Trade off.

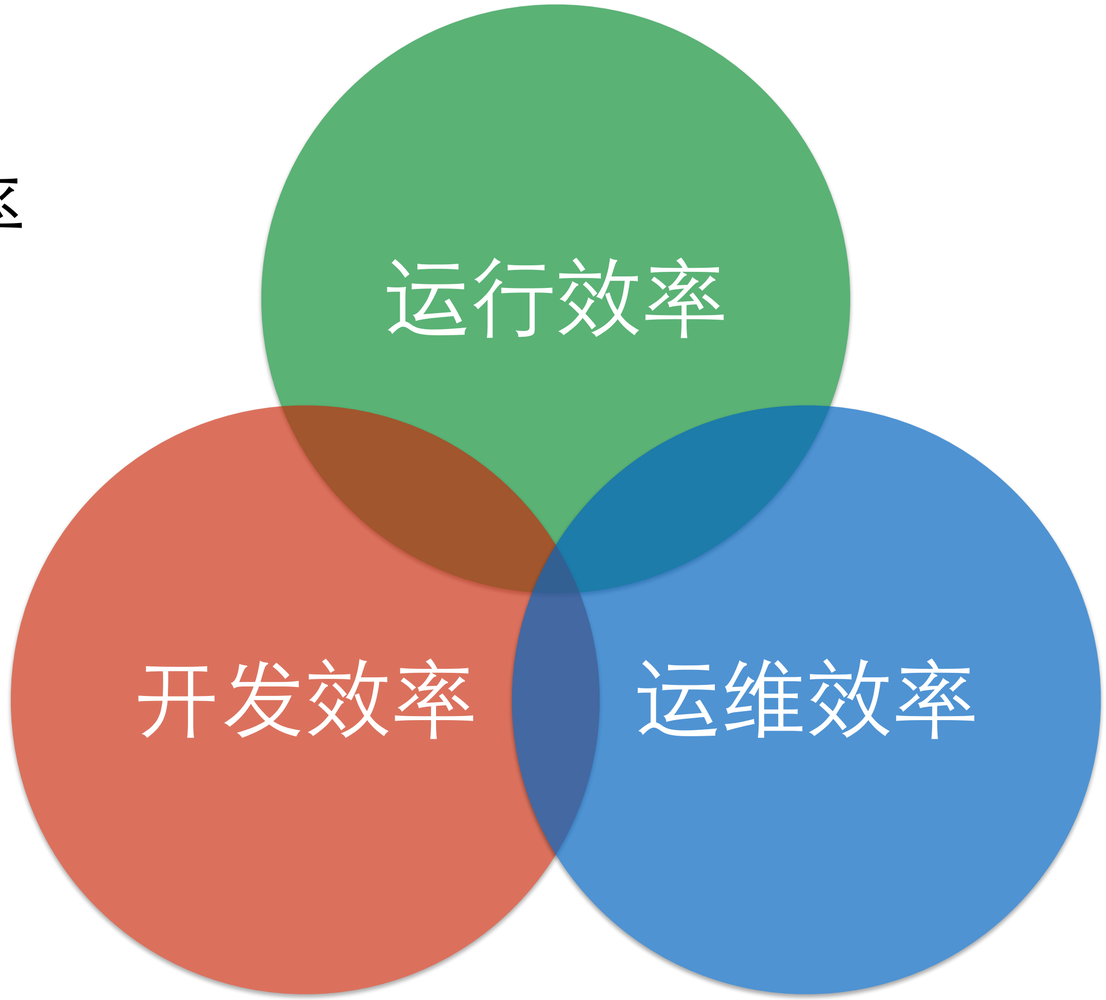
Most of the time, you can’t get them all.

# 全局视角，并非只有运行效率

理想：Get them all.

现实：很难全局最优，Make your choice.

其他变量：资源成本，技术选型等



# Latency and Bandwidth

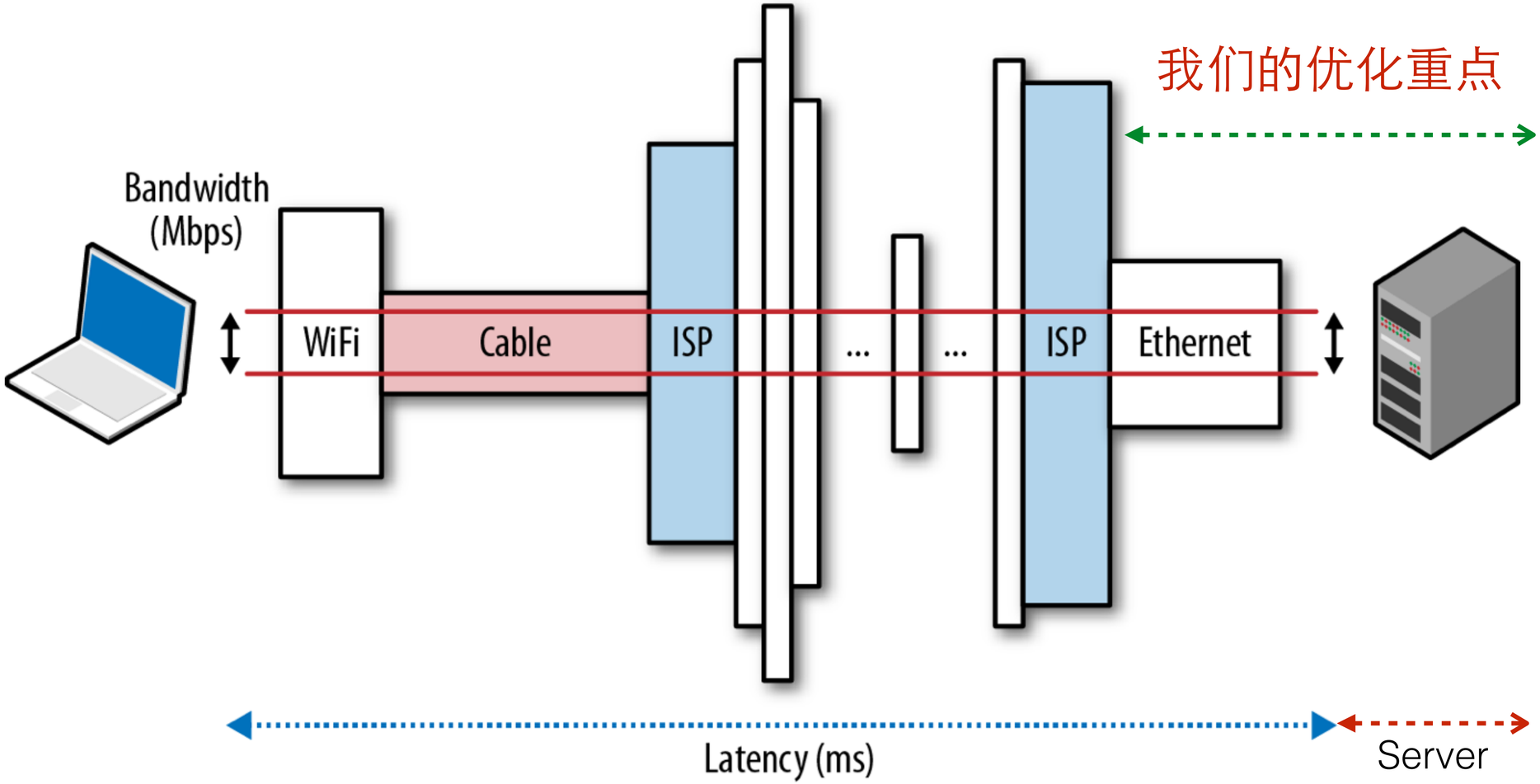



Photo credit: 《High Performance Browser Networking》

# 优化目标

- 用户体验：降低响应时间
- 节省资源：提高吞吐，重复利用资源

# 系统化视角

- 性能问题哪里来?
- 性能问题怎么定位?
- 怎么发现性能衰减? 事前、事后

# 系统化视角

- 性能问题哪里来?
  - 设计阶段：架构问题
  - 开发阶段：实现问题
  - 在线运行：子模块故障，流量上涨等

- 动态问题：时间因素

# 层次化视角

- 业务层
  - 代码逻辑
  - 算法层次
- 基础层
  - 运行层、交互层
  - 系统层次

# 怎么定位性能问题

- PIaaB: Performance Issue as a Bug :)

# 怎么定位性能问题

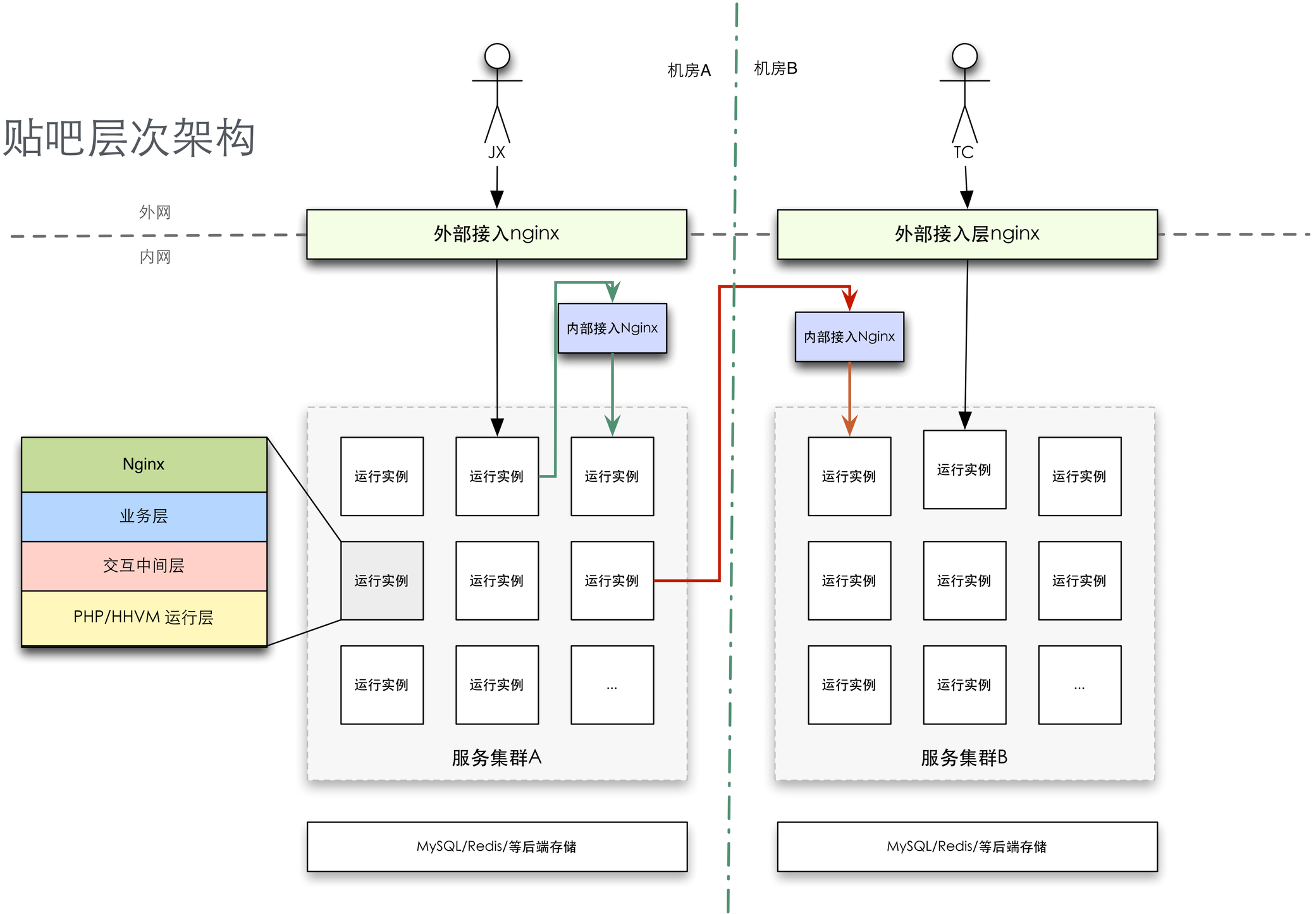
- Toolbox
  - xhprof、valgrind-callgrind、gperftools
  - jemalloc、HHVM AdminServer
  - systemtap… etc…
- 业务性能监控平台

# 百度贴吧

- 最大的中文社区

- 百亿级流量
- 业务增长飞速，规模大：100+子模块
- 业务120+ 上线/天

# 贴吧层次架构

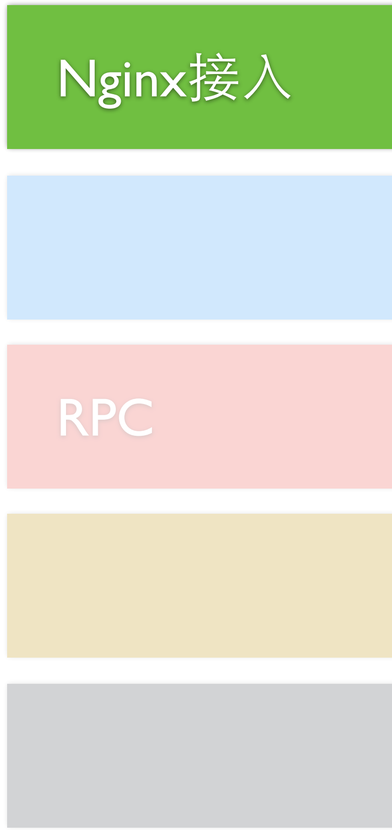

机房A
机房B
JX
TC
外网
内网
外部接入nginx
外部接入层nginx
内部接入Nginx
内部接入Nginx
Nginx
业务层
交互中间层
PHP/HHVM 运行层
运行实例
运行实例
运行实例
运行实例
运行实例
运行实例
运行实例
运行实例
运行实例
...
运行实例
运行实例
运行实例
运行实例
运行实例
运行实例
...
服务集群A
服务集群B
MySQL/Redis/等后端存储
MySQL/Redis/等后端存储

# Nginx优化

- Nginx Cache
  - 内部数据接口缓存
  - 及早返回，避免到PHP运行层，提升响应速度
  - Cache命中率： 60% 适合高成本接口

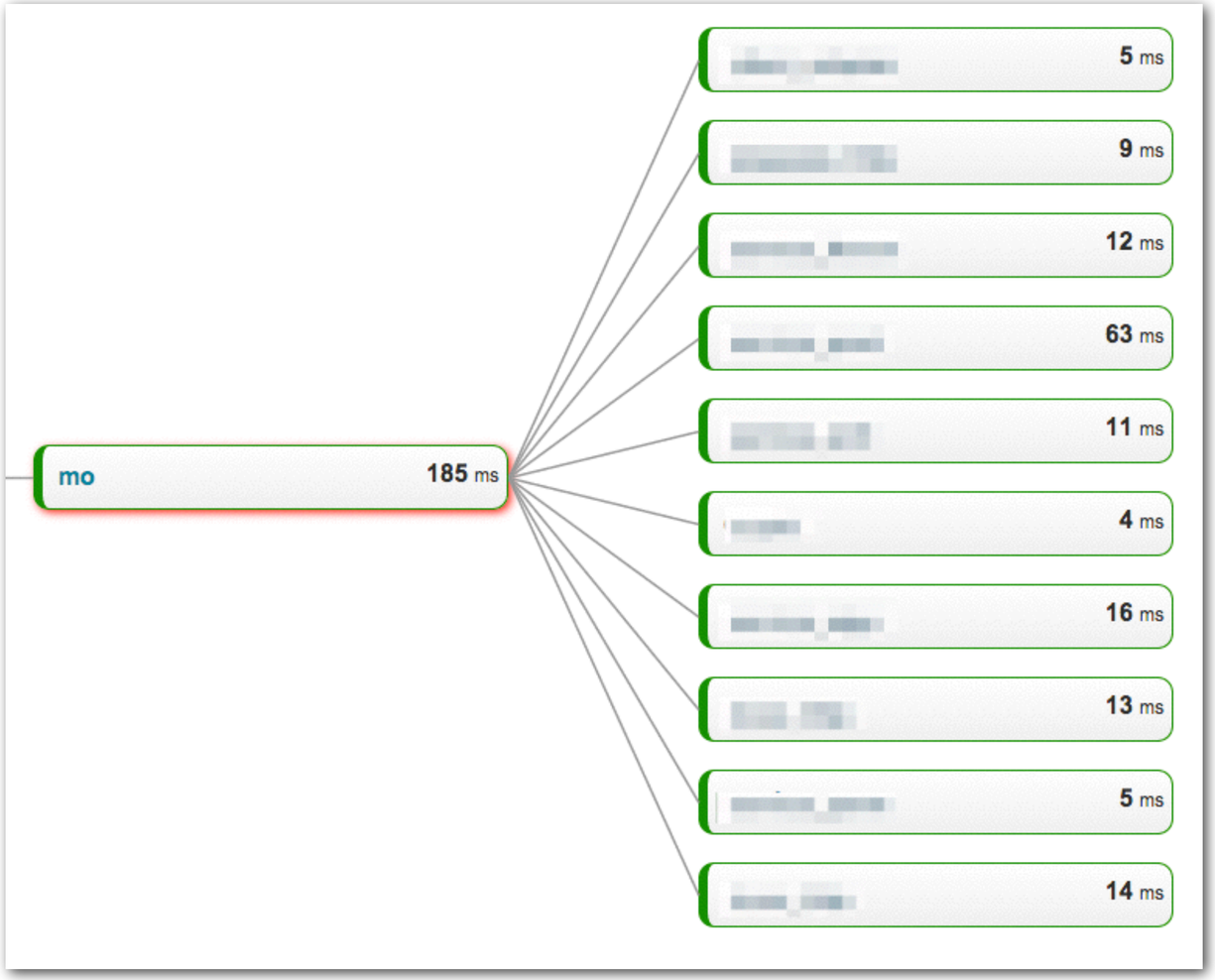
# Nginx优化

- 内部调用使用长连接
  - 跨机房请求 ~ 2ms左右，同机房1ms以内
  - 使用长连接避免连接性能损失
  - 级联效应



# 业务优化

- 并行化
  - 后端调用串行变并行
  - 最理想情况下：
    185ms -> 63ms




Nginx

业务层

RPC

# 业务优化

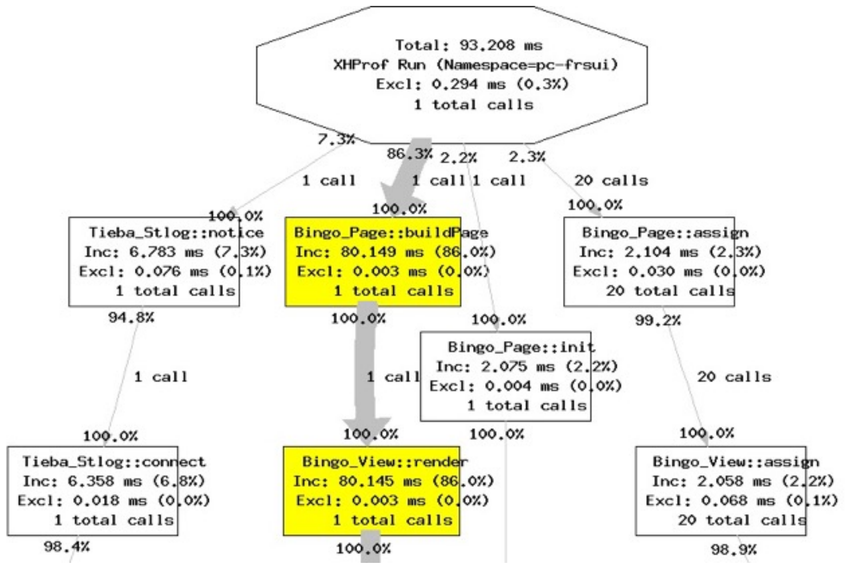
- 并行的矛盾：开发效率与效率的矛盾
  - 业务依赖关系，无法全并行
  - 模块化设计：可并行的难以并行
  - 自动化并行：开发效率无法满足

# 业务优化-代码及逻辑优化

- 代码优化：
  - 使用本地apc cache缓存配置
  - 逻辑优化

# 业务优化-代码及逻辑优化

- 前端模板性能优化
  - CPU密集型
  - 页面性能瓶颈



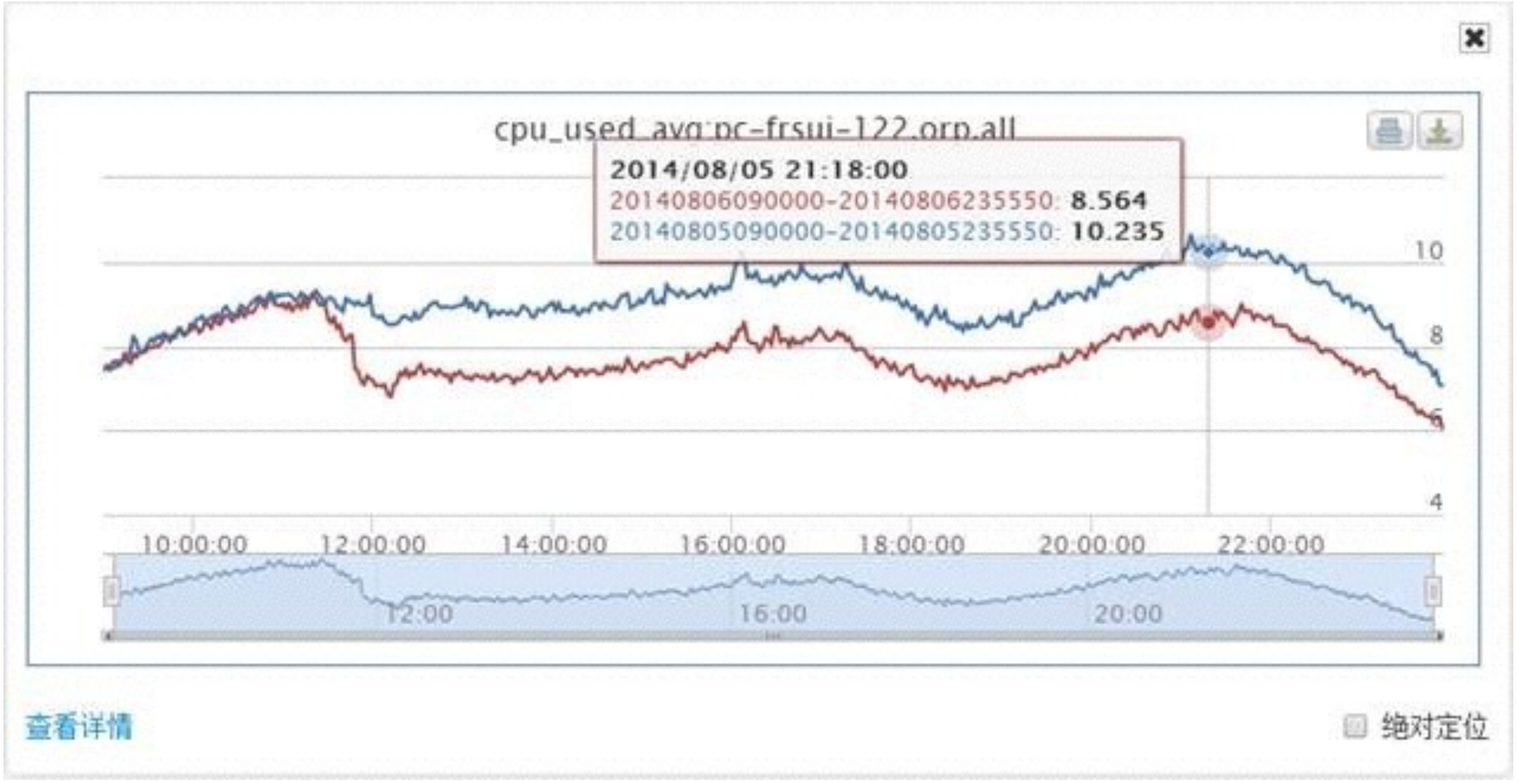

Nginx

业务层

RPC

# 业务优化-代码及逻辑优化

- 热点函数函数扩展实现
  - 性能提升17%
  - 响应时间降低 32.5%



# 业务优化-代码及逻辑优化

- Spider优化：帖子页 21%来自爬虫
  - 针对Spider提供简版页面：平均响应时间下降11%
- 带宽占用优化
  - 使用CDN，缩小页面体积
  - 节省 35%

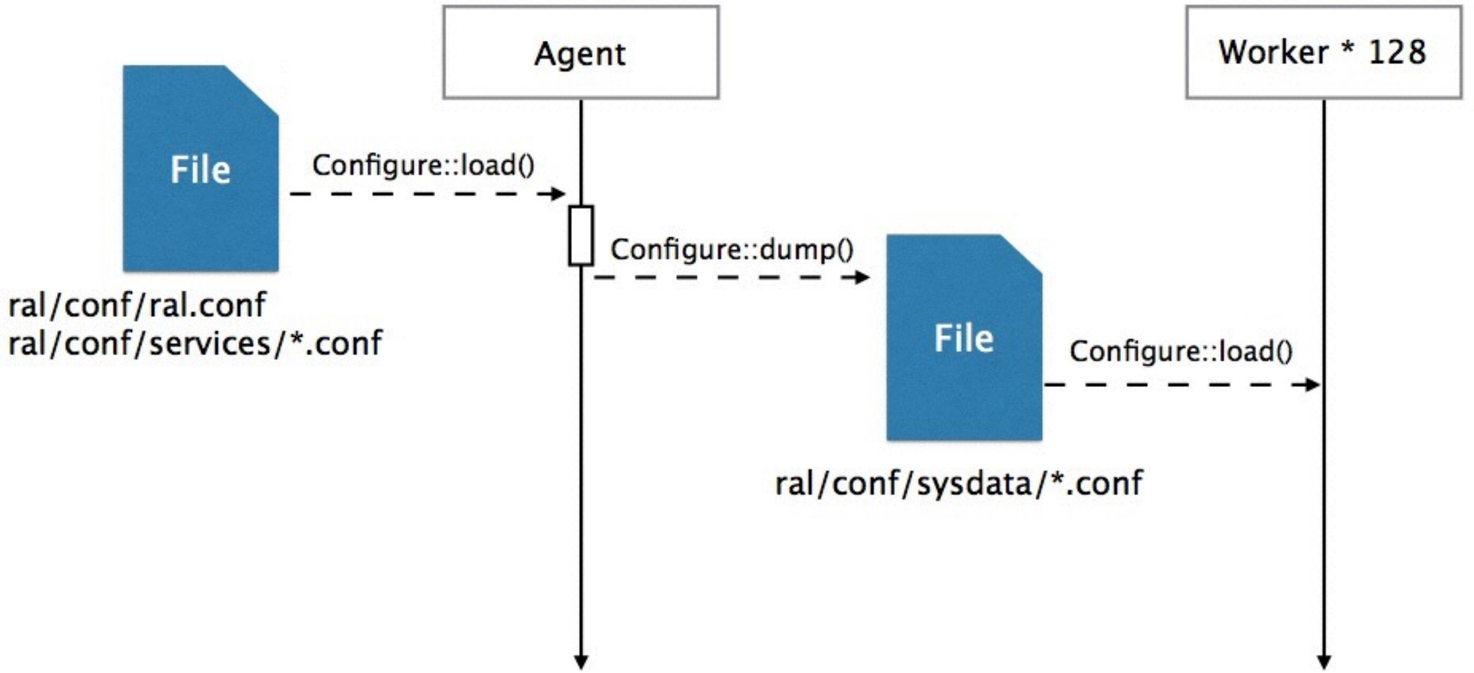
# RPC优化

- 700+后端配置
- 配置文件大小：~ 5M
- 变动频繁

- 实时加载配置不可行，热加载！

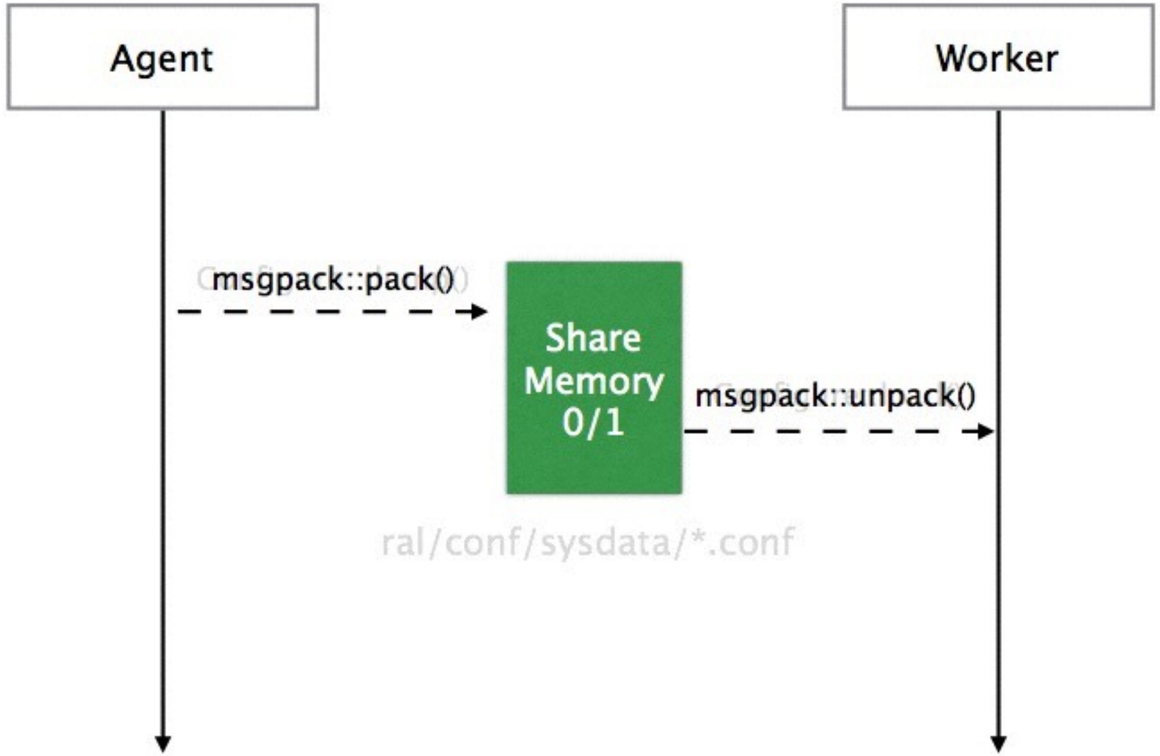
# RPC优化

- PHP的无状态
  fpm max_request
- 扩展实现
- 独立进程更新配置



# RPC优化

- 共享内存
- 使用更快的序列化格式



# RPC优化

| **序列化(s)** | case1 | case2 | case3 | case4 | case5 |
|---|---|---|---|---|---|
| PHP | 0.216778 | 0.217359 | 0.253435 | 0.162621 | 0.214441 |
| JSON | 0.249784 | 0.205442 | 0.325695 | 0.124034 | 0.384287 |
| Mcpack | 0.215553 | 0.425235 | 0.458352 | 0.129073 | 0.196122 |
| Msgpack | 0.218894 | 0.212432 | 0.250249 | 0.048074 | 0.096459 |

- 使用更快的序列化格式：msgpack > php serialize > json

# RPC优化

- 整体性能提升：24.6%

- 其他尝试思路：
  - 更短的调用路径，减少架构层次
  - 内部使用更高效的协议，而不是HTTP等

# 更快的PHP

- 另一个PHP实现：HHVM （HipHop PHP Virtual Machine)
  - Facebook开源项目 http://github.com/facebook/hhvm
  - 解决PHP运行效率问题
  - 开发速度和效率兼得

# 更快的PHP

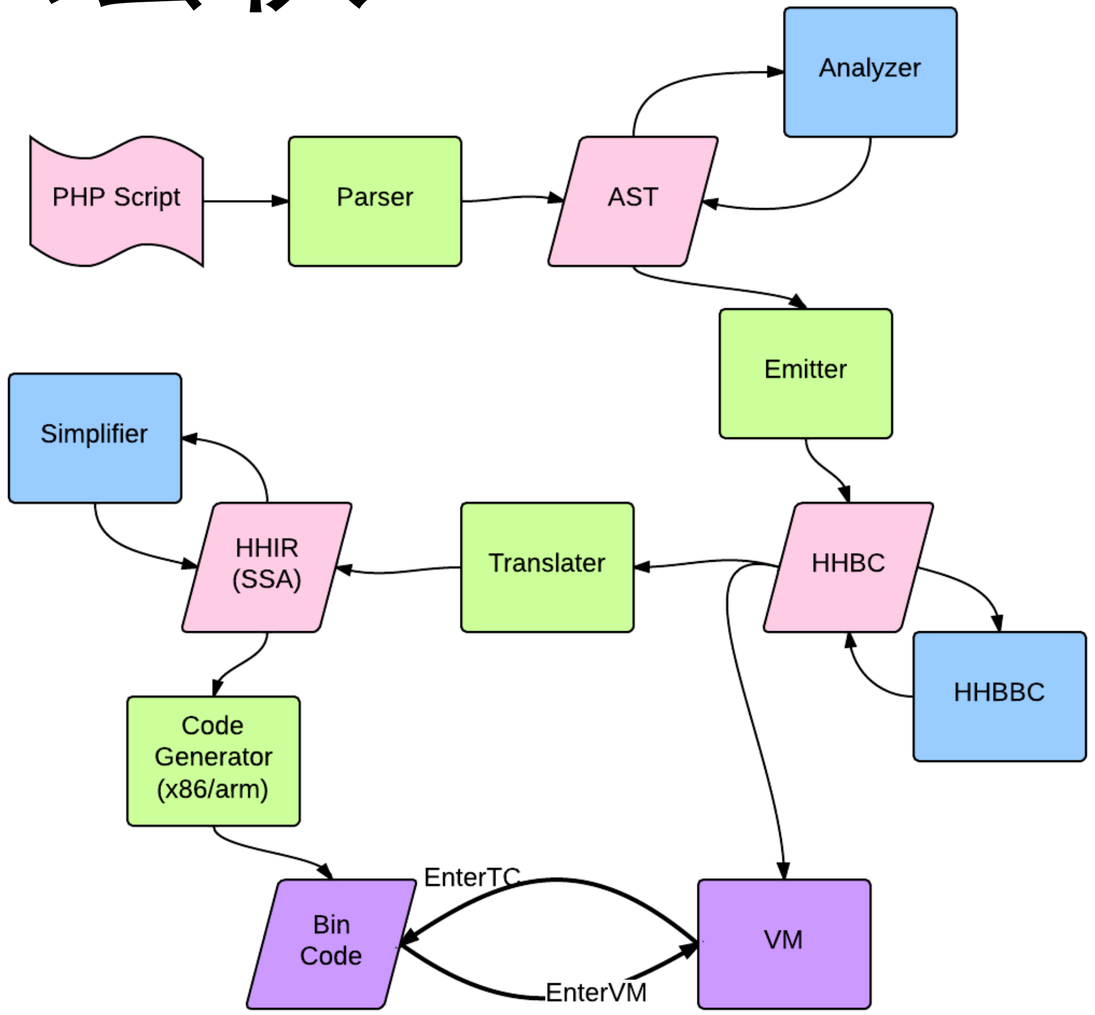
| | bench.php 耗时(s) | mico_bench.php耗时(s) | bench_third.php耗时(s) |
|---|---|---|---|
| PHP 5.2 | 6.692 | 41.890 | 9.226 |
| PHP 5.5 | 3.609 | 14.972 | 5.893 |
| PHPNg | 2.361 | 12.292 | - |
| HHVM 2.2 | 0.579 | 5.832 | 2.869 |

# 更快的PHP

| | 纯业务耗时 | idle | Vs 5.2提升比 |
|---|---|---|---|
| PHP 5.2 | 127ms | 50% | - |
| PHP 5.5 | 107ms | 45% | 15.7% |
| HHVM 2.2 | 72ms | 65% | 43.3% |

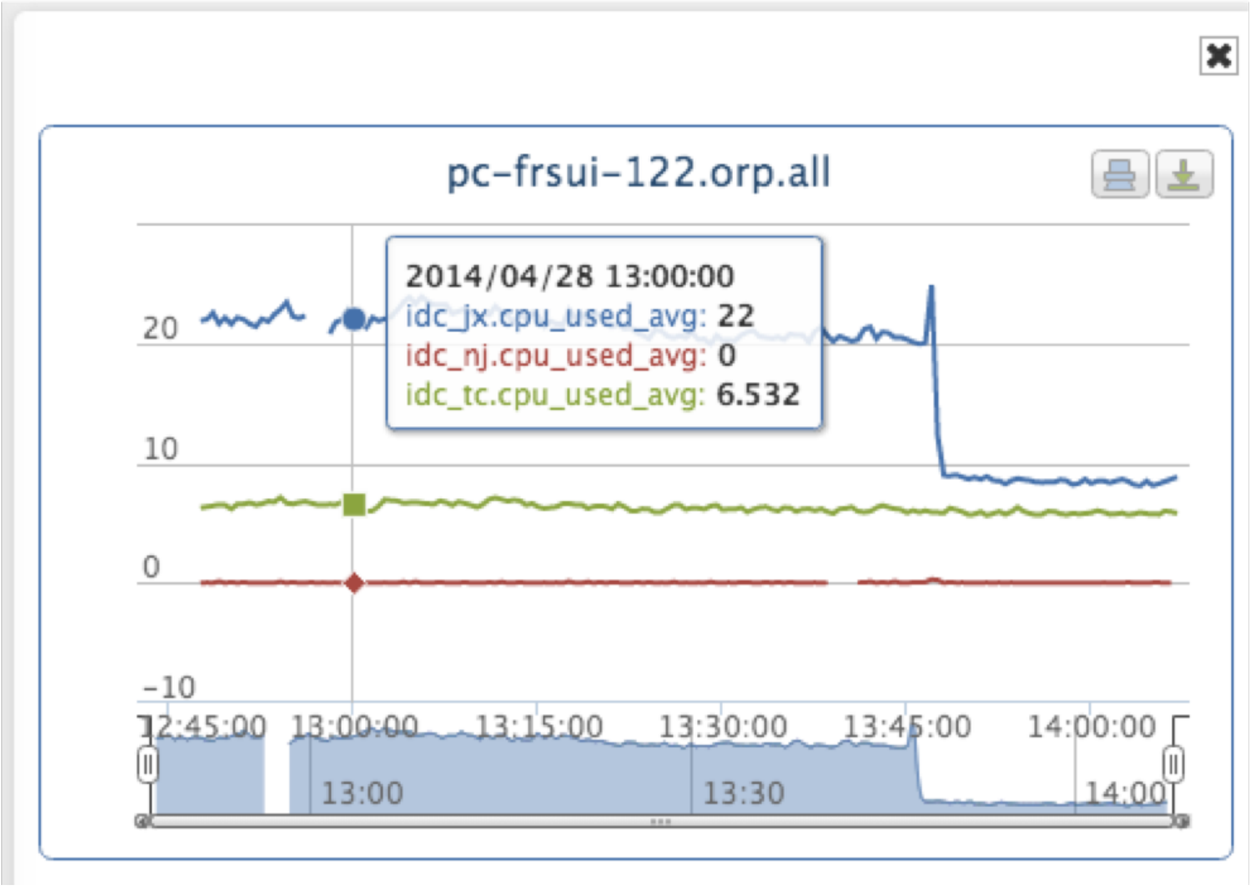
# 为什么会快

- 类型推导
- JIT
- 多级优化



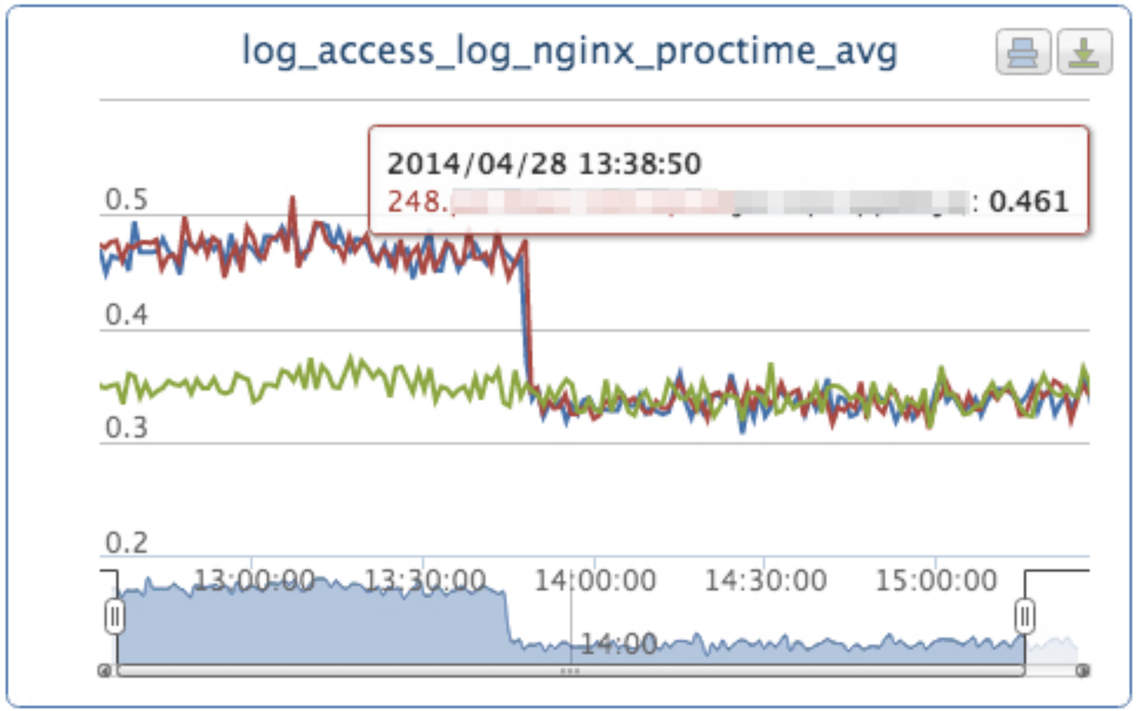
# 真实业务数据

- CPU占用率减少 11.9%
- 平均提升： 59%



# 真实业务数据

- 响应时间下降： 126ms
- 平均提升：27%



# 进展

- 完成绝大多数在线应用的迁移
- 服务性能平均提升41%+，最大69%

# 系统化思路

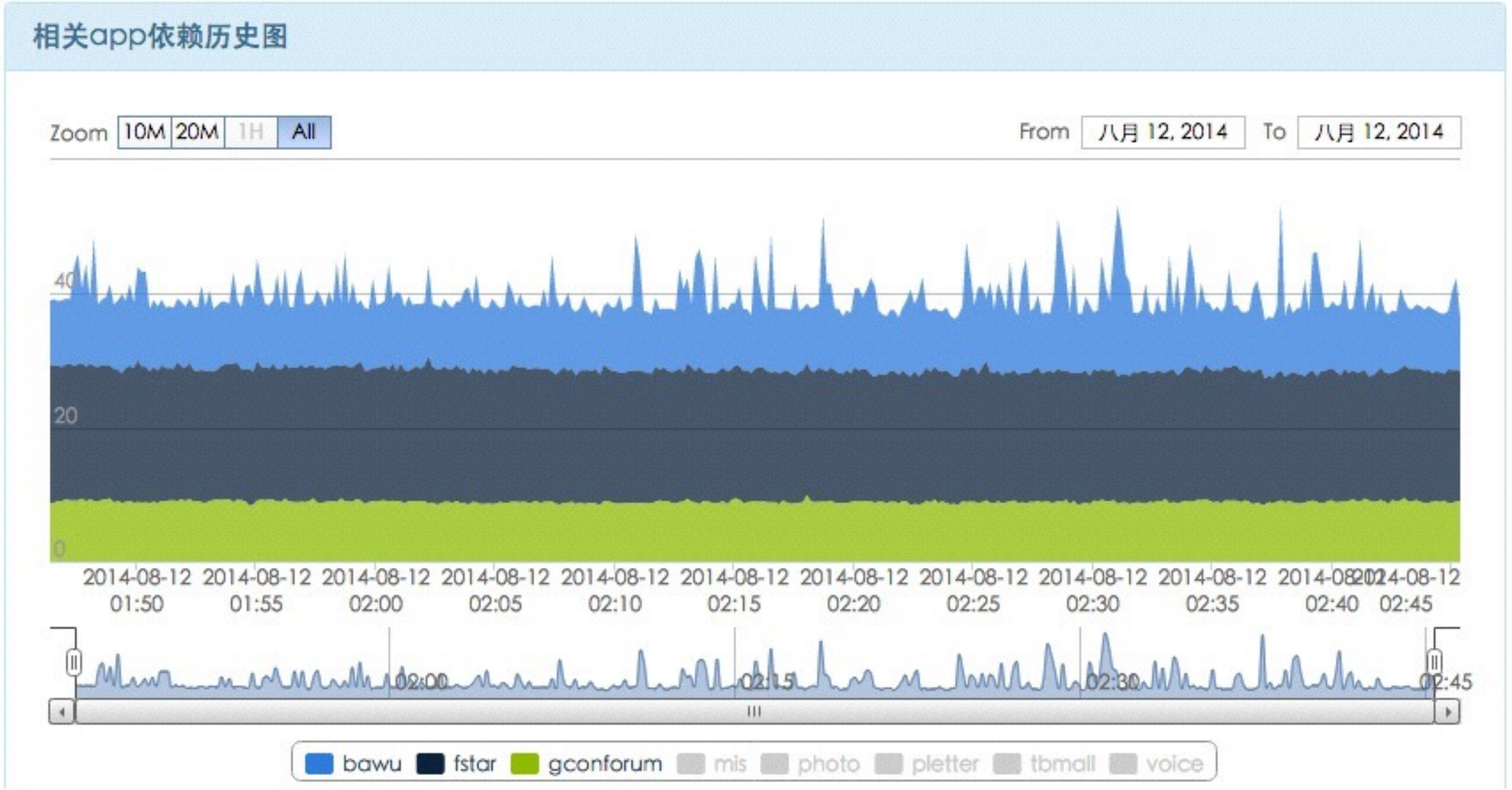
- **防治**：线下性能监控
- **监控**：线上性能监控
- **定位**：性能问题分析

# 系统化思路

- 防治：线下性能监控平台
  - 业务拓扑关系复杂
  - 线上线下环境差异大
  - 难以获得真实数据
- 难以自动化

# 性能问题早发现

- 在线性能监控平台
- 性能变化早知道



# 总结和展望

- 性能优化是个迭代过程
- 全局思考、有所取舍

- 优化在线性能问题定位效率
- 得出线下性能衰减有效手段

# Q&A