



光载无限监控体系的 变革与演进

欧曜玮 ou.yaowei@wifire.net

在开始之前



▶ 设备监控、资源监控、网络监控、服务监控、网站监控、业务监控、性能监控、应用性能管理 (APM)、应用程序监控、安全监控

















Application

Performance

Management

Mobile Real-User

Monitoring

Browser Real-

User Monitoring

Database

Monitoring

Server

Monitoring

Application

Analytics

Synthetic

Monitoring

AppDynamics

Platform

目录



- 过去的监控系统
- 我们为什么选择了 Open-Falcon
- ▶ 光载无限如何玩转 Open-Falcon
- ▶总结











过去的监控系统









过去的监控系统





- 难以扩展
- 二次开发难度高
- 多套系统,不同的操作方式
- 需要管理不同的配置,系统间同 步
- 难以保证监控全复盖







需求总结



- > 统一监控工具,降低维护以及排查的成本
 - 降低值班人员的学习成本与技术门槛,将解决问题的经验积累到系统上
- 提高监控系统的效能、可用性、用户体验
- ▶高可扩展性
 - 目标至少支持一万台服务器的规模
- 能满足内部的二次定制需求













寻找开源解决方案













有这么多的开源监控系统...



Zabbix, Nagios, Cacti, WhatsupGOLD, Sensu, Open-Falcon, Prometheus, Flapjack, Bonsun, OMD, Check_MK, graphite, collectd, Zenoss, monit, Munin, OpenNMS, Ganglia, Consul...







也有这么多的商业监控系统...



New Relic, Logical Monitor, DataDog, Splunk, 监控宝, Elastic-Marvel, Librato, Stackdriver...







我们如何选择开源项目?



- 现代化的设计与良好的架构
- 活跃的社区与大公司的支持
- 模块化、可扩展性高,易于二次开发
 - 可以很容易地利用既有的 Zabbix Script 或 Plugin
- 满足光载无限的业务需求
 - 网路质量、7x24 小时值班人员、等等











依照这标准,剩下...



- Sensu
 - 2011
 - Heavy-Water
- Prometheus
 - 2012
 - SoundCloud

- Consul
 - 2013
 - HasiCorp
- Open-Falcon
 - 2015
 - Xiaomi













我们为什么选择 Open-Falcon?



- 系统设计简洁,易于二次开发
- 分布式组件,水平扩展容易,支持量级高
- 高效数据展示与查询
- ▶ Golang 二进制部署方便
- 户完善的基础采集项
- 人性化的告警策略与模板设计

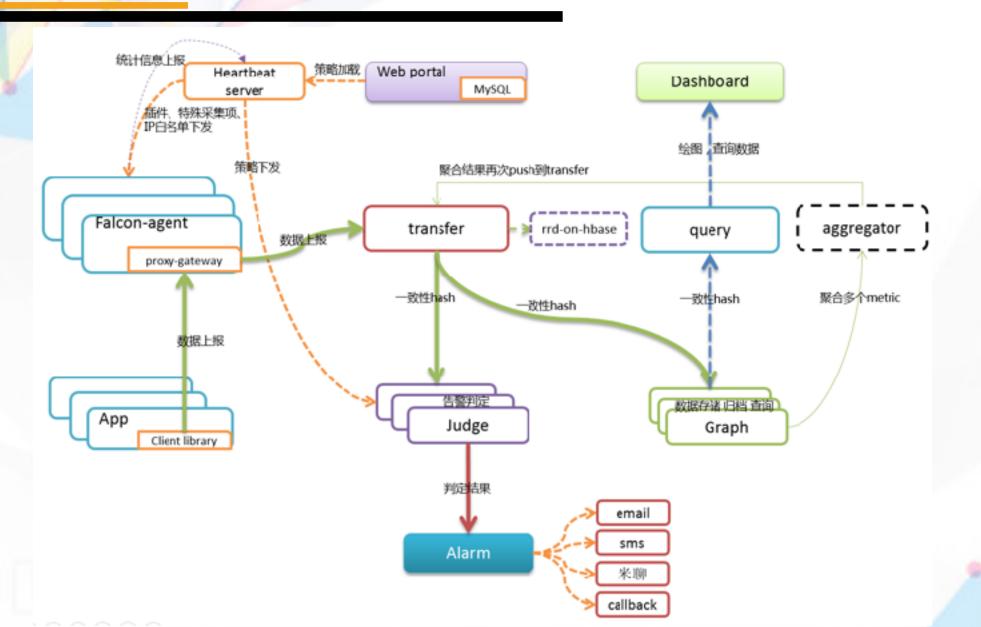


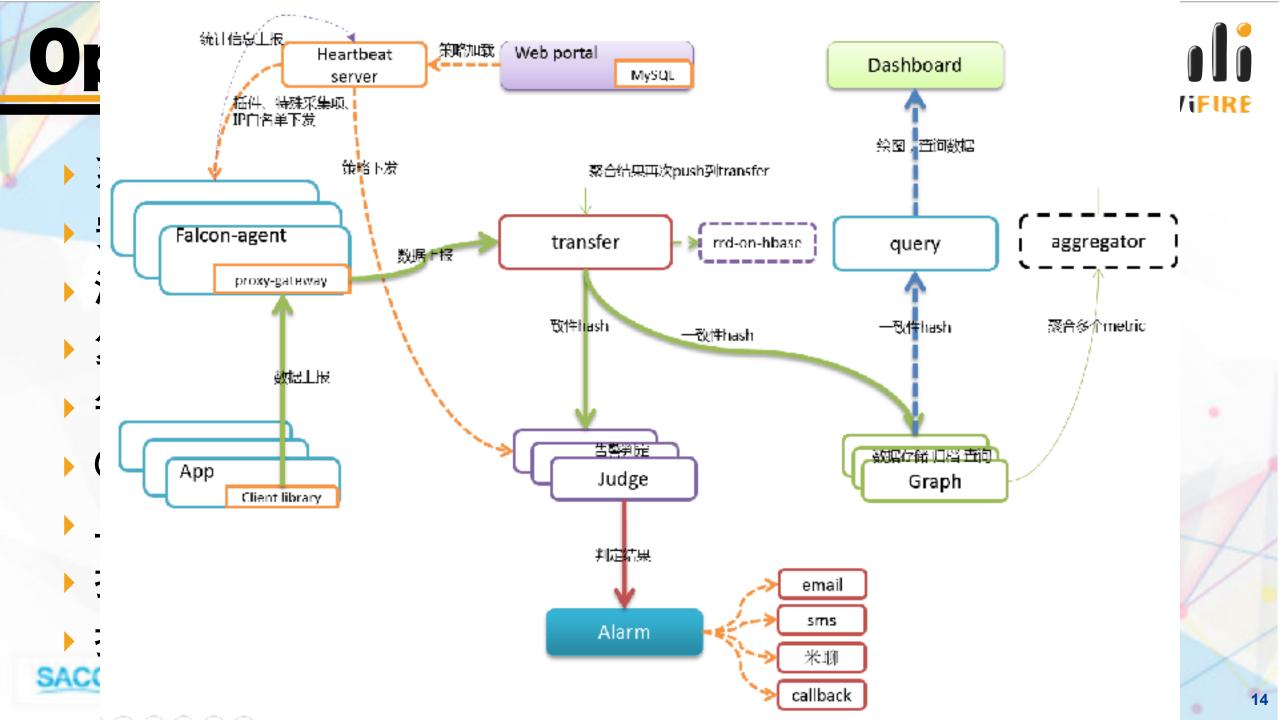




Open-Falcon 的问题



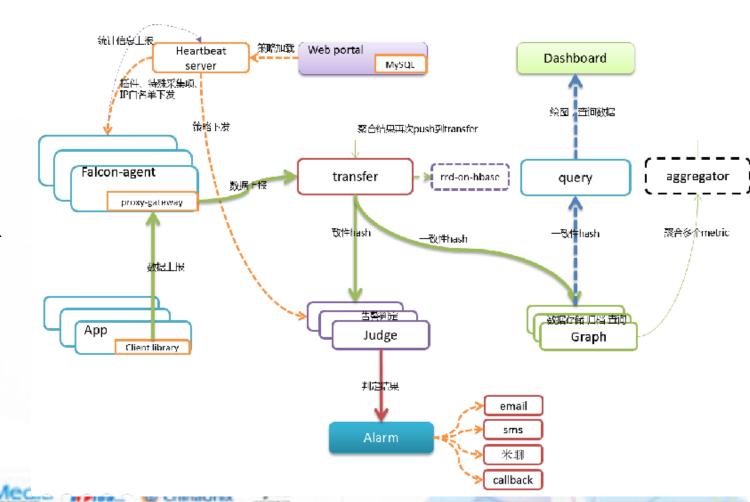




Open-Falcon 的问题



- > 系统不易分发
- 安全性考量不到位
- 没有通盘考虑的权限设计
- ▶ 复杂度稍高
- ▶ 每个 Graph 实例均是单点
- ▶ Graph 扩容有损
- ▶ 上下游组件没有 naming
- 报警没有入库
- 报警现场没有保存





二次开发 Open-Falcon









玩转 Open-Falcon



- Container/Monitoring as a Service
- 安全性补强
- 用户界面改善与体验优化
- ▶ CMDB 与 Open-Falcon 结合
- 运维内部接口适配
- 后端模块统一与简化
- ▶ CDN 网络质量监测





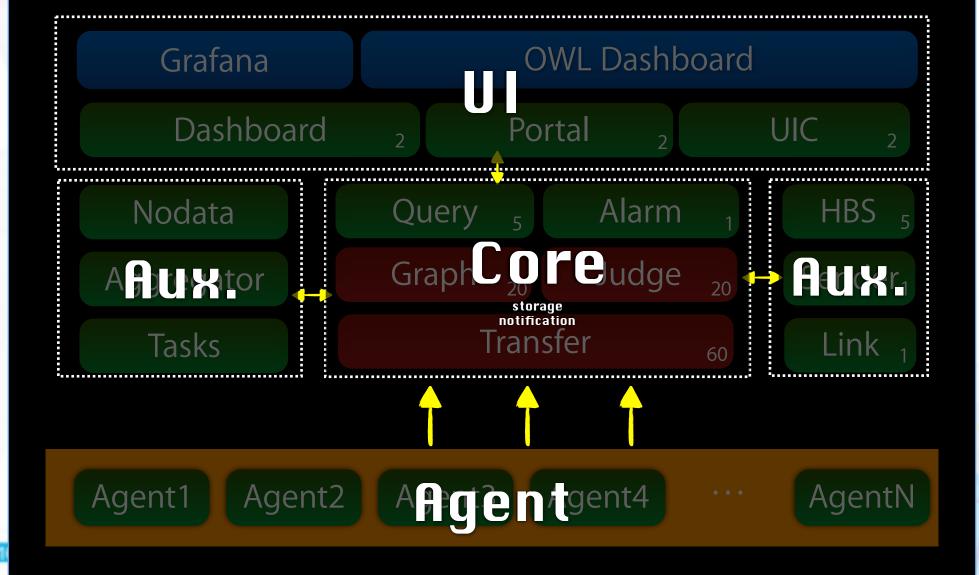


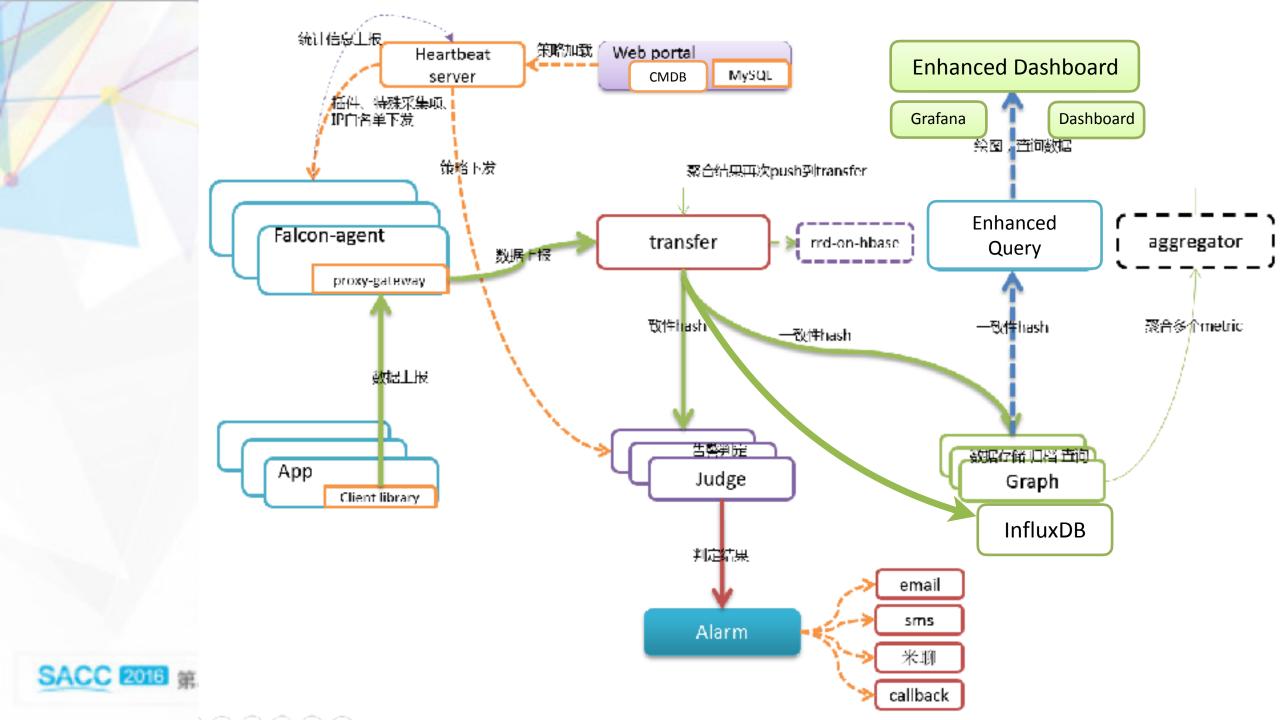




我们是这么看待 Open-Falcon 的







Container as a Service



Open-Falcon 所有组件容器化

[root@		400.00	DockerCompose-Ope Containers
Name	Command	State	
graph	./run.sh	Up	0.0.0.0:6070->66/w/ccp, w.w.w.w.w.u.z->ow/1/ccp
hbs	./run.sh	Up	0.0.0.0:6030->6030/tcp, 0.0.0.0:6031->6031/tcp
judge	./run.sh	Up	0.0.0.0:6080->6080/tcp, 0.0.0.0:6081->6081/tcp
query	./run.sh	Up	0.0.0.0:9966->9966/tcp
			A A A A COCA CACALL A A A A A A A A A A A A







.0.0:6060->6060/tcp,

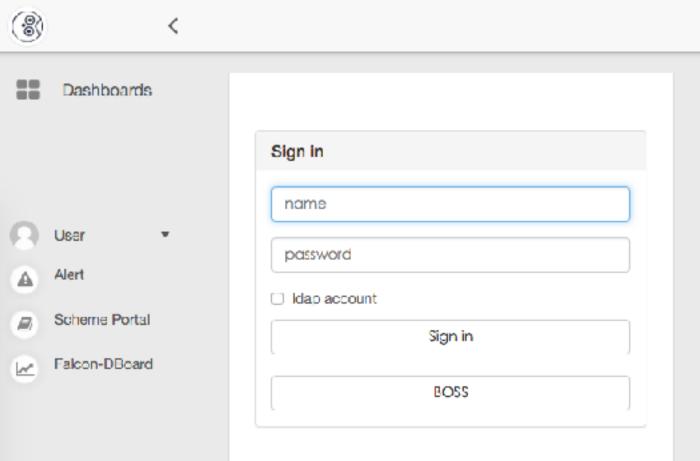


安全性补强



- ▶ SSO 单一登入
- Grafana 与 Open-Falcon的账号整合









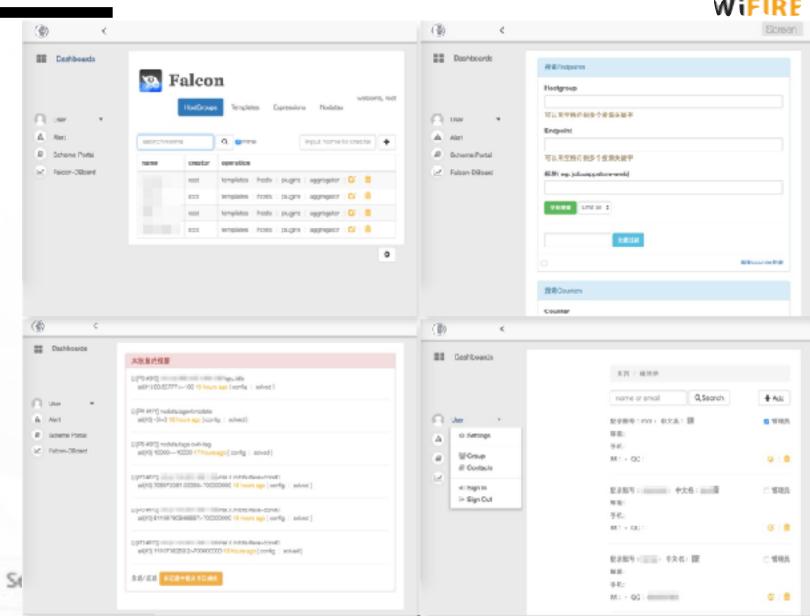




用户界面改善与体验优化(1/4)



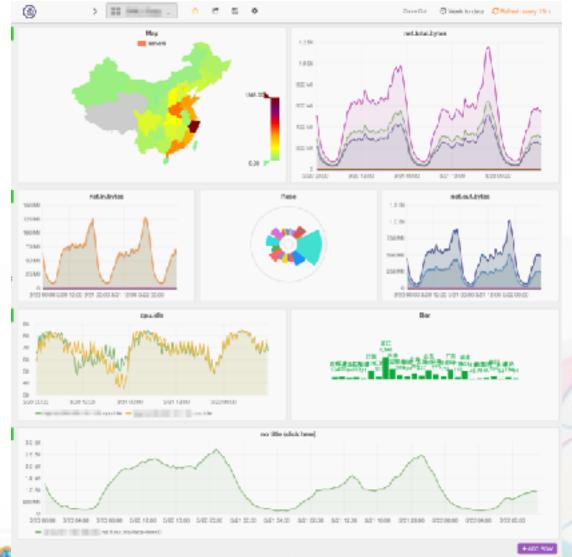
- ▶ 统一入口不再迷失
- ▶ 加入侧边栏、导览列



用户界面改善与体验优化(1/4)



- 探索趋势图以外的可能
- Marafana 与 Open-Falcon 的 深度整合





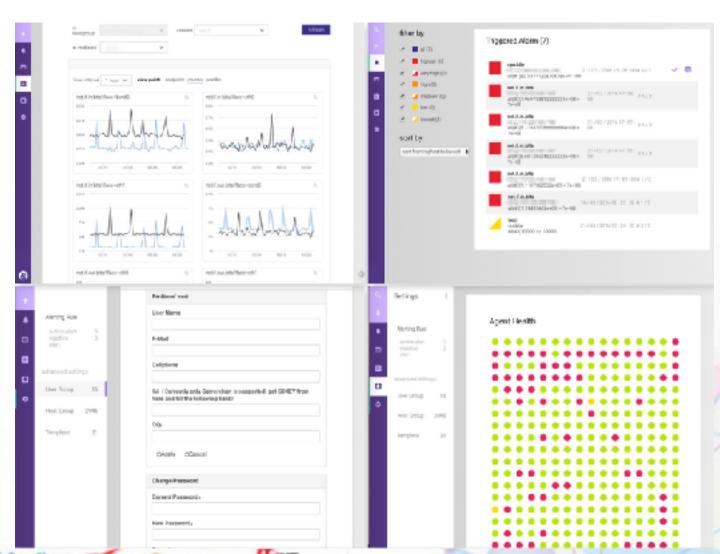




用户界面改善与体验优化(3/4)



- Isomorphic Web Design
- 单页式设计
- 服务端渲染
- 前端组件统一



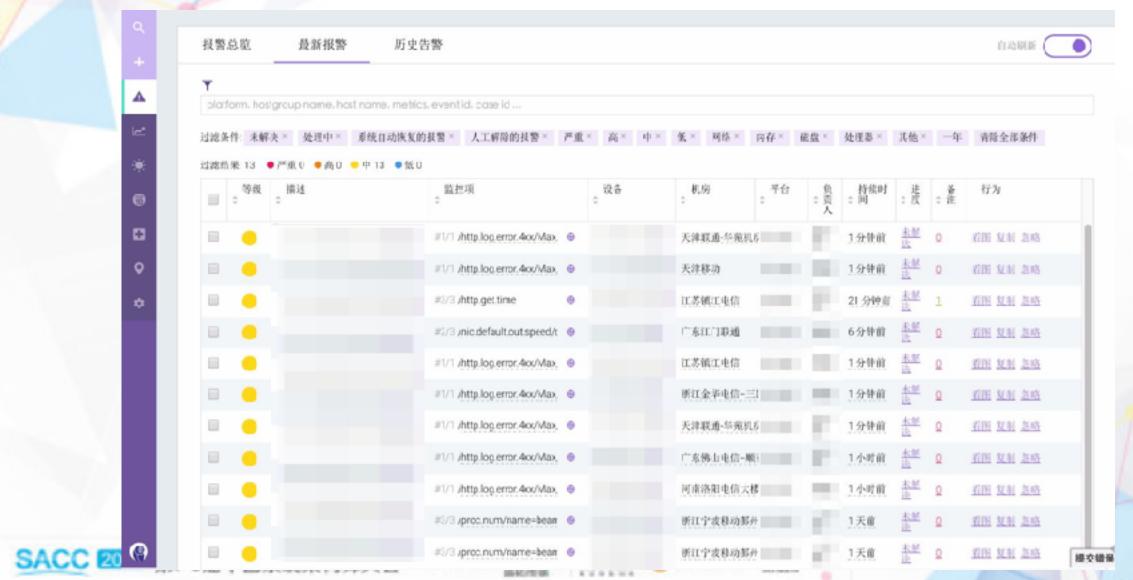






用户界面改善与体验优化(4/4)





CMDB 整合 (1/2)



- > 支持以内部 BOSS 账号登入
- 设备列表跳转至概览页面
- 资产信息同步









CMDB 整合 (2/2)



▶ CMDB 统一管理服务与角色,专用接口替换模板变量

平台		平台服务器角色	ES	P&MD	备注
	v	CDN边缘节点 ‡	请选择 ✓ fastcache nginx	4	
	v	CDN边缘节点 \$	squid fastlive fastmedia		•
	*	CDN边缘节点 ‡	powza	‡	
■ 添加服务绑定					_



运维内部接口适配



- ▶ 协助运维更快上手
 - Screen, Template, HostGroup 的初始范本
 - Zabbix 脚本移植、提供插件范本
- Agent 安装流程与配置简化
 - 如:RPM 包安装、/etc/init_d 启动脚本
- 支持 Zabbix API 接口
- ▶ 支持以 HostGroup 为单位查找,设备同步自 CMDB





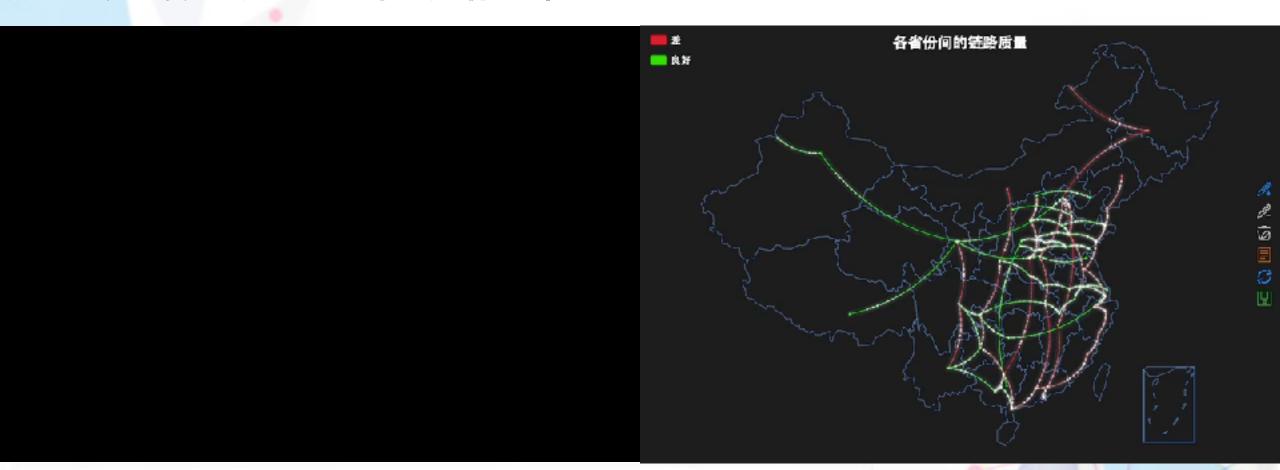




网路质量监测



点对点的网络指标数据采集









网路质量监测的挑战



- 多维度
 - 机房、省份、城市、地区、运营商
- 批次计算
 - 平均值、最大/最小值、标准差
- ▶ 不同的分组:
 - 内网、外网、平台
- 向量
 - 方向与大小



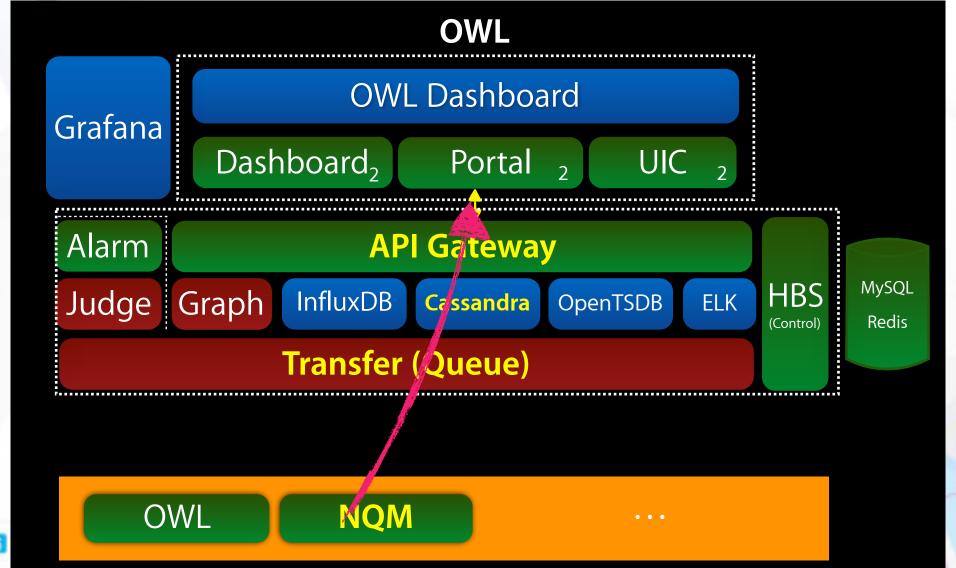






架构上的调整



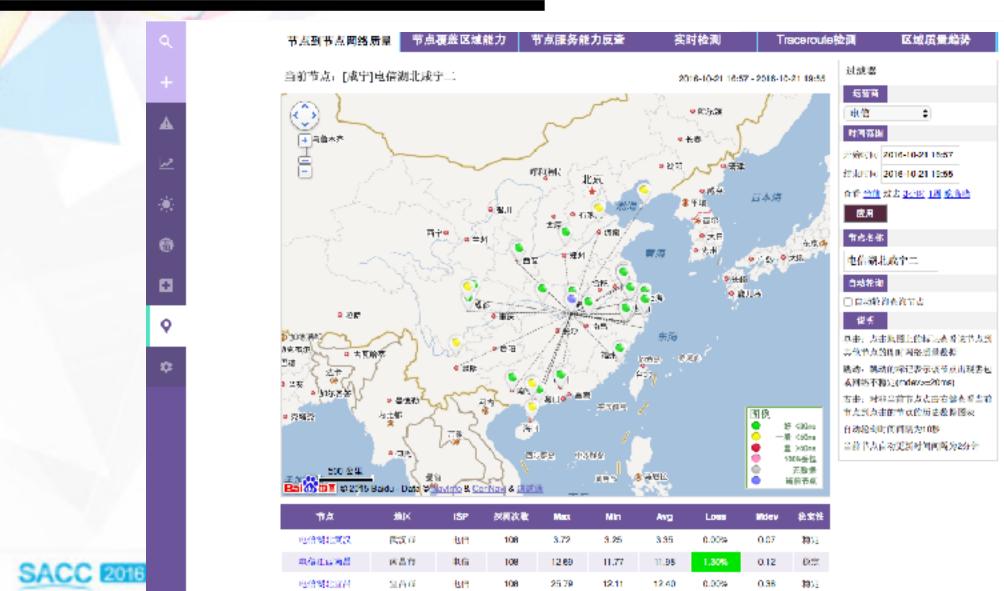


网路质量监测(1/3)

电信狂热情红华教训

独江市



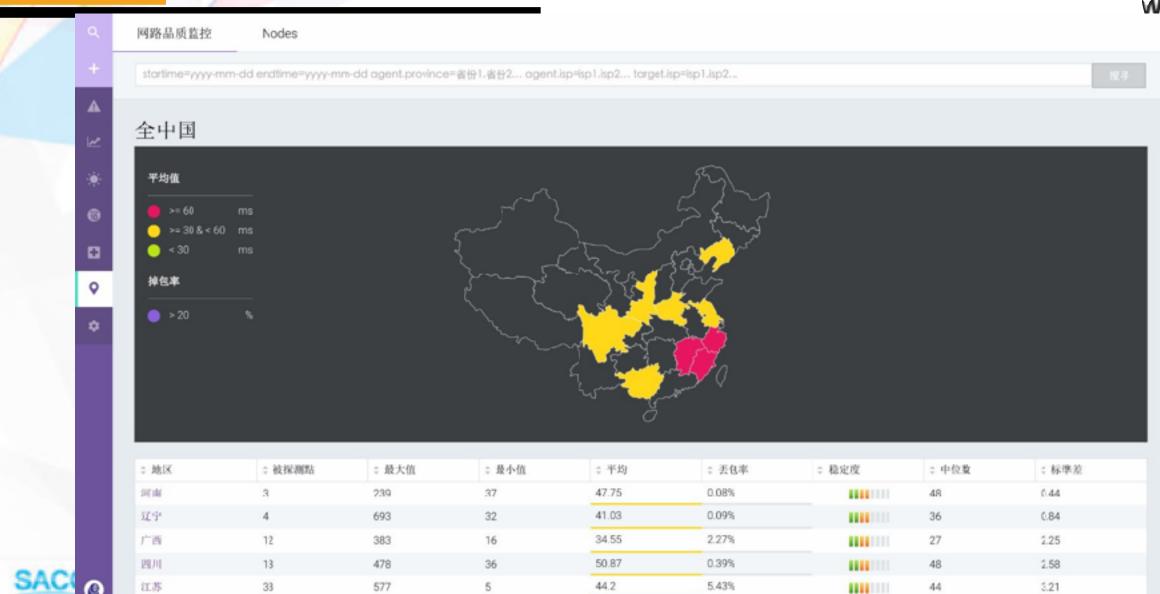


15.04

0.17

网路质量监测(2/3)





后端模块统一与简化(1/2)



- ▶ 借鉴 Consol 与 Docker-Compose
- 简化服务端维护与部署的负担
 - 配置档统一管理
 - 打造周边工具链
 - docker, vagrant, test, scripts, make
- 集中管理,聚人气
 - 代码、文档、问题、反馈、PR
 - ★399 agent (~2016/04/09)
 - ★262 of-release (~2016/04/09)



□ README.md

Documentations

- http://book.open-falcon.org
- http://docs.openfalcom.apiery.io.

Get Started

git clone https://github.com/Cepave/open-falcon-backend.git cd open-falcon-backend git submodule update --init

Compilation

all modules make all # specified module make agent

Run Open-Falcon Commands

Agent for example:

./open-falcon agent [build|pack|start|stop|restart|status|tail]







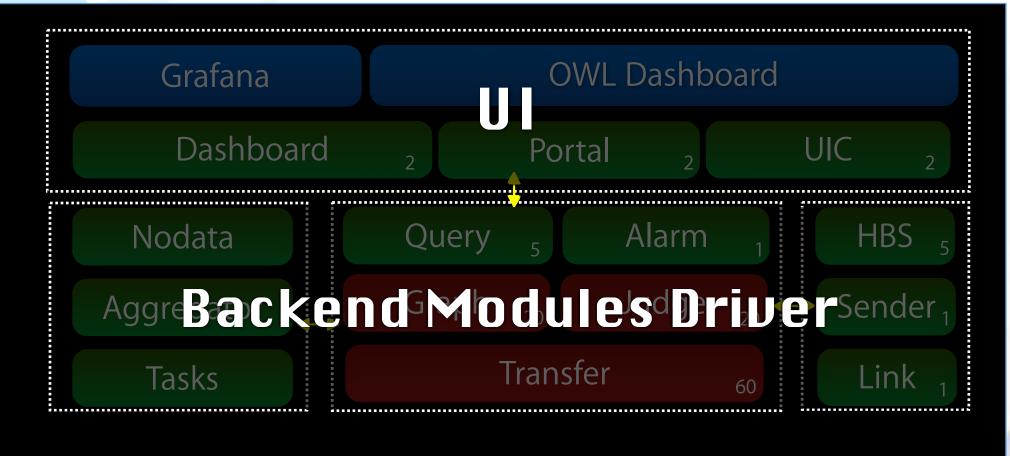


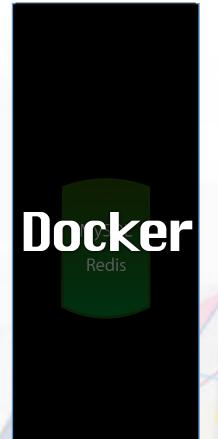




后端模块统一与简化(2/2)















总结







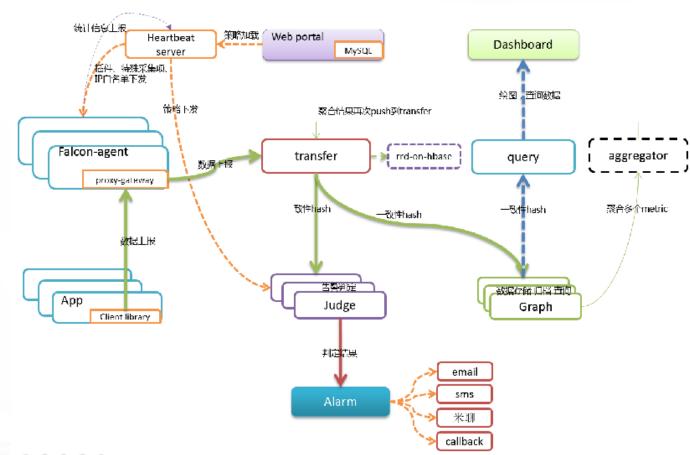




回顾 Open-Falcon 的问题



- > 系统不易分发
- 安全性考量不到位
- 没有通盘考虑的权限设计
- ▶ 复杂度稍高
- ▶ 每个 Graph 实例均是单点
- ▶ Graph 扩容有损
- ▶ 上下游组件没有 naming
- 报警没有入库
- 报警现场没有保存



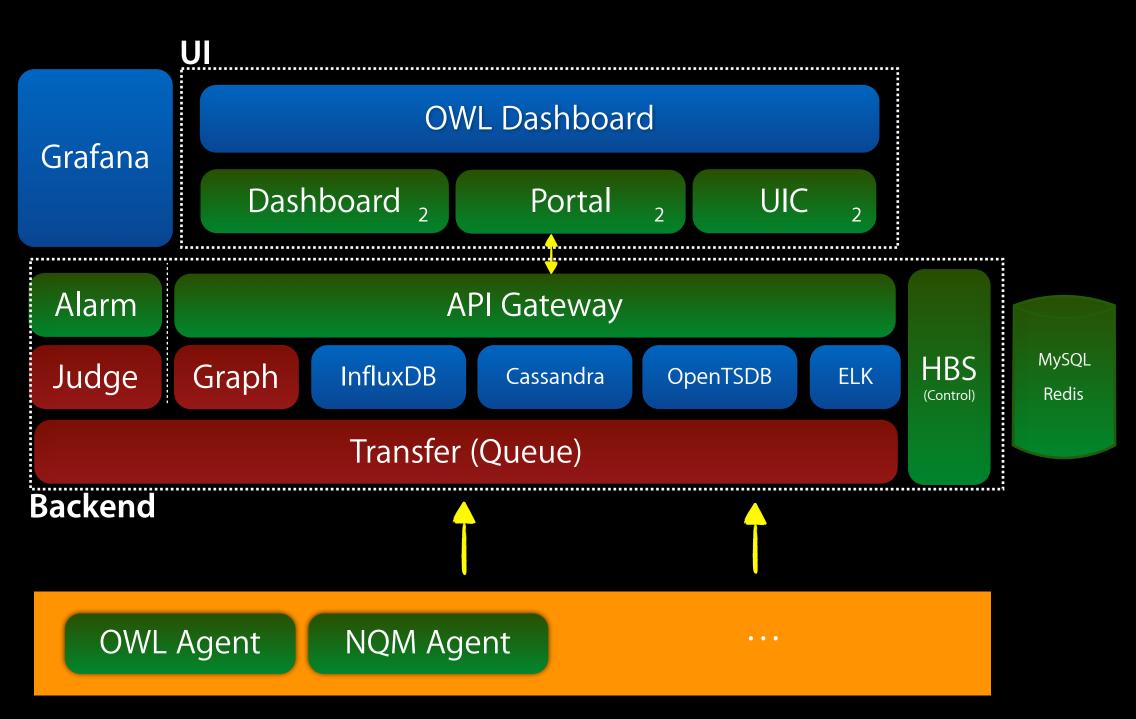














社区发展



