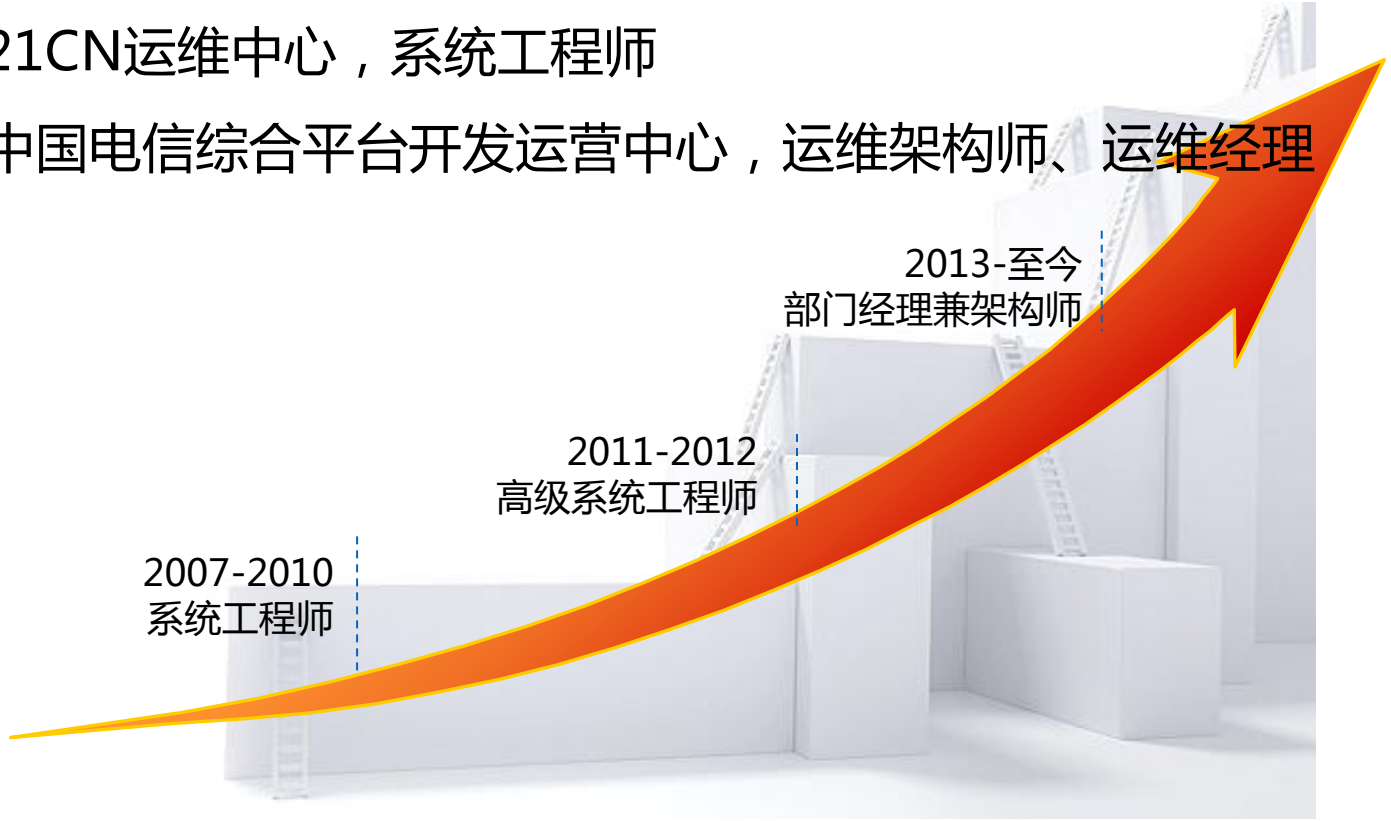
电信亿级互联网产品的运维实践

仇国祥
中国电信综合平台开发运营中心
运维经理

– 2009年加入21CN运维中心，系统工程师

– 2013年加入中国电信综合平台开发运营中心，运维架构师、运维经理

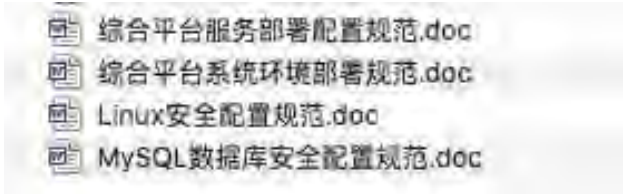


质量 效率 成本 安全

- 各公司阶段不同，技术能力不同，手段不同，但是运维目标是一致的
- 质量，产品的灵魂，运维工程师首要任务，保证服务稳定和较好性能
- 聪明的运维懂得提高效率，释放自己，要有目标，但也不用盲目攀比，用脚本、还是界面化平台，阶段不同顺手即可
- 控制成本
- 安全不松懈，出现重大安全问题，功不抵过（安全规范、扫描、渗透、WAF）

减少配置差异

- 基线安全规范
- 系统规范
- 部署规范
- 日记规范
- 路径规范
- 持续交付规范



持续交付的标准化-运维

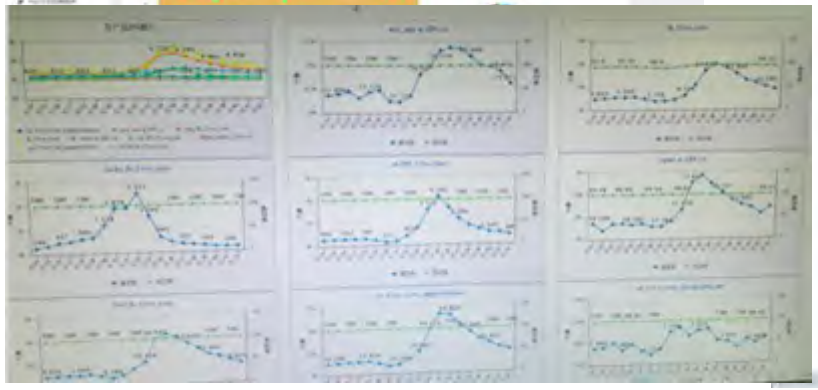
- 中间件配置标准化
  - 名字服务管理（DNS/HTTP）
  - 系统配置批量变更（内核/limits/磁盘配额等）
  - 常规Nginx配置web化编辑（更新和下发）
  - Java容器配置标准化（工程配置/性能相关参数/数据源）
  - 资源量化分配（Mesos/Marathon）
- 流程标准化
  - 制订持续集成交付流程（svn->开发->测试->灰度->生产）
  - 配置变更流程（变更->审核（仅灰度/生产环境）->下发）
  - Marathon/监控事件处理流程

持续交付的标准化-研发

- 代码工程的标准化
  - 工程名唯一（容易识别）
  - 代码目录（固定，分类）
  - 日志输出（本地转网络，涉及logstash/ceph/flume-ng）
- 应用的标准化
  - XML配置（避免混乱）
  - 应用解耦，无状态化（减少跨工程访问底层数据存储）
  - 第三方依赖（分布式协调/缓存/数据库）
  - docker构建（jenkins+maven，一次构建，多处运行）

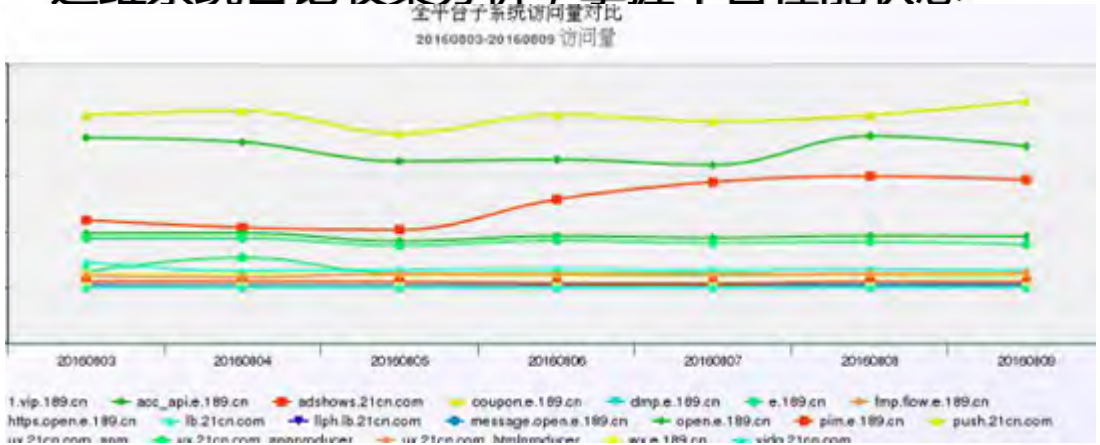
运维专业化

- 优化监控系统（nagios zabbix cacti 业务拨测），监控收敛、日记收集、大数据、性能分析
- 避免人为故障
- 容量评估，限流
- 运维参与架构， 动静态分离 、业务耦合合理分离 、内外部接口分离 集群化、无状态化
- 流程规范

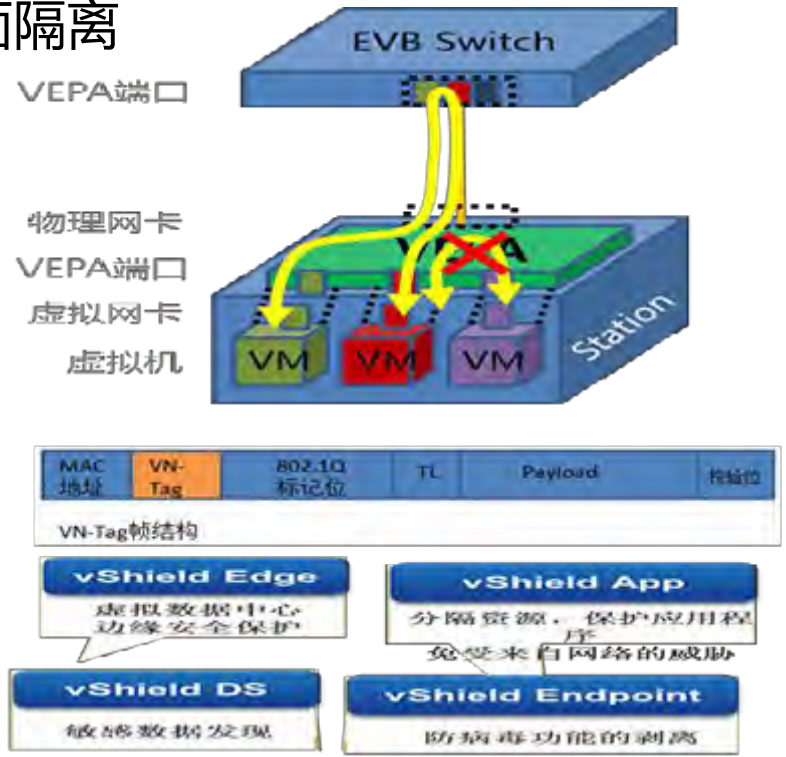


数据-业务和运维数据分析，主要通过flume采集日志

- 业务数据基于Mysql、MyCat分库实现分布式存储
- 对于有事务要求的交易数据基于Atalas实现读写分离
- 运维系统日记收集分析，掌握平台性能状态



网络隔离-基于VEPA的802.1Qbg，通过Vswitch的VLAN，访问策略控制保证网络层面隔离

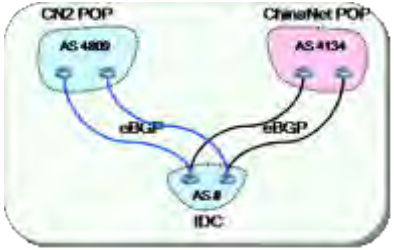


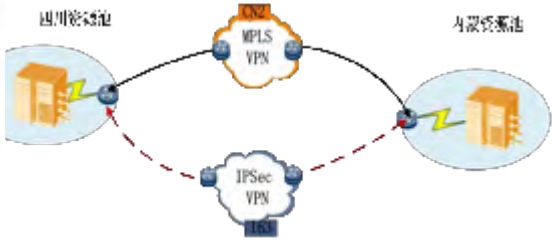
从容灾备份向多活过度

- 任重道远
- 没有演练过的备份是无效的
- 网站、认证方面的服务相对容易实现
- 交易类服务，对架构的要求非常高，有较大复杂度

南北互通和加速-整体上大网越来越好，没有想象的那么差

- CDN
- BGP
- 多节点
- DNS分流

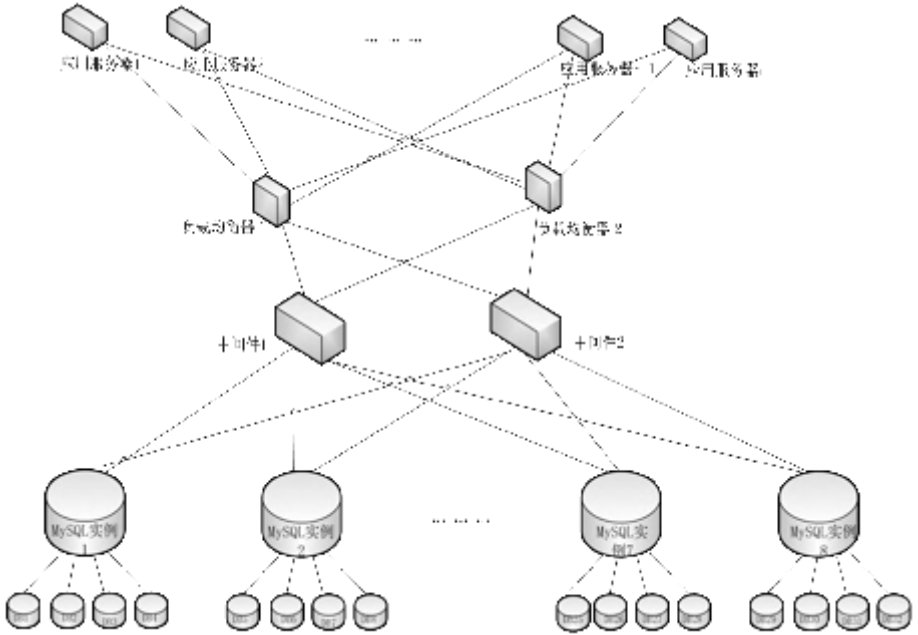




长时间对比测试，专线的稳定性和延时还是有优势

分布式应用：对开源方案胆大心细

- Ceph
- 分布式数据库
- OpenStack



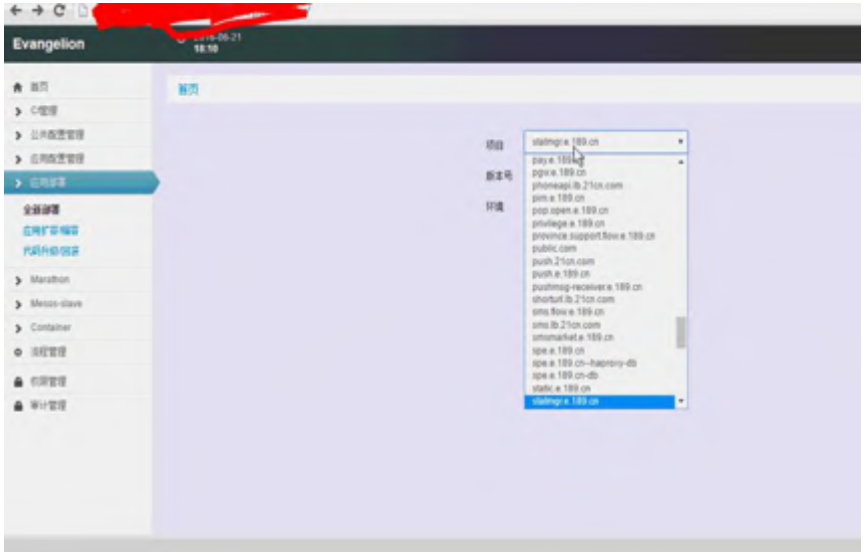
高效使用Nosql，减少数据库压力

- Codis
- Redis
- Memcached
- 选型、高可用
- 实例大小、持久化

- memcache、redis等
  - key-value存储，简单易用，易扩展
  - 物理上分布式存储，逻辑上集中单点
- HBase、Cassandra、MongoDB
  - 支持数据自动复制，使用较复杂
  - 适用于PB级以上数据的存储

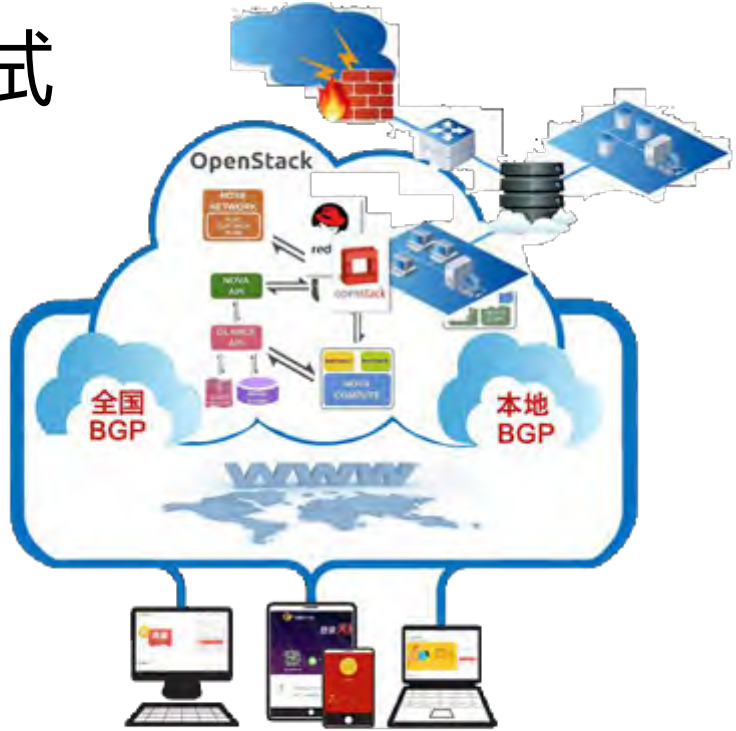
自动化尝试-由通过脚步自动方便地进行软件配置和运维，向使用高效易用的运维自动化平台过度

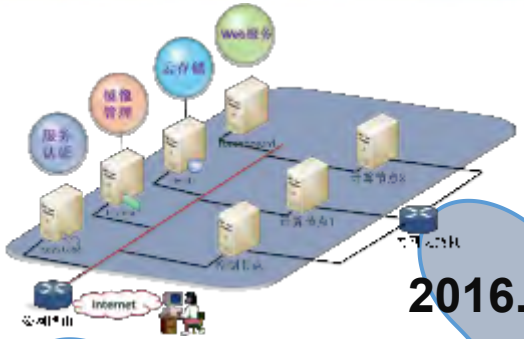
- Python+django
- Mesos、marathon框架
- 结合saltstack、docker技术
- Cmdb配置库
- 存储集群ceph
- 配置统一下发
- 代码持续交付
- 故障自愈？



基于openstack的私有云尝试

- Nova
- Glance
- Nova-network
- Ceph
- Swift





2016.01

项目基础设施建设启动，完成网络设备对接、服务器分布规划、服务器管理、磁盘分区等管理配置

2016.02

项目基础建设完成，完成存储搭建、完成openstack+kvm私有云架构搭建,顺利开出虚拟机、虚拟机网络对接

2016.03

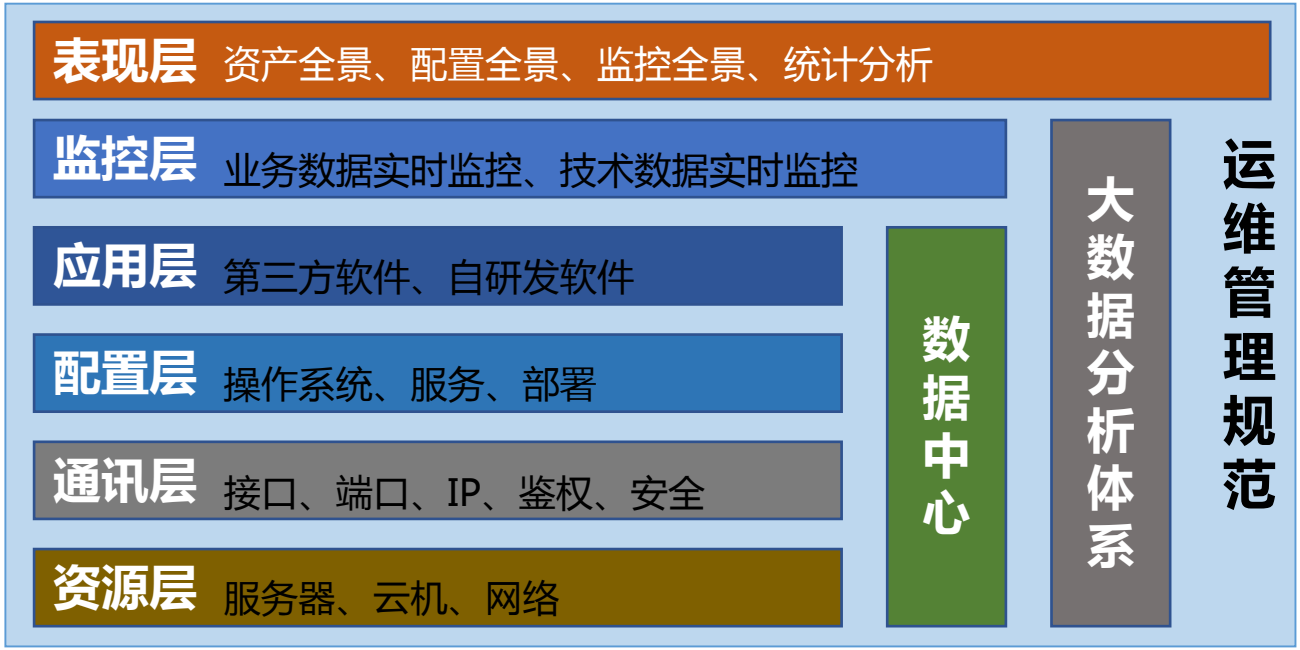
异网互联互通正式承载生产业务上线运营、整体运营质量数据分析及优化、部分核心业务迁移

2016.06

扩容优化、网络安全加固升级、业务迁移

– 有规划的保障体系

– 2015年规划全平台保障体系，目前已完成部分平台的建设及优化



资产管理系统

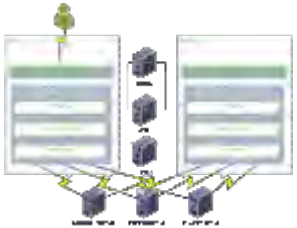
- 提供**云资产**和**物理资产**的集中管理；
- 符合公司的个性化定制和管理规范；
- 灵活的资产模板动态设置；
- 通过自定义公式，统计资产分布、成本详情。

通讯总线系统

- 基于lvs+nginx+lua架构改造；
- 单机每秒处理的并发请求可达**1万**以上，平均耗时**50毫秒**以内，系统资源占用少；
- **高可用**，服务提供方故障不会影响总线自身能力；
- 目前信用等级子系统已割接到总线，至今服务稳定。

数据中心

- 覆盖**43**个集群，**114**个项目，**1200+**台服务器；
- 共收录**1471**种日志，其中运营类日志**133**种；
- 每天执行任务约**4174**次，收集**4677**个日志文件，总大小约为**500G**；
- 上线后至今服务稳定，共收集约**30T**日志（压缩后）。



监控集成中心

- 收集共**8**个子监控系统的数据；
- （应用）覆盖**42**个集群，**114**个项目，**1200+**台服务器；
- （数据库）覆盖**25**个项目，**157**台主机，**187**个实例；
- 收集当前平台中间件**56**种实时信息。

大数据分析中心

- 支撑**153**个模块的监控和客服、数据库的需求；
- 系统采用高性能分布式架构；
- 可支持每日**TB**级别的日志分析，每秒钟可处理**10万**条日志。

精细化运营

- 开发延伸至生产 开发加入反馈 开发嵌入运维 运维也要开发
- 运维关注运营
- 开发关注运营
- DevOps

建立良好的分享机制，分享是更好的总结，避免重蹈覆辙、浪费精力

客户端为NAT环境，在大量并发情况下，如果开启快速回收，新连接的时间戳小于peer机器上次TCP到来时的时间戳，且差值大于重放窗口戳。丢包概率放大65535倍，服务基本不可用。

TIME_WAIT快速回收在Linux上通过net.ipv4.tcp_tw_recycle启用，由于其根据时间戳来判定，所以必须开启TCP时间戳才有效。

建议：如果前端部署了三/四层NAT设备，尽量关闭快速回收，以免发生NAT背后真实机器由于时间戳混乱导致的SYN拒绝问题。

运维体会

- 不吝啬技术分享，在分享的过程中，理解的更深刻，避免重复跳坑
- 为变化而设计、使用新的软件新的架构
- 先规范后实施
- 多一些横向扩展，少一些纵向扩展
- 时刻关注冗余、备份
- 监控正确的东西
- 有关数据图形化，关注历史数据
- 日记是非常重要的
- 优化是长期的工作
- 安全不松懈、巡查有保障

谢 谢！

qiugx@corp.21cn.com