私有PaaS在乐视的实践

陈轶飞 2016/05/14

关键词及适合听众

- 私有：
  - 公有技术更复杂；要考虑问题太多
  - 私有的更关注核心问题
- 实践：
  - 以务实为指导思想； 紧密结合业务特点和需求；
  - 不求高大上，但求实用
  - 只解决每个阶段的核心问题

- **适合听众**
  - **小型互联网创业公司**
  - **传统企业互联网化**

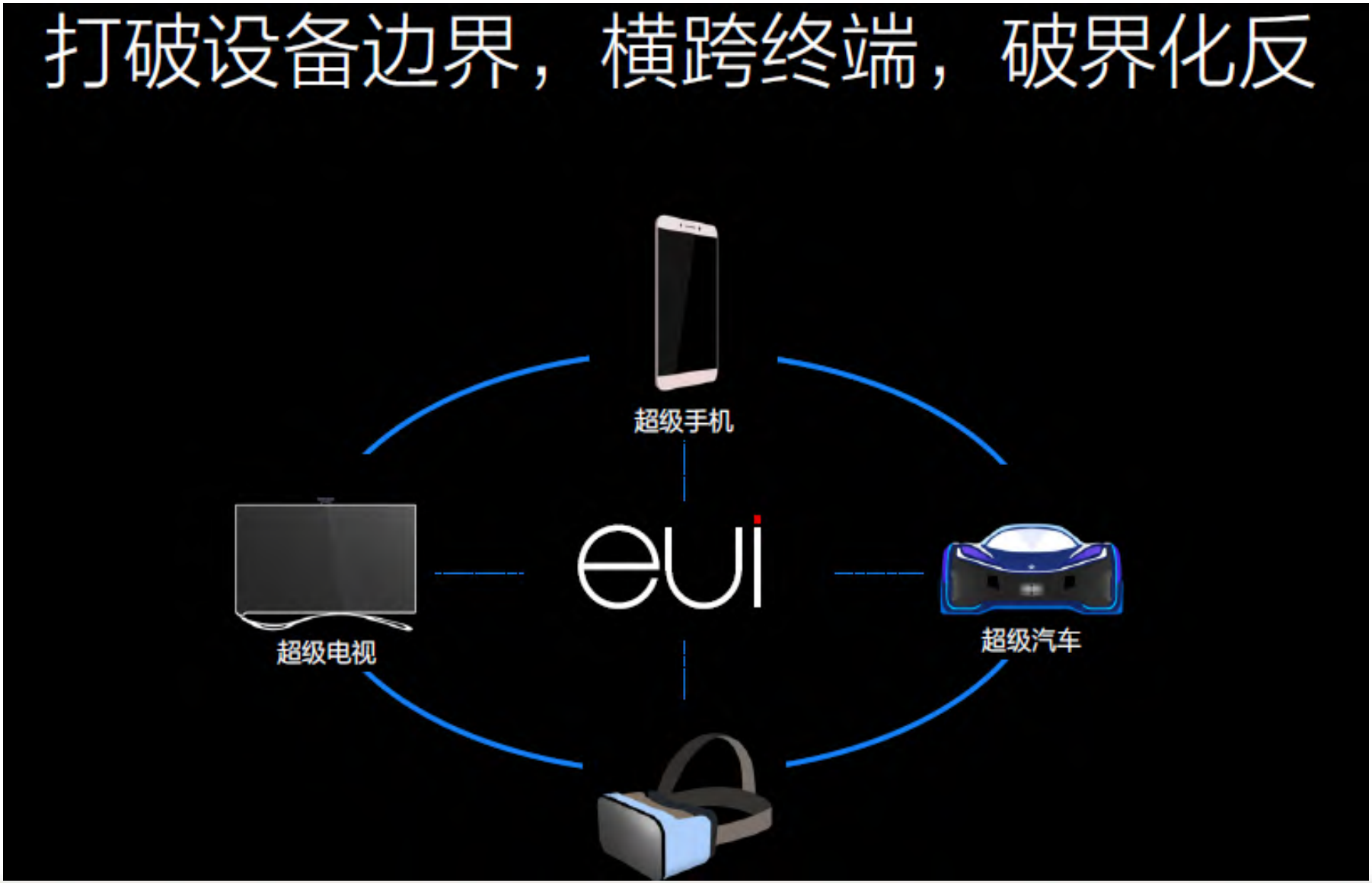
大纲

01 背景及发展历程

02 技术实现

03 微服务探索

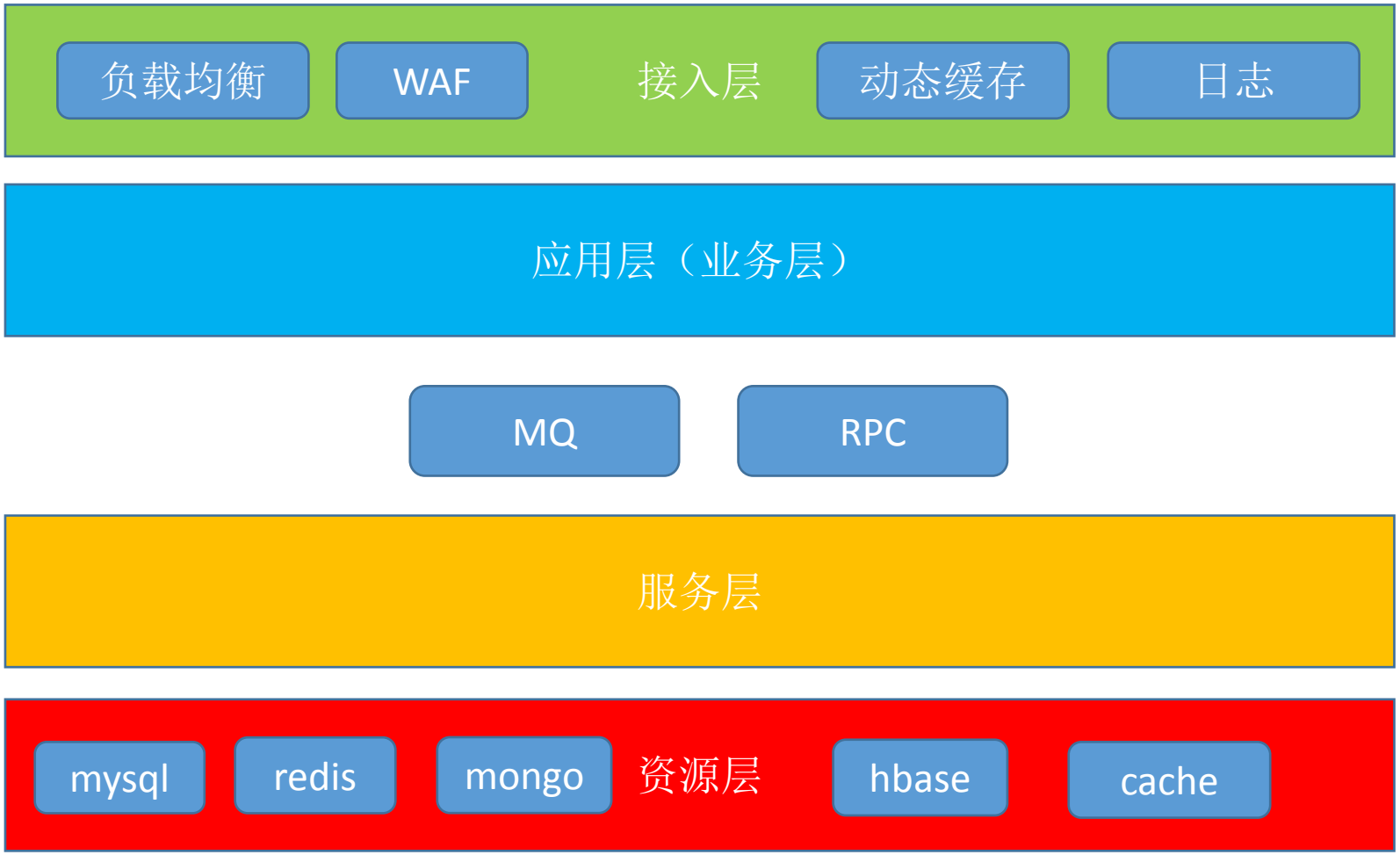
背景及发展历程


打破设备边界，横跨终端，破界化反
超级手机
eui
超级电视
超级汽车

智能设备云平台

推送	同步	OTA	设备管理	远程控制
视频通话	多屏互动	网络短信	群聊	边看边买
天气	黄页	壁纸	日历	闹钟
浏览器	免流量	个人云	桌面插件	.....

云平台技术架构


负载均衡
WAF
接入层
动态缓存
日志
应用层（业务层）
MQ
RPC
服务层
mysql
redis
mongo
资源层
hbase
cache

PaaS在云平台中的作用

- 打通接入层、应用层、服务层
- 承载了云平台95%以上的业务

PaaS发展历程：阶段一

- 问题：
  - 团队刚起步，基础设施一穷二白
  - 典型的一体式应用，所有的逻辑放在一个大的JAR 包里
  - 上线频繁，仅有的一个运维被折磨
- 实现方案：
  - 中控脚本
  - docker

PaaS发展历程：阶段二

- 问题：
  - 机器挂了要人工重新部署
  - 容器挂了要重新部署
  - 资源无法合理分配
  - 业务之间互相影响
- 实现方案：
  - Mesos + marathon + docker

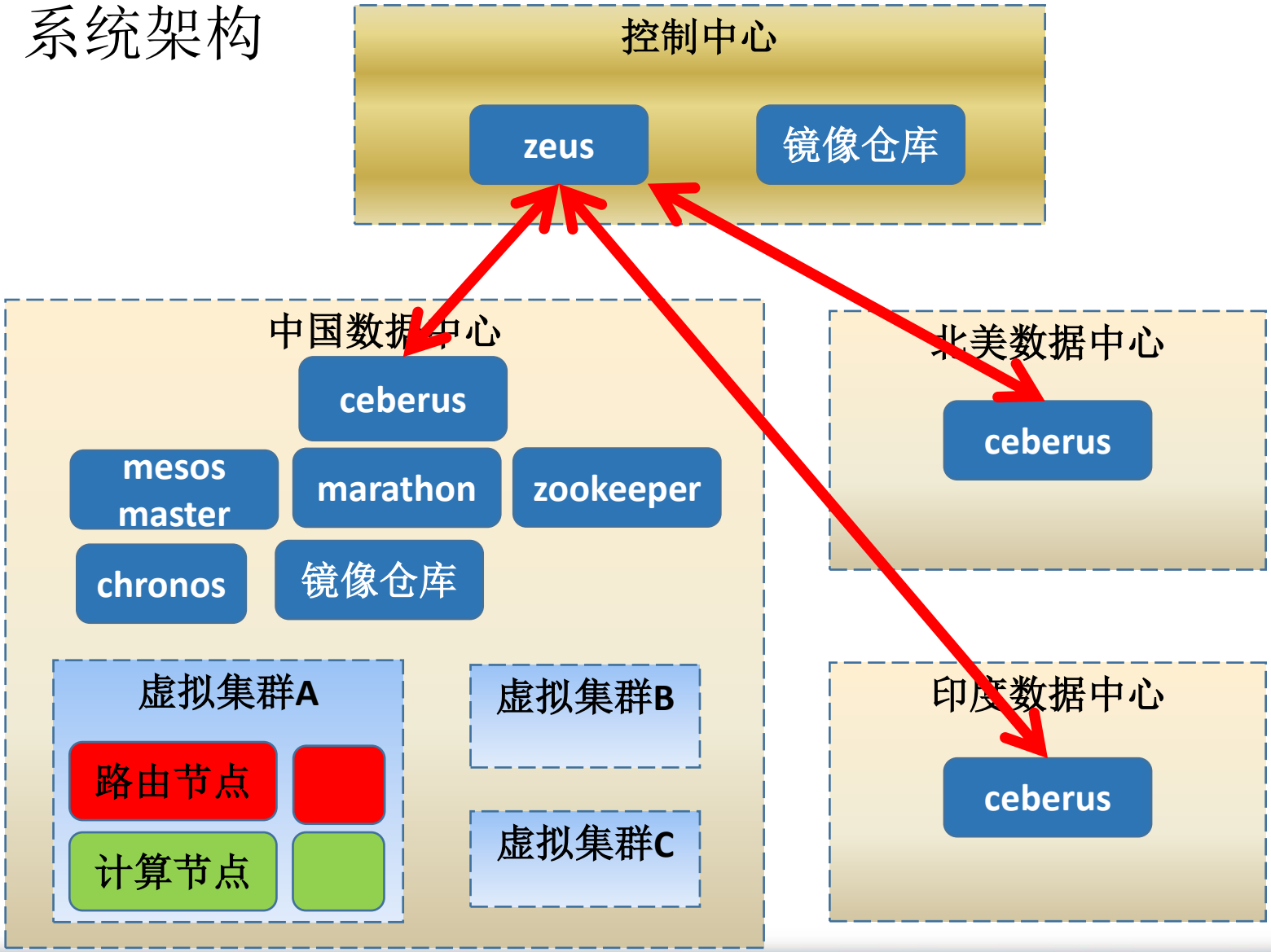
PaaS发展历程：阶段三

- 问题：
  - 业务逐步复杂化； 一个业务由多个服务组成
  - 多个业务实现了重复的功能，需要抽出来成为独立服务
  - 灰度发布、集中配置等新需求
  - 业务人员不知道需要多少资源； 容器评估： 健康、 忙、闲？
- 方案：
  - 支持灰度发布、配置集中化、微服务

PaaS平台现状

技术实现

系统架构

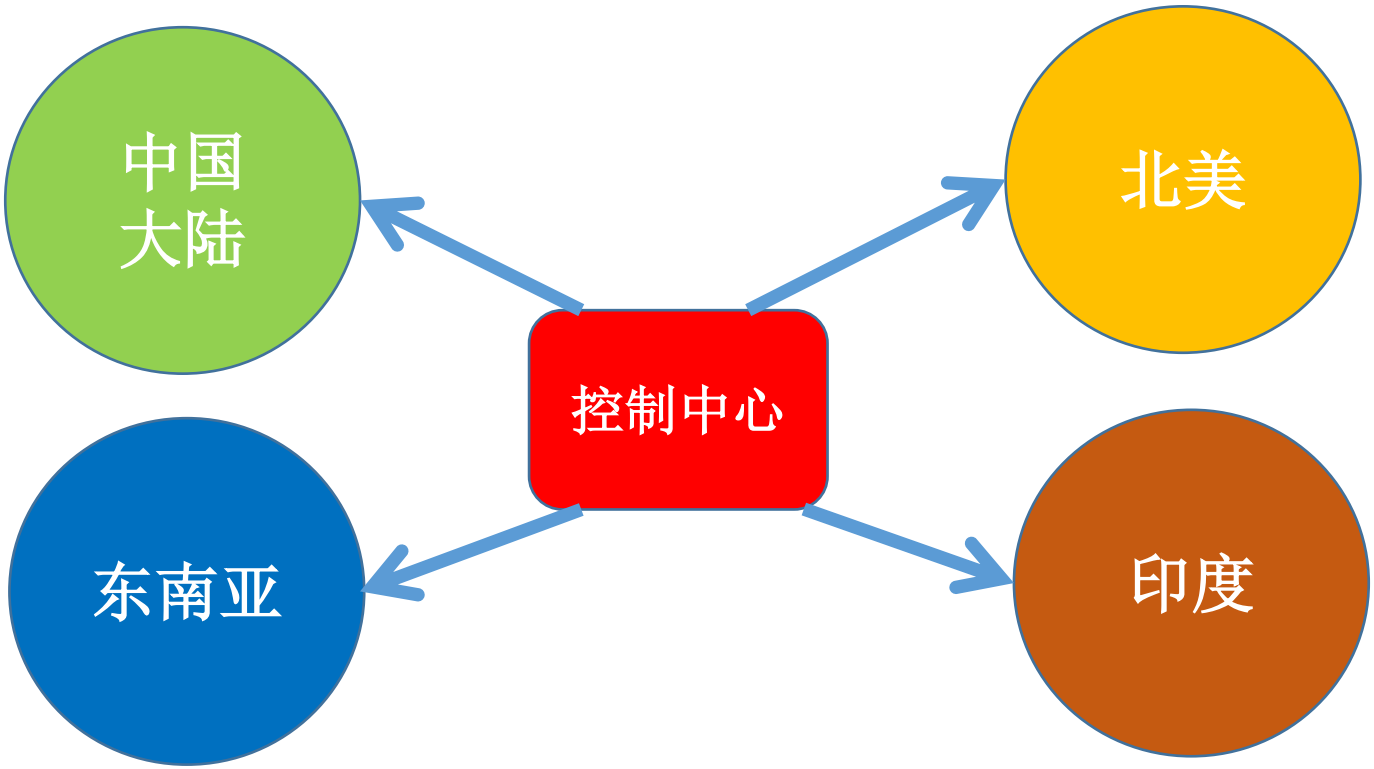

控制中心
zeus
镜像仓库
中国数据中心
ceberus
mesos master
marathon
zookeeper
chronos
镜像仓库
虚拟集群A
路由节点
计算节点
虚拟集群B
虚拟集群C
北美数据中心
ceberus
印度数据中心
ceberus

核心组件

- zeus + ceberus： golang自主开发
- mesos： >= 0.23
  - 尝试自己做 containerizer和executor
  - 目前选择的是原生的docker-containerizer
- marathon： long-run 应用
- chronos： cron 应用
- docker： >= 1.0
- zookeeper
- nginx：七层负载均衡
- haproxy： 四层负载均衡

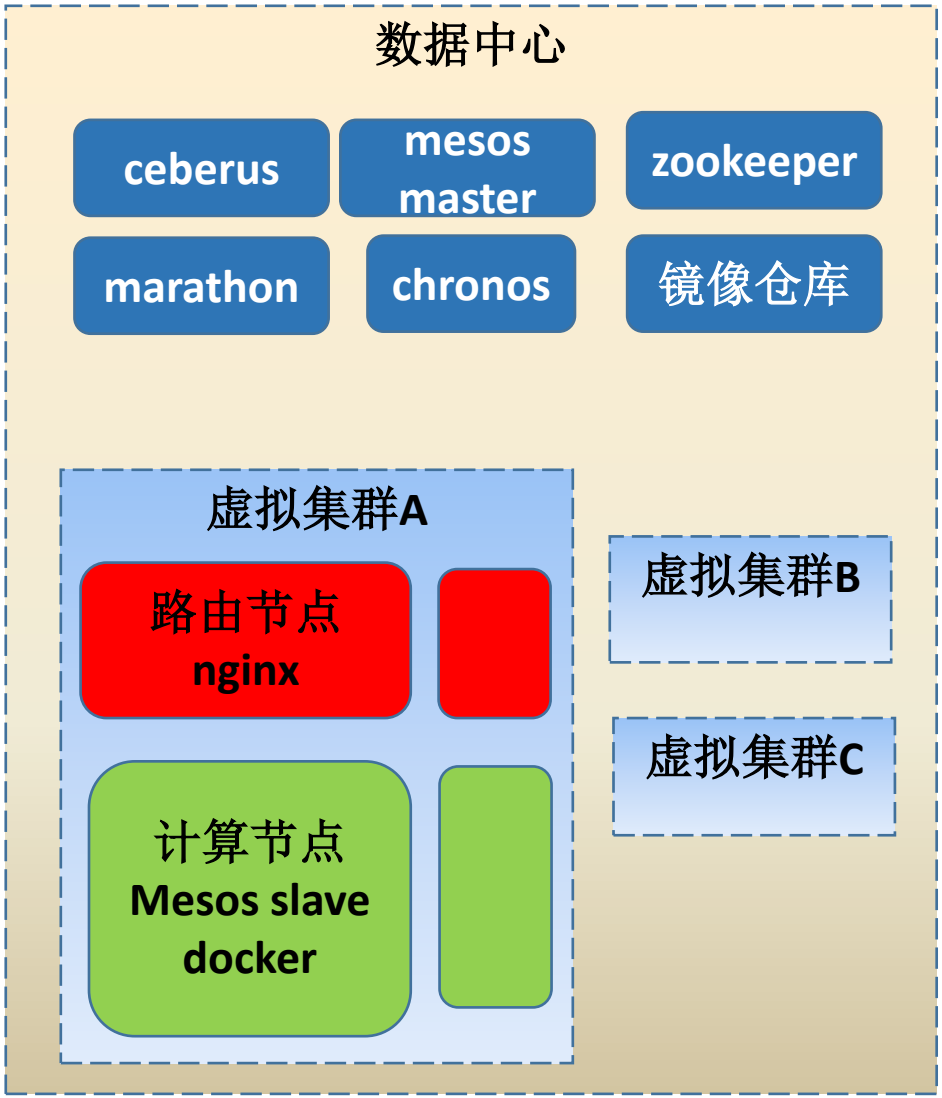
多数据中心

- 全球化发展战略要求应用全球部署
- 数据中心到控制中心之间的网络状况不稳定



多虚拟集群

- 每个数据中心使用一套基础组件管理多个虚拟集群
- 集群与集群之间使用mesos attribute进行逻辑隔离
- 每个集群包含一组计算节点和路由节点
- 计算节点：
  - mesos slave
  - docker
- 路由节点：
  - nginx 或 haproxy



两种发布方式

- 基础镜像 + 应用代码
  - 基础镜像包含支持业务代码运行的基础组件
  - 应用代码由zeus进行打包并分发到不同的数据中心
  - 不同业务可以共享相同的基础镜像
  - 接近于业务开发的习惯，学习成本低，易于接受
  - 代码源支持svn，git以及压缩包
- 纯镜像方式
  - 业界常用的容器云部署方式
  - 支持手动/自动触发编译镜像
  - 通过全球镜像架构保证各个数据中心镜像一致

Project + App的组织方式

- App是部署的最小单位
- 用Project来聚合一组逻辑上相关的App

多种App类型

- WEB： 暴露HTTP端口
- Cron： 周期性任务
- Task： 一次性任务
- Worker： 通常不对外暴露端口
- Service： 微服务； 暴露TCP端口

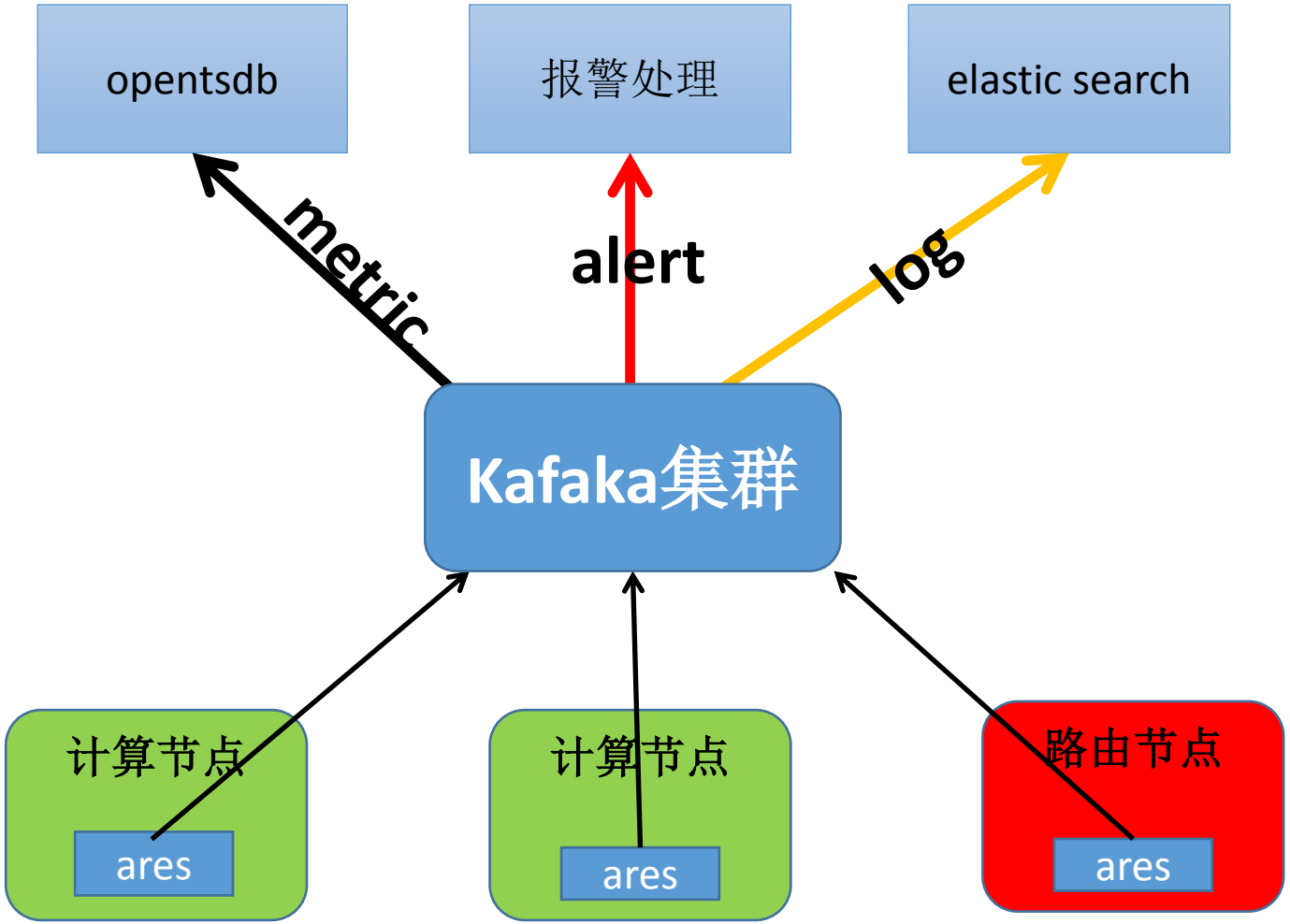
接入层负载均衡

- 七层负载均衡： nginx
  - Nginx上部署agent，接受规则变更通知
  - 规则加载：
    - 早期：Nginx reload 实现规则加载
    - 现在： 自定义 lua 模块，实现规则平滑加载
  - 支持nginx自定义配置

Metrics collect

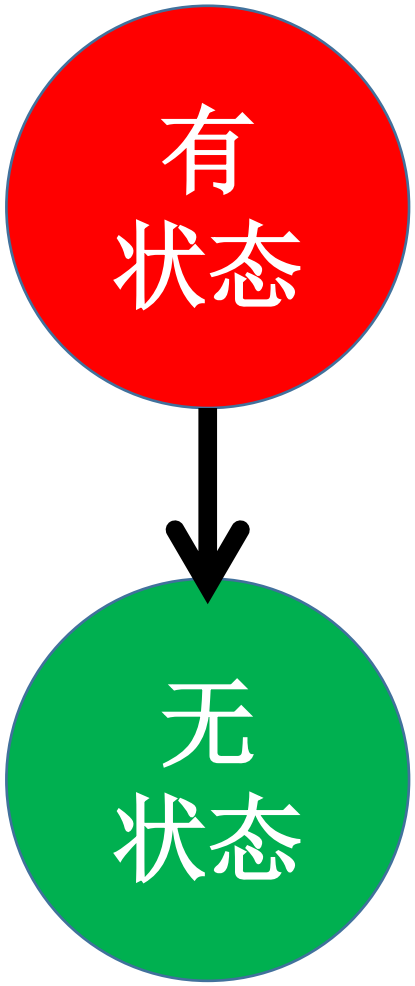
- 目的：
  - 及时报警
  - 容器资源使用图表
  - 为容器评估及扩容/缩容提供依据
- metrics
  - 容器：
    - 内存、CPU、网络I/O
    - 在容器外采集，更优雅
  - 前端nginx：
    - 请求数
    - 请求处理时间
    - 5xx个数

Metrics collect 实现

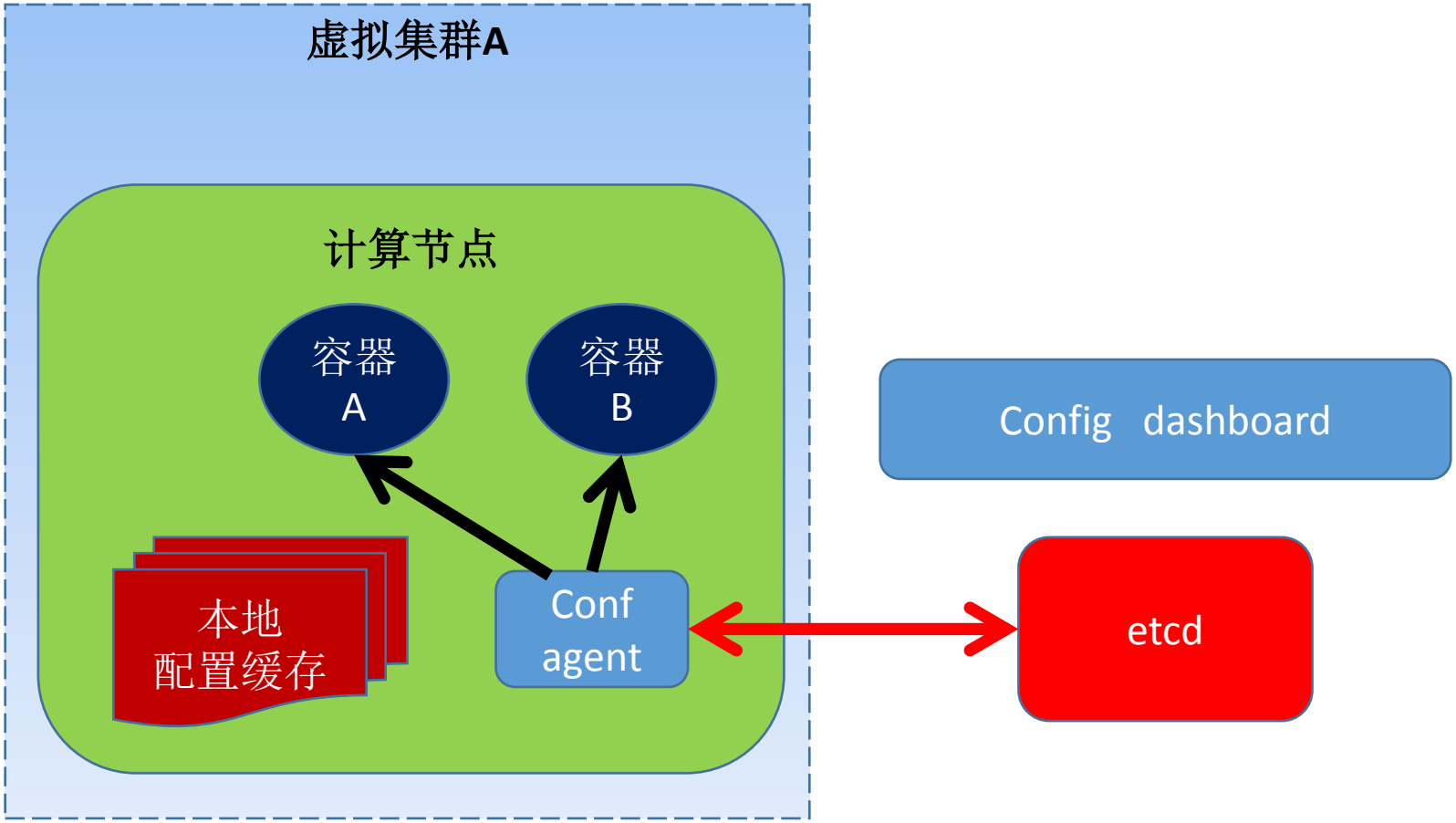


配置集中化及变更通知

- 传统配置管理方式的缺点
  - 将配置文件和代码一起维护和部署。
  - 当运行环境存在多样性，配置与运行环境适配就存在很大的难度。
- 采用配置文件集中管理的好处：
  - 方便维护：不同应用不同环境的配置文件的统一管理保存，并提供可视化界面进行文件内容的增删改查。
  - 方便部署：一次修改多个运行环境多个实例及时生效。



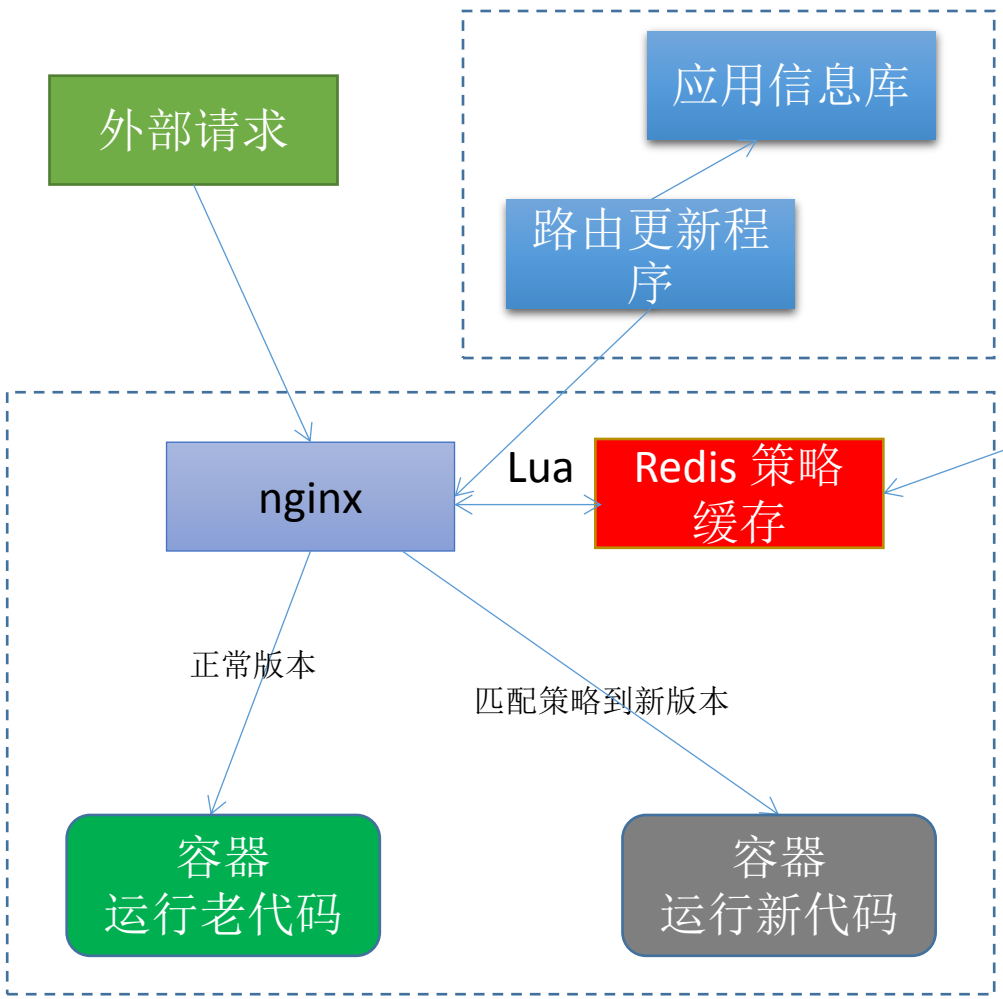
配置集中化实现



灰度发布

- 分流点
  - 接入层
- 分流方案
  - Nginx + LUA模块 + redis策略库
- 分流依据
  - 设备型号
  - IP地址
  - 区域
  - 流量百分比
  - Cookie

灰度发布架构



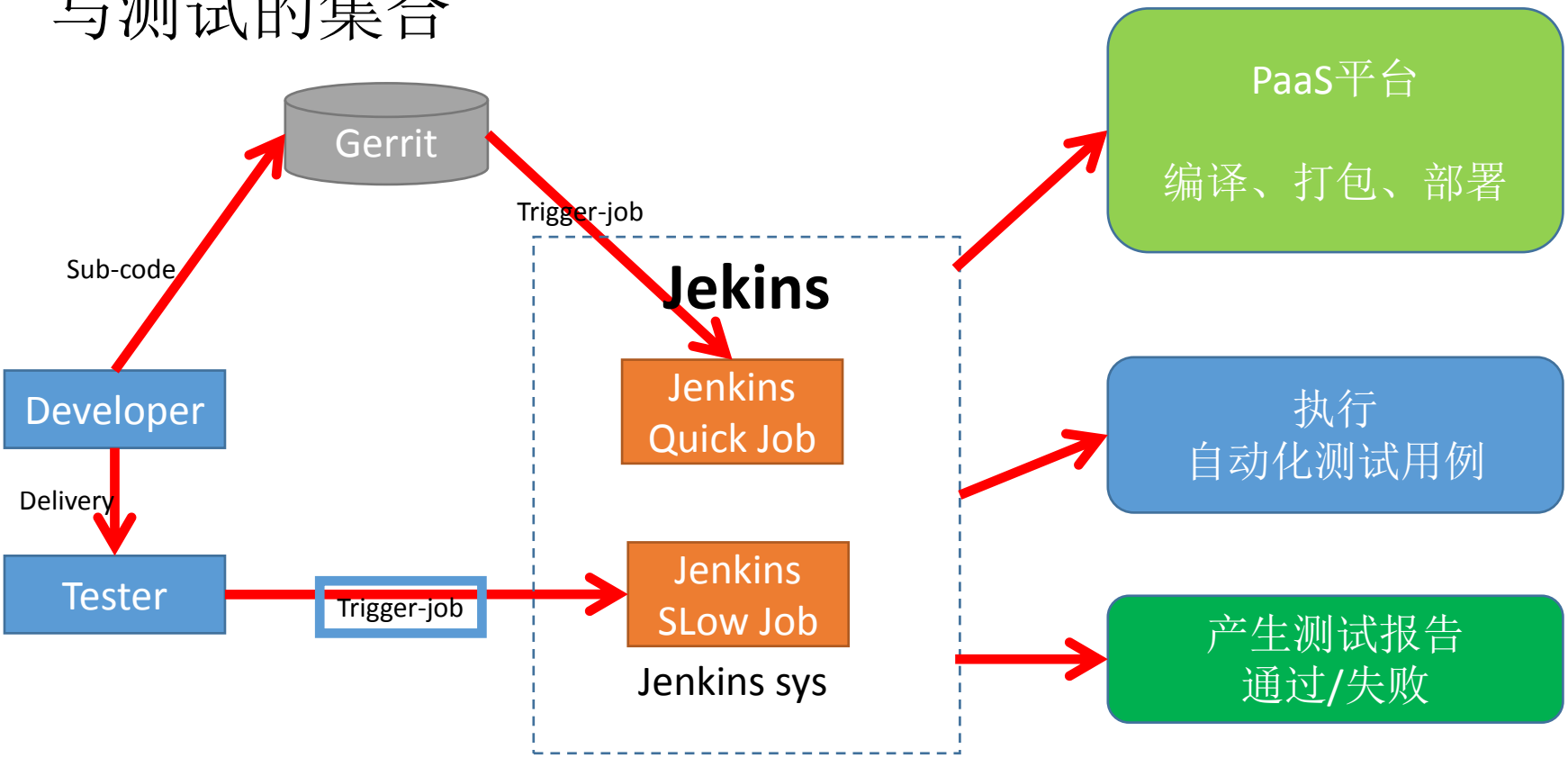
分流特点

1.没有灰度版本的应用不会走灰度分流环节
2.简单策略直接在nginx配置
3.复杂策略调用lua代码分流

分流流程

1.部署灰度应用
2.创建灰度策略
3.关联灰度版本及分流策略
4.应用策略生效

与测试的集合



Quick Job流程:

1. 开发工程师提交源码到Gerrit，自动触发job；
2. Job 调用Pass rest-api，自动构建并部署测试环境；
3. Job 触发冒烟测试用例，并生成“测试报表”；
4. Job 执行PASS，“合入主干”。否，CI失败。

Slow Job流程:

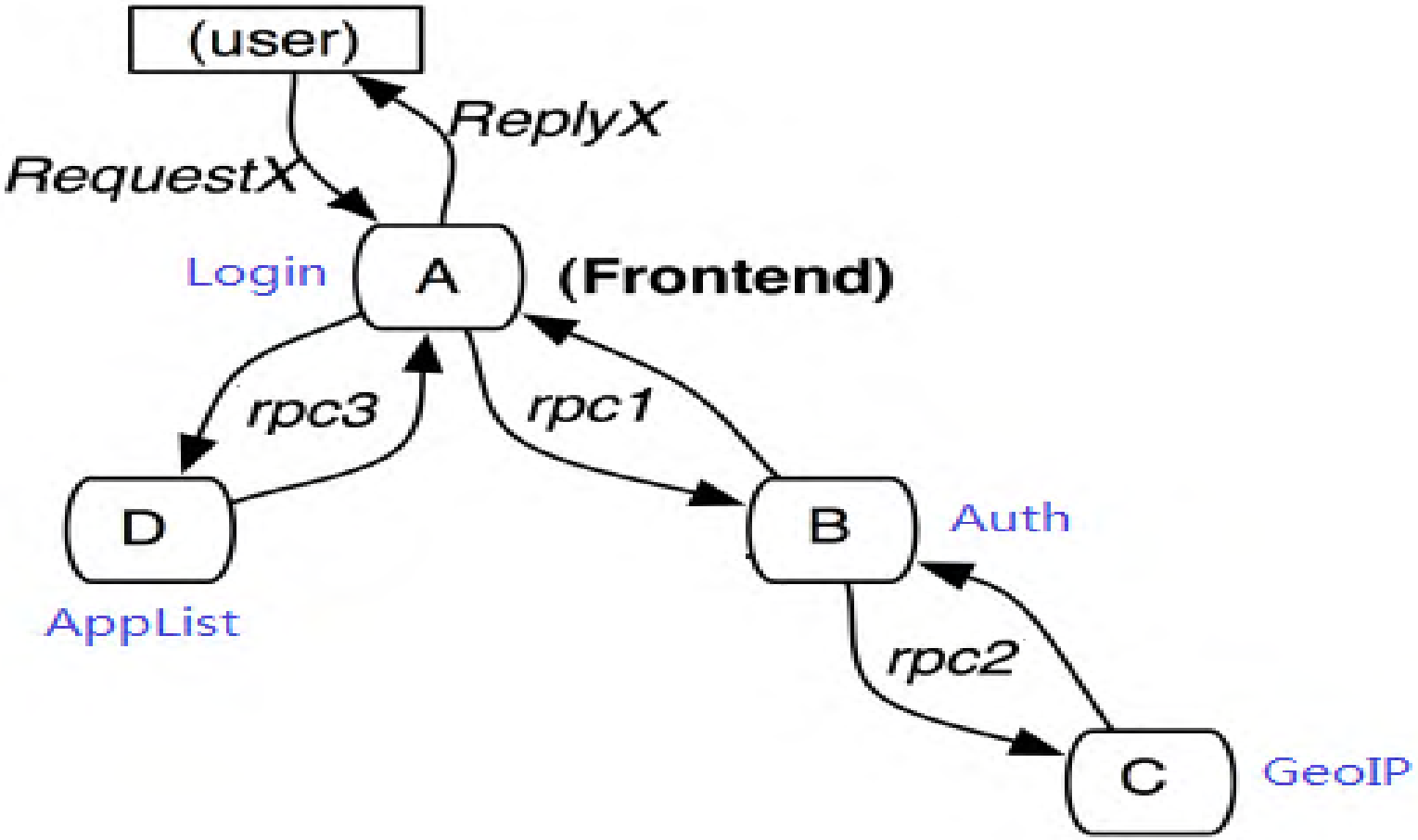
1. 开发工程师交付项目于测试工程师，自动触发job；
2. Job 调用Pass rest-api，自动构建并部署测试环境；
3. Job 触发全量测试用例，并生成“测试报表”。
4. 如果job 执行PASS，“测试准入”。否，准入失败。

微服务探索

为什么需要微服务

- 业务逐步复杂化，从开发效率、易维护性、性能等角度考虑，需要将功能拆解，形成独立的服务
- 多个业务可能都实现了相同的功能，需要提取出来形成公共的服务

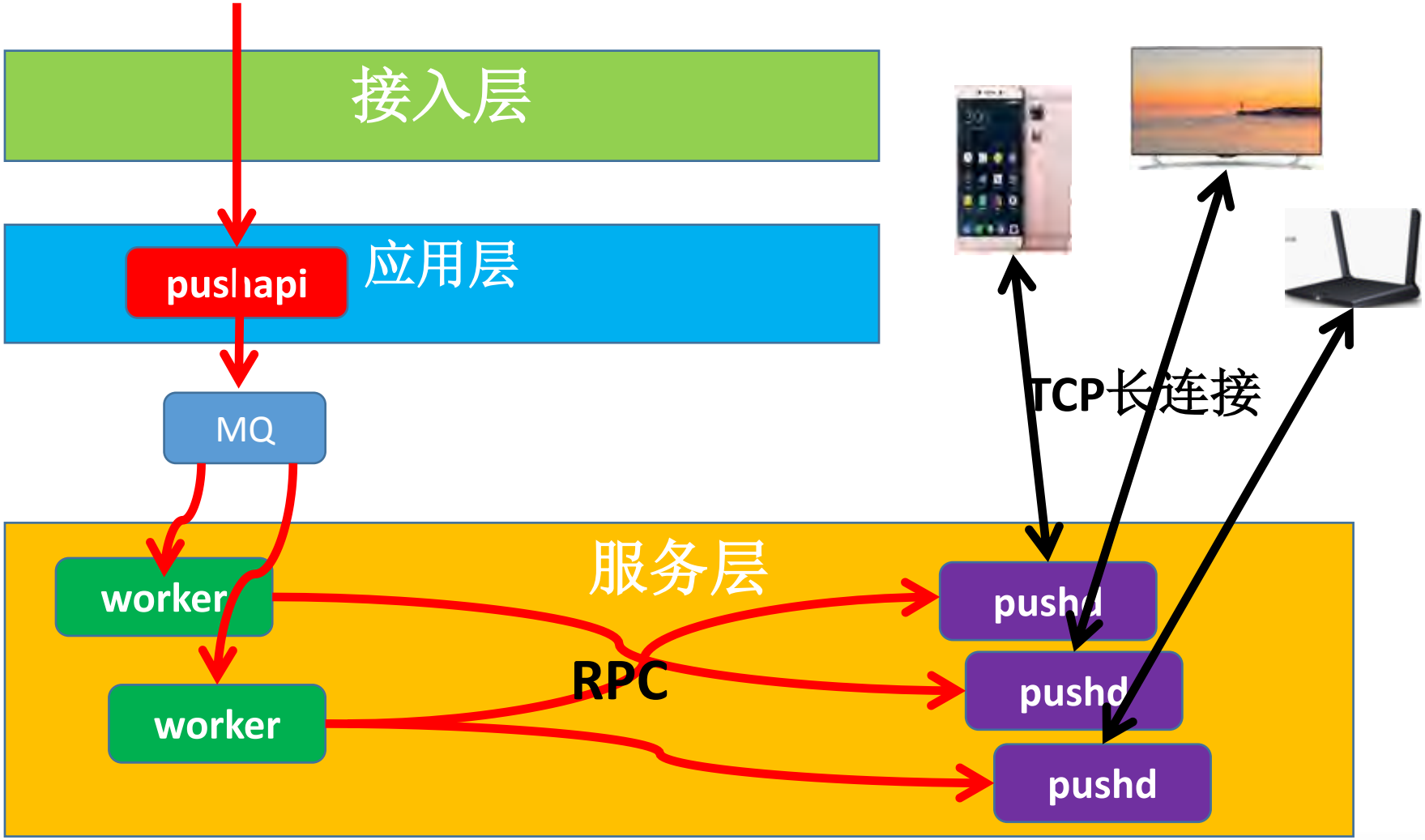
举个例子


(user)
ReplyX
RequestX
Login
A
(Frontend)
rpc3
rpc1
D
B
Auth
AppList
rpc2
C
GeoIP

微服务架构实现

- 两种微服务形式
  - 同步类微服务： 基于RPC直接通信
  - 异步类微服务： 基于MQ
- 轻量级微服务框架
  - 基于thrift RPC框架改造，支持多语言
  - 修改thrift，支持调用链trace
- 服务注册中心
  - 目前支持 etcd
  - 下一步支持 consul
- 服务发现与负载均衡
  - 目前支持Smart client（性能更好）
  - 下一步支持Load balancer

示例： PaaS平台对微服务的支持



服务调用链trace

- 实现机制：埋点 → 日志输出 → 收集和存储日志 → 分析调用链 → 展示
  - Trace ID：关联一次请求相关的日志，全局唯一，在系统间传递
  - RPC Span ID：识别日志埋点顺序和嵌套关系，在系统间传递
  - 开始时间
  - 调用类型
  - 对端IP

- 参考Google Dapper论文

“Dapper, a Large-Scale Distributed Systems Tracing Infrastructure”

http://research.google.com/pubs/pub36356.html

服务调用 trace 日志

[ipq] RPC cost: 271218ns trace id(956-937-823) {"result":0,"data":{"country":"中国","province":"北京市","city":"朝阳区","isp":"中国联通"}}

[auth] RPC cost: 5091080ns trace id(956-925-937) auth success

[applist] RPC cost: 3989ns trace id(956-925-331)

1) "com.letv.LeRouter.iphone.client"
2) "com.letv.remoteControl.iphone.client"
3) "com.letv.LeHomeTime.iphone.client"
4) "com.kuyun.common.androidtv"
5) "com.stv.message"
6) "com.android.calendar"
7) "com.letv.android.personalized"

[mydashboard] HTTP cost: 12818469ns trace id(956-956-925) login success

遇到的问题及解决办法

- 故障诊断： 由研发人员登录机器， docker enter 进入
- 迁移后的日志保留
- 日常开发： 开发人员按自己习惯在本地开发
- 镜像制作：

待完善功能

- 镜像仓库&镜像管理
- WEB控制台
- 集群自动化管理
- 过载保护
- 自动伸缩
- 服务降级

谢谢！