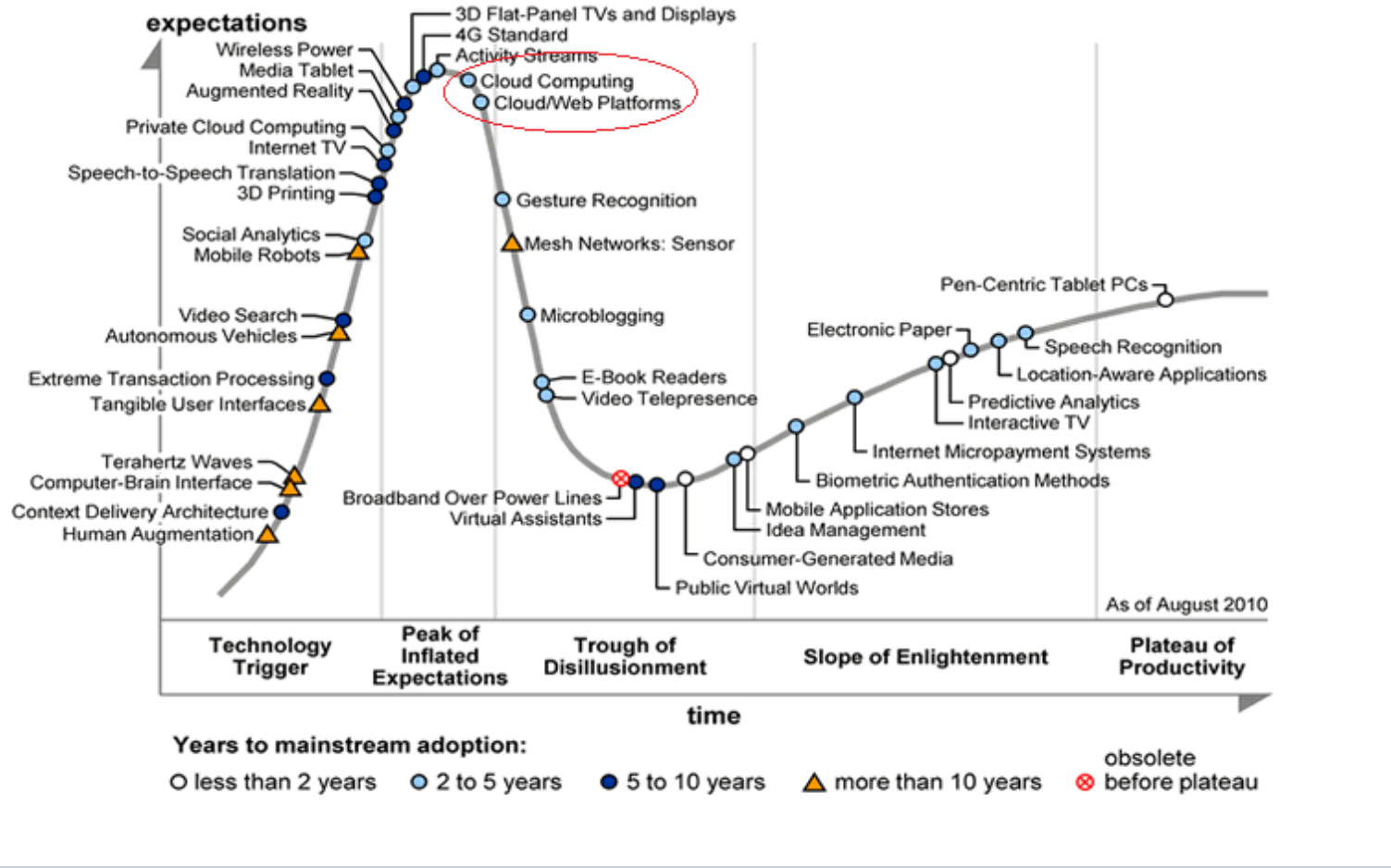
企业私有云应用交付平台

Ronnie.shi79@gmail.com

文档说明

- 背景
  - 云计算在经过一轮概念炒作后已进入发展期，今年后将达千亿规模
  - 项目建立在腾讯云计算和创业型公司的云计算实践基础之上，经过半年的市场调查和竞争分析后确定
- 讨论范围
  - 文档主要从技术和市场的角度讨论产品的商业场景，市场机会，产品特点以及它的技术架构和面临的问题等
  - 文档所讨论的只是问题的基本框架，用以作为后续决策与开发的指导，并不代表产品的最终形式
- 潜在读者
  - 面向市场人员与开发人员
  - 部分可以作为客户，投资人沟通的材料

Gartner 2010 Hype Cycle



目标说明

- 两个维度的考虑
  - 按部署模式分为公有云，私有云，混合云等
    - 公有云：国内信用和契约精神缺乏坚实的社会基础，数据安全和可靠性成为公有云开展的拦路虎；国内公有云缺乏应用热点和旗舰型企业
    - 私有云：带来了云的便利的同时规避了企业云应用的安全性问题，目前尤其是大中型企业云部署的首选
  - 按服务模式分为IAAS，PAAS和SAAS
    - SAAS：一般部署在共有云上，企业有数据安全的担忧
    - PAAS：引入了新的编程方式， 对国内大部分IT维护人员，学习成本高昂；稳定性，可靠性让它成为巨头们的舞台；目前只适合新应有的部署
    - IAAS：IT设施整合的基础模式；最好的兼顾已有应用；迁移成本最低的云服务模式
- 我们的选择：基于IAAS的私有云应用交付平台
  - 基于市场调查和以上技术的分析
  - 目的是在为企业搭建一个私有云平台的基础之上，构筑企业应用的管理和交付平面，方便企业应用的部署和资源的合理化利用

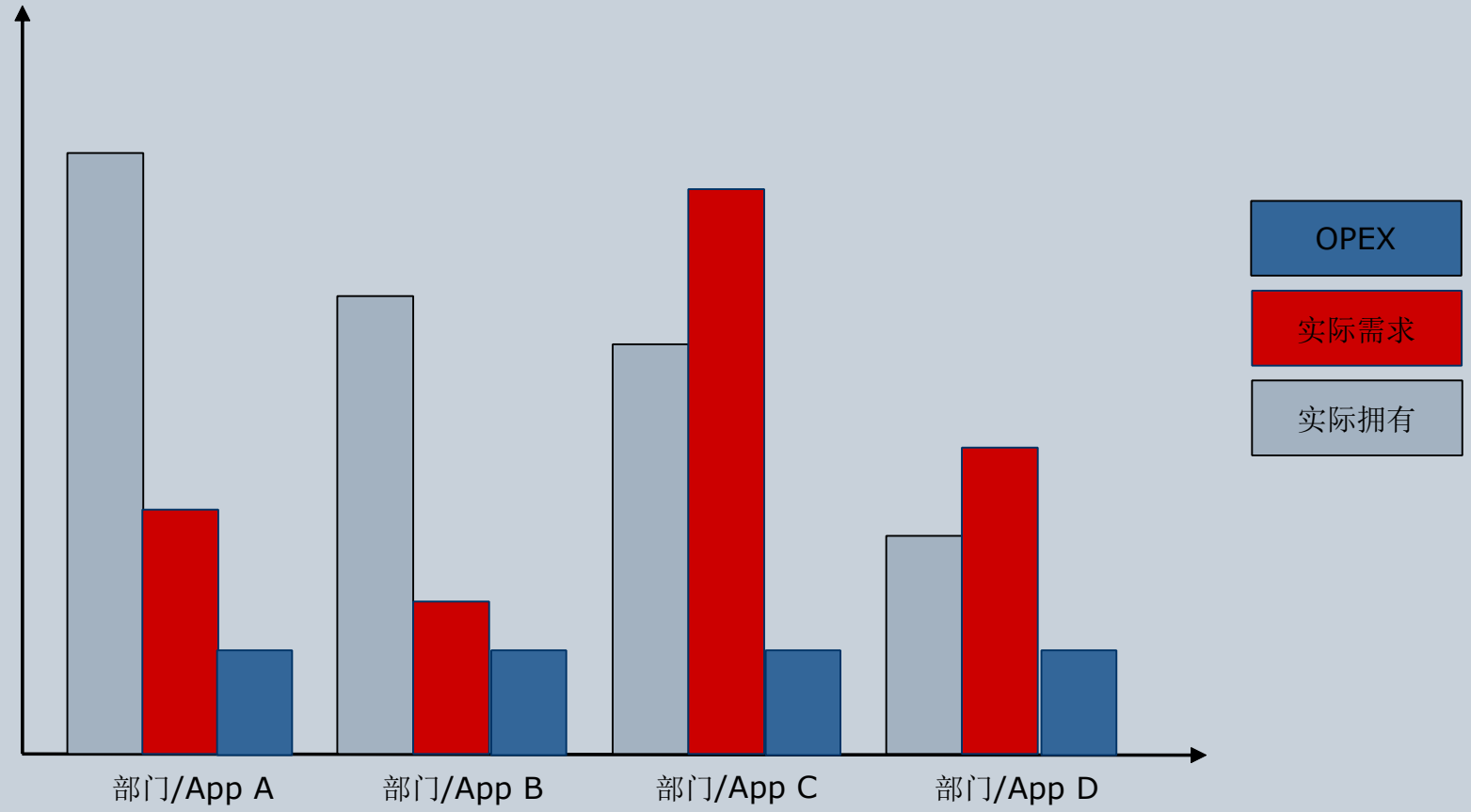
Business Use Case

Ronnie.shi79@gmail.com

企业IT的困扰

- 烟窗式企业计算环境，资源利用率低
  - 在资源有限情况下，无法应对峰值流量；而为峰值运算部署大量资源，导致资源利用率低，一般在20%
  - 资源回收，跨应用协调困难
- 管理复杂，成本高
  - 存在部门墙，不同部门使用相互隔离的IT资源；同部门资源竞争无序
  - 存在信息孤岛现象，IT设施技术管理与维护复杂，人员成本高
  - 资源缺乏统筹管理与调度；不透明，缺乏IT决策基础
- 业务部署慢，降低企业应变能力
  - 新的业务需要新的硬件，包括规划，采购，配置，部署，周期长
  - 不能适应快速多变的应用需求，降低公司竞争力

企业IT的困扰（续）

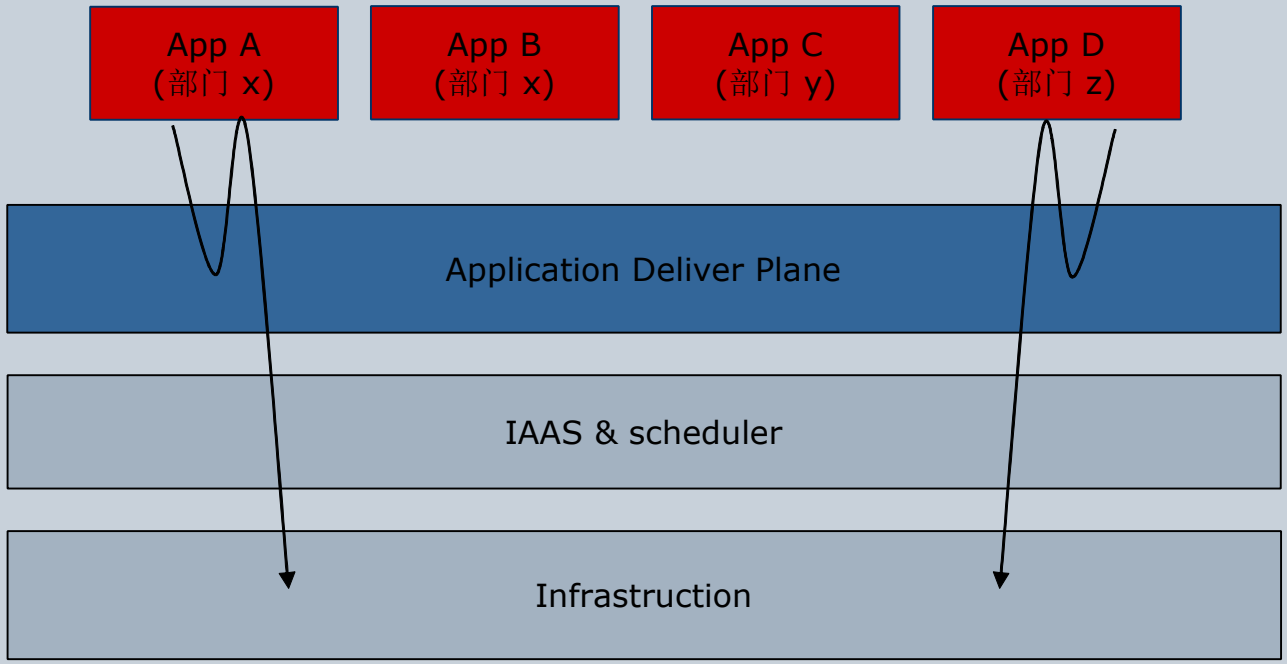


解决之道

- 引入统一管理的，弹性的IAAS平台
  - 统一管理IT资源，避免资源的部门墙
  - 弹性复用的IAAS平台提高资源的利用率
  - 有效的资源监控和回收手段降低资源的使用冲突和浪费

- 引入面向应用的自动交付平台
  - 提供自动化的应用部署平台，降低IT管理的复杂度和使用成本；减少IT运维人员的需求
  - 加速业务的部署，提升企业竞争力
  - 按需使用，智能调度手段保证高优先级业务的可用性

解决之道(续)

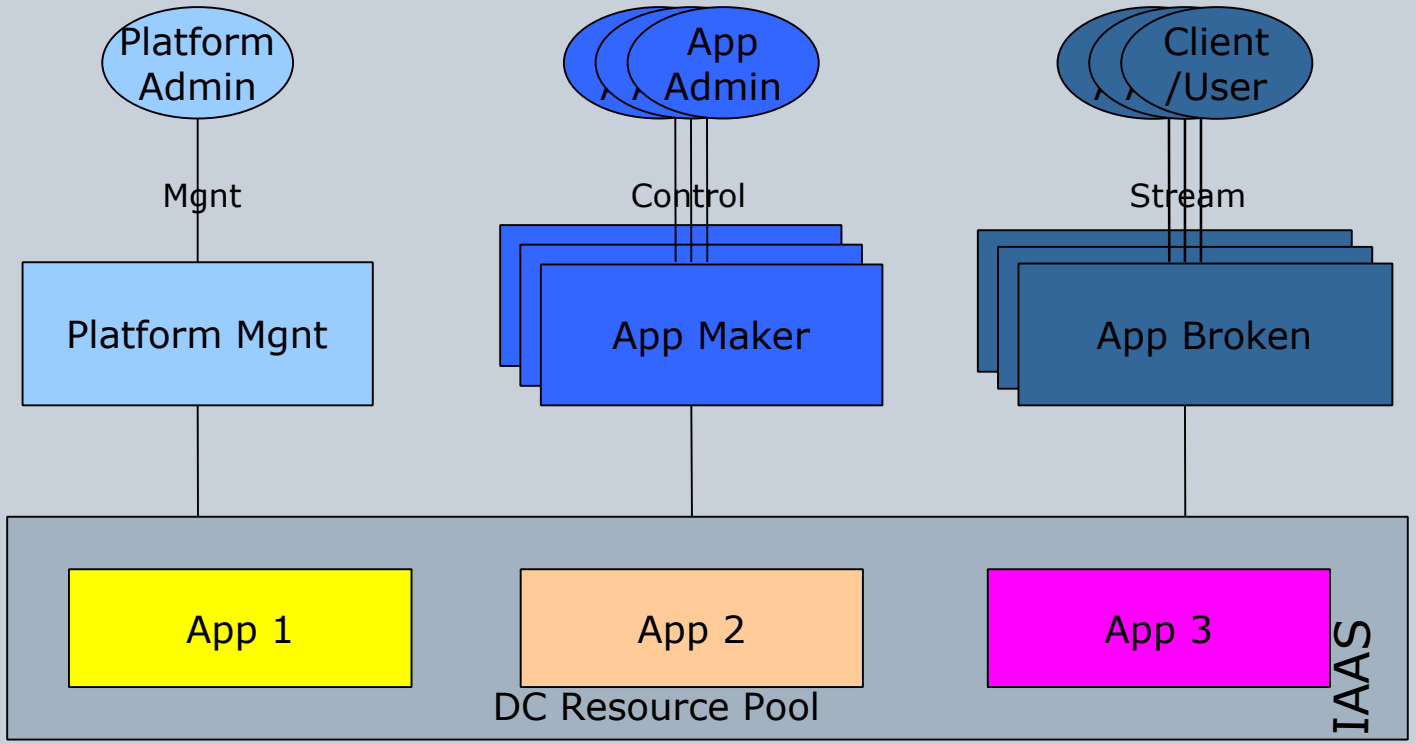


使数据中心及其应用的管理/监控/使用像Windows一样简单高效

产品理念

- 以企业私有IT应用为中心
  - 提供灵活，弹性的私有IT平台，满足各种应用需求
  - 为各种应用提供快速部署平台
  - 最大限度利用已有资源，降低IT成本
- 创新，差异化；平滑，安全迁移企业应用
  - 基于IAAS的应用交付平面，与IAAS差异化；与主流云IT供应商差异化，如HP， IBM，Cisco，HW等
  - 立足云计算技术，应用逐步迁移，渐进演进
- 以最终使用者为出发点，简单易用
  - 简化IT管理，减轻IT使用复杂度
  - 灵活满足使用者需求；提升IT使用体验

Use Scenarios



Use Scenarios

- 在DC IAAS平台， 创建一个App
  - Platform Admin 在该平台上通过Mgnt平面预配置该应用，分配App资源，指定App的优先级， 控制权限等
  - App Admin 通过App Maker 完成App的访问配置，资源使用方式等，然后使能该应用
  - 最终，User/Clinet 可以通过App Broken 访问该应用
- 各App在IAAS平台复用各种硬件资源
  - 如存储，网络，计算资源等；支持智能调度，确保关键业务的可用性
- Platform支持并发操作
  - 多App 可以同时促发使能，App Maker支持并行处理
  - 多User可以同时访问各种App，Broken支持负载均衡和并行处理

平台特点(一)

- 自助式IT交付平台
  - 改变IT使用习惯，打破IT资源的管理方式
  - 减少IT的管理成本，提高IT资源的可达性和使用效率
- 面向应用，提供一键组合的应用交付方式
  - 基于IAAS的应用交付平面，与普通IAAS差异化
  - 提供应用模板，组合使用；简化IT管理
  - 实现应用的一次配置，多次使用
- 职责清晰
  - 清晰的四平面设计
  - 清晰的平台角色和职责定义
  - 个性化的用户界面展现

平台特点(二)

- 按需使用，自动快速部署
  - 统一的资源管理，打破IT使用隔阂，降低IT投入
  - 基于虚拟化的资源池管理，提供灵活，弹性的资源策略
  - 自动匹配业务需求，按需部署，自动回收资源
  - 基于优先级的业务策略保障关键业务
- 模块化，可配置的系统接口
  - 灵活适应各种IT网络
  - 适应企业现有IT架构，“即插即用”
- 强大的安全机制
  - 用户认证
  - 多平面网络隔离；用户平面与管理隔离

平台特点(三)

- 简单灵活
  - 提供最直观的管理界面
  - 面向系统接口，隐藏复杂技术，如VM
  - 适应不同环境需求
- 纯系统软件方案
  - 硬件独立，充分利用用户已有资源
  - 与HP，VCE，IBM，Oracle区别化
- 多平面并发处理
  - 业务流和管理流可同时并发
  - 高效处理平台

市场机会，现状与挑战

Ronnie.shi79@gmail.com

市场机会

- 云计算代表未来IT使用方式
  - 异构的IT基础架构有融合的趋势，IT巨头的加入加速了这个过程
  - 未来属于内容，管道上的内容百花齐放，需要的数据中心越来越多
  - 弹性复用，高度智能化的IT基础设施进一步降低了业务部署的门槛，Service开发可以将精力集中在业务层面
- 用户需求强烈
  - 相对共有云，市场对私有云的认同度在提高；私有云因其安全性会首先成为云计算的热点。IT调查咨询机构Info-Tech公布的报告显示，76%的IT决策者会首选或只选部署私有云
  - IT资源的整合管理和有效利用越来越被企业重视
- 与大公司差异化；竞争小
  - 技术门槛高，投入的小公司少
  - 大的公司如HP/CISCO/HW等，提供的是Total Solution，有硬件绑定

市场机会(续)

- 产品定位
  - 企业私有云改造方案，为企业提供循序渐进，自然的云演进方案
  - 减少用户的新增IT投入，降低使用成本
  - 面向应用，与普通IAAS差异化，降低IT管理和使用复杂度

- 目标客户
  - 首先面向有IT复杂环境，但技术能力薄弱的行业
    - 如教育，高校，医疗，政府等
  - 面向有复杂IT环境，需求多变的中大型企业和行业应用
    - 公共计算平台；异地分支复杂的企业
    - 面向业务复杂的互联网企业
  - Service数目在30-2000左右的数据中心
    - 过大的，复杂度高，目前系统难以满足；过小的，效果不明显

项目规模与现状

- 项目规模
  - 项目涉及到网络，虚拟化，文件系统，存储，应用系统，Web开发
  - 初始系统考虑不提供HA，跨数据中西支持等复杂功能；大概12万行代码左右
  - 充分利用开源软件
- 现状：产品原型开发与验证
  - 基本完成产品框架的定义
  - 完成产品概念，竞争分析，用户分析
  - 先期完成产品的原型，Demo开发

资源需求

- 资金需求
  - 办公设施，办公场所
  - 基础设施：服务器，网络设备等
  - 人工成本

- 人力需求
  - 强力的企业产品营销人员
  - 网络开发 2名
  - 存储开发 2名
  - 虚拟化开发 3名
  - Java J2EE开发 4名
  - Web开发 2名

困难与挑战

- 面向复杂的企业IT管理市场
  - 产品面向复杂IT的IT管理，需求多样且复杂， 需要强大的市场引导
  - 需要非常好的市场人员和策略
  - 营销成本高
- 云计算是个系统工程，需要系统思维的资深工程师
  - 涉及到数据中心各方面的技术，如虚拟化，存储，分布式计算，集群文件系统，网络，数据库等， 复杂度高
  - 工程师需要统筹各方面技术，系统思考；涉及到数据中心各方面的技术都为基础技术， 要求较高
  - 产品竞争力最终体现在细节，系统还有太多细节需要考虑
- 涉及面广，初次投入成本高
  - 硬件成本，人力成本，营销成本相对其他技术高

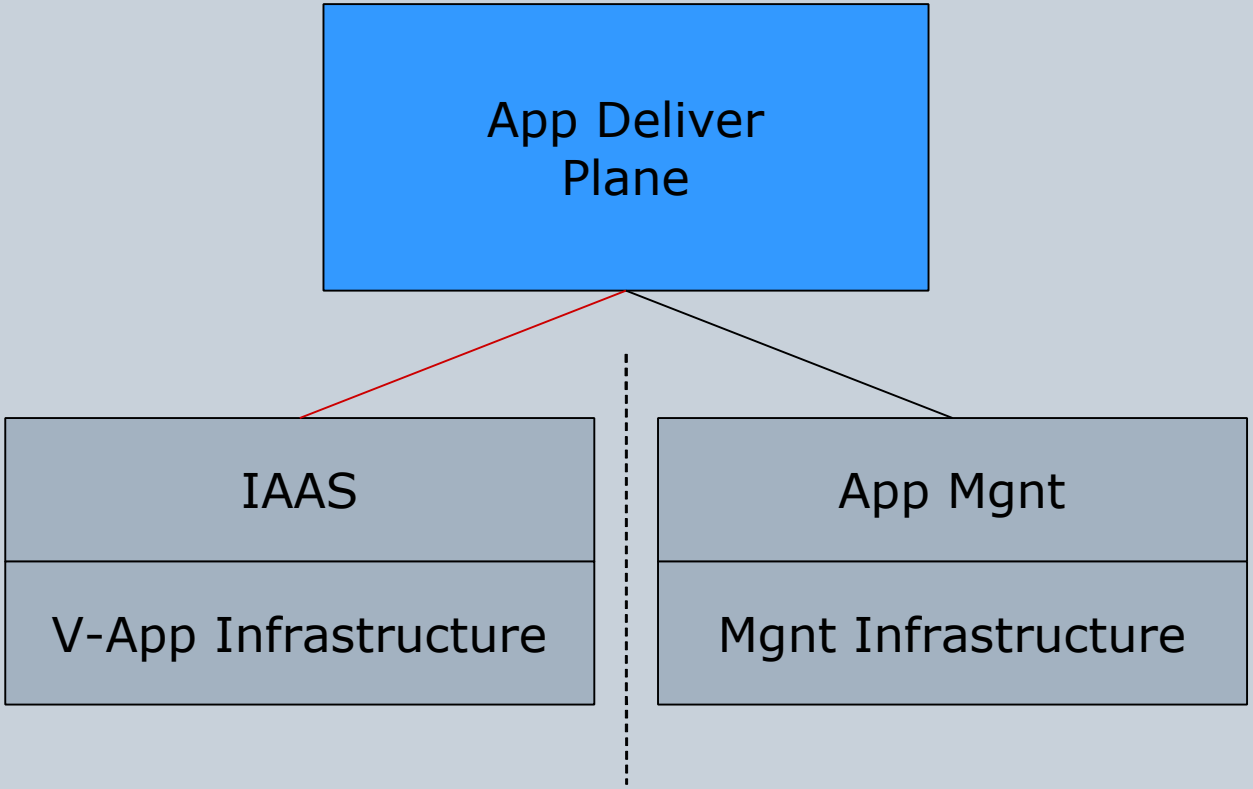
系统架构定义

Ronnie.shi79@gmail.com

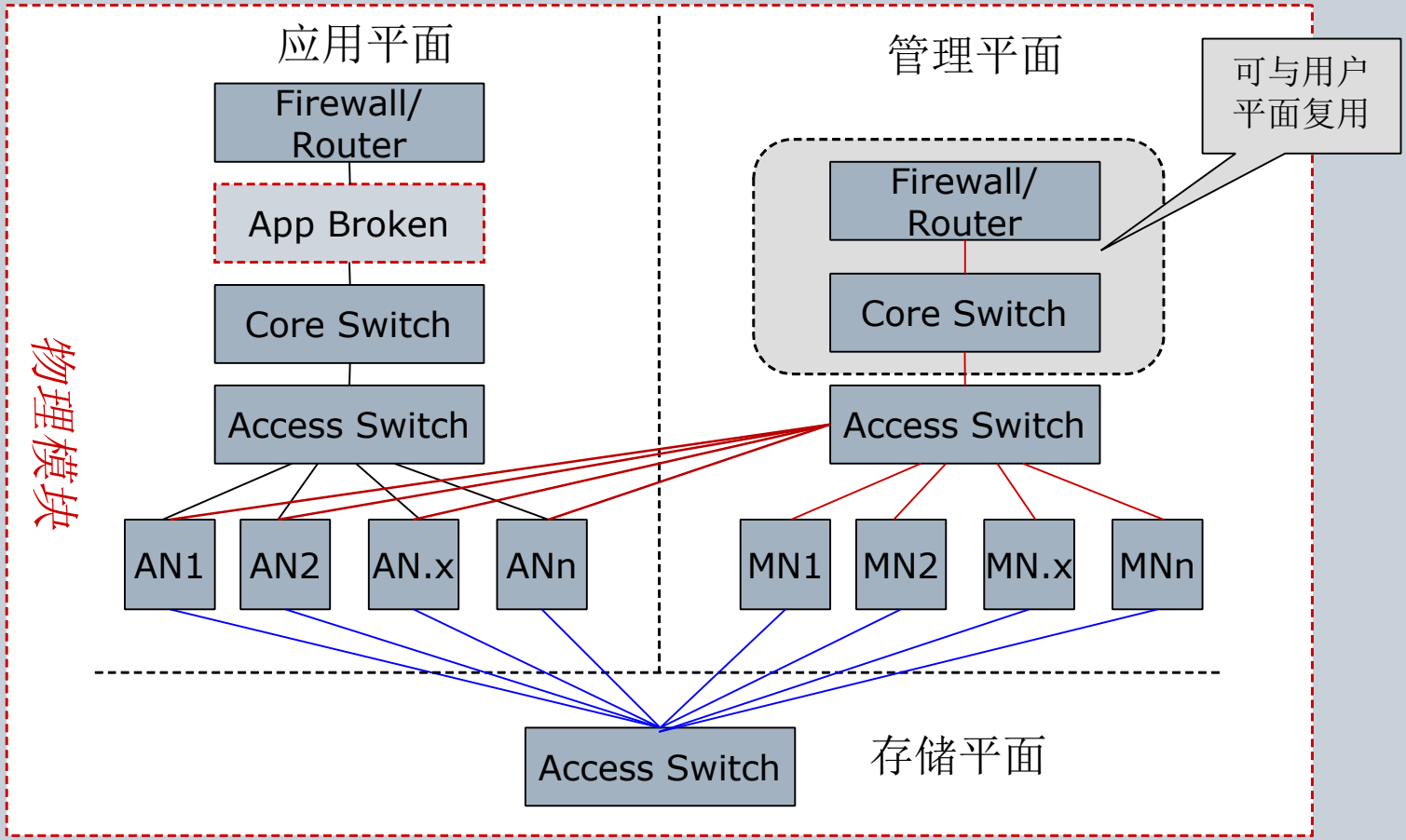
架构理念

- 充分利用成熟技术
  - 成熟，低成本；避免新硬件的引入，快速
  - 有没可能利用现有技术解决数据中心虚拟化带来的网络问题，避免思科,HP等国外巨头的技术垄断？？？？？
    - 比如智能网管？ Openflow?
- Simple Design
  - 复杂是系统万恶之源
  - 云计算IAAS是各种技术的综合，需要化繁为简
- 只解决系统当前问题
  - General solution需要引入大量新技术和硬件且工作量大；同时，其也是各方诉求的平衡，需要大量需求的输入，一定是一种中庸的Solution
  - 只为当前系统考虑架构，简化工作，快速部署
  - 为未来提供可能的扩展性

系统概述


App Deliver Plane
IAAS
V-App Infrastructure
App Mgnt
Mgnt Infrastructure

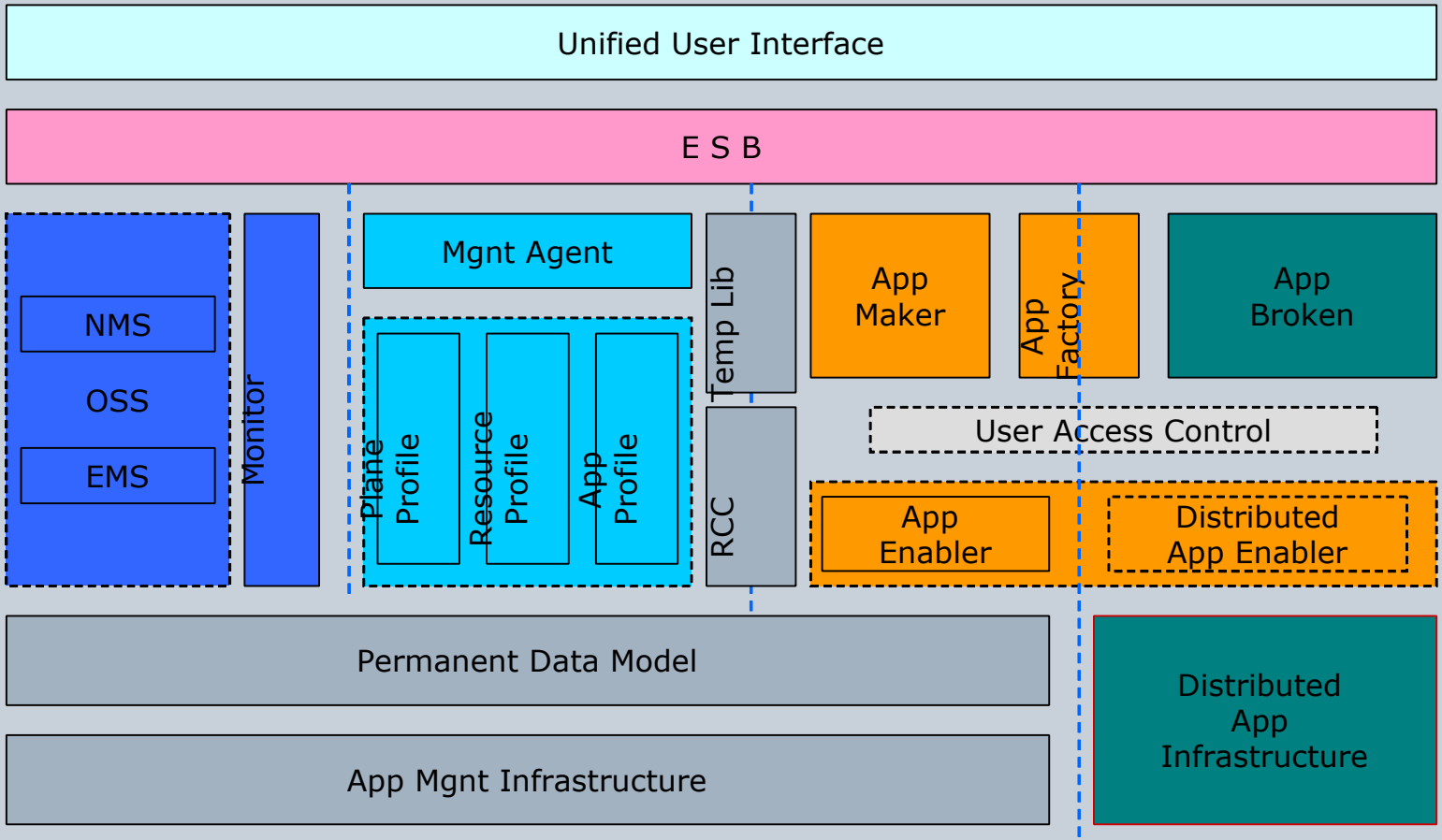
物理模块架构-三平面


应用平面
管理平面
可与用户
平面复用
Firewall/
Router
App Broken
Core Switch
Access Switch
Firewall/
Router
Core Switch
Access Switch
物理模块
AN1
AN2
AN.x
ANn
MN1
MN2
MN.x
MNn
Access Switch
存储平面

物理模块架构-三平面（续）

- 独立的管理控制平面
  - 对整个平台的管理与控制，如App的管理，如创建，删除，迁移等
  - Log，用户数据，App信息管理
- 独立的用户App平面
  - App与最终用户的交互
  - 用户负载均衡
- 独立的存储平面
  - 使用集中式存储IP-SAN or 分布式文件系统
  - 提高存储效率和安全性
- 物理 or 逻辑隔离
  - 独立的平面和网络，内部可见；提高系统的安全性

系统总体架构


Unified User Interface
E S B
NMS
OSS
EMS
Monitor
Mgnt Agent
Plane Profile
Resource Profile
App Profile
Temp Lib
RCC
App Maker
App Factory
App Broken
User Access Control
App Enabler
Distributed App Enabler
Permanent Data Model
App Mgnt Infrastructure
Distributed App Infrastructure

系统架构-用户类型

- Plane Administrator
  - 平台的配置与监控
  - App的预创建和资源策略的定义
  - App的配置与控制
- App Administrator
  - App的配置
  - App的控制
  - Component的管理与控制
- App Internal-User
  - 内部用户，如企业私有云开发测试服务
  - 可以访问App Component
  - 在小的App里，可以设置和App Admin拥有相同权限
- App External-User
  - 外部用户，一般是App 黑盒用户
  - 对开发测试私有云服务来说，没有这种用户，只有Plane Administrator & Internal-User

系统架构-四平面

- 独立的四平面机制，采用SOA架构
  - 松耦合，各平面独立发展
  - 可裁剪的四平面组合以满足不同的IT场景

- Mgnt Agent-系统管理代理，管理和监控平台（应用监管平面）
  - Plane Profile
    - 系统接口：路由策略，网络接口等
    - 迁移策略，安全策略，系统监控参数
    - 平台访问策略，用户控制，访问认证方式
  - Resource Profile
    - 系统网络资源：管理IP pool，内网IP，外网IP pool等
    - 系统RM，存储IP pool，存储资源
  - App Profile
    - App 类别，Image等；访问策略，安全机制等
    - 资源策略，冗余策略，网络资源，QoS/Policy策略等
  - 使用者：Plane Administrator

系统架构-四平面(续)

- App Maker（应用生产平面）
  - 根据App配置，完成App的创建，删除，修改，中止等
  - 维护App的属性和其对应的component
  - 维护App 生命周期及其Session的管理与维护
  - Factory的交互，存储
  - 提供App各Component的管理与维护
  - 支持可编程，开放API？
  - 使用者：App Administrator or Internal-User
- App Broken（应用消费平面）
  - App访问代理
  - Client/User 的访问控制
  - 接入负载均衡等
  - 使用者：最终User

系统架构-四平面(续)

- 系统支撑系统OSS & Monitor
  - OSS-Operator Support System
    - 主要包括NMS 和 EMS系统
    - 网元节点的信息收集，拓扑视图呈现
    - 网元的配置与管理
    - 网络流量的监控与管理，实时网络告警管理
    - 网络实时分析与呈现
  - Monitor
    - 监控主机状态，如主机CPU，温度
    - 监控VM状态
    - 监控各个控制单元
    - 实时缺陷捕获与告警
    - 为系统提供分析视图
    - 提供系统负载分析与处理接口

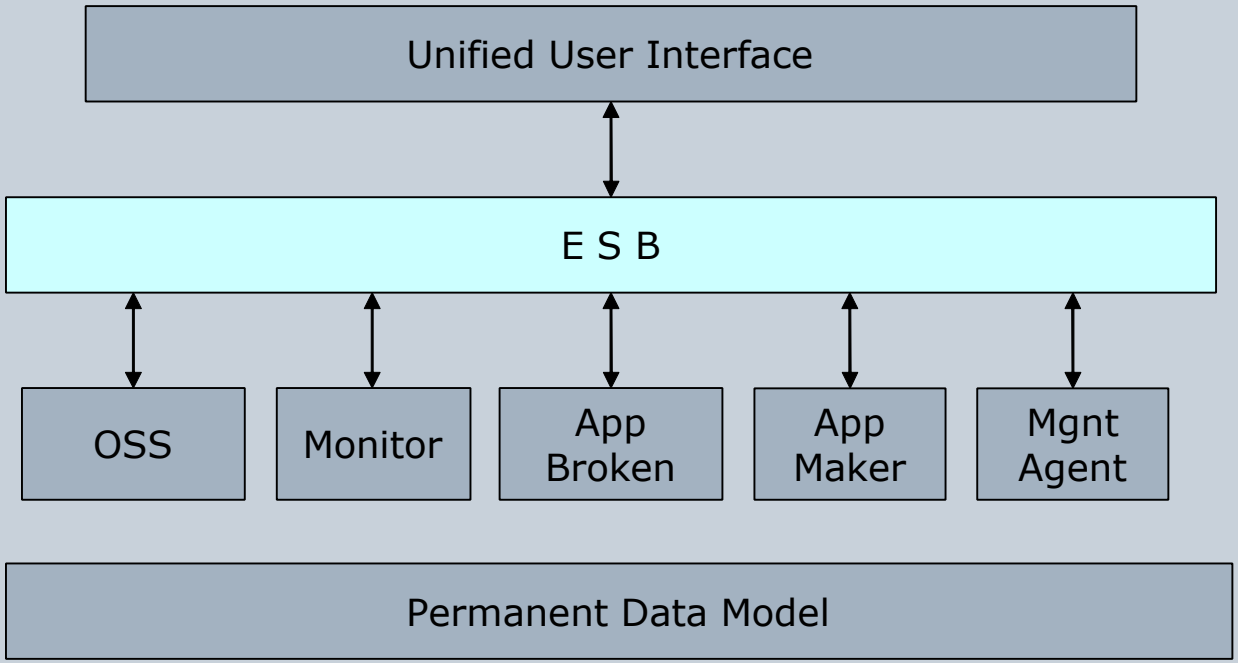
系统架构-UUI

- 平台管理与监控界面
  - 面向平台管理员
  - 平台配置界面
  - App预配置界面
  - 系统状态界面
  - 所有App信息界面
- App管理与监控界面
  - 个性化，面向App Administrator or Internal-User
  - App 详细配置界面
  - App控制界面：Enable，Stop，Release
  - App状态，信息界面
  - Component 状态，信息界面
  - Component 访问界面
- CLI
  - 符合IT人员使用习惯

系统架构-ESB

- 中间件，各组件的连接中枢
  - 消除系统组件间的耦合性
  - 为各组件提供事件驱动和标准接口
  - 考虑开源软件：Apache Synapse，Servicemix ？
- ESB下的事件生产者
  - UUI：Admin的管理与配置
  - OSS：网元拓扑发现与收集（包括VM）
  - Monitor：系统故障，告警与处理
  - App Maker：网络配置
- ESB下的事件消费者
  - OSS：App Mgnt的网络配置
  - Mgnt Agent：UUI的配置
  - App Maker：UUI的App管理

系统架构-ESB（续）


Unified User Interface
E S B
OSS
Monitor
App Broken
App Maker
Mgnt Agent
Permanent Data Model

系统架构- PDM

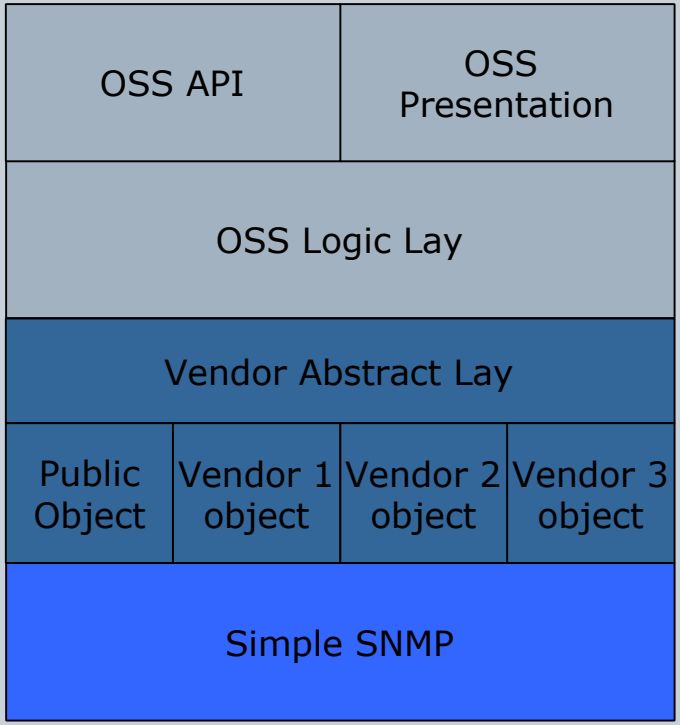
- Permanent Data Model维护系统的数据及数据间的关系
  - 配置信息
  - 网络信息，存储信息
  - 主机，VM信息
  - App 信息
- 提供数据访问的标准
  - 分解数据，简化系统设计
  - 响应其他组件的查询请求
- 采用key-value DB
  - 数据持久化
  - 内存数据库Redis

系统架构-OSS

- Operation Support System运营支撑系统，包括NMS，EMS
  - 类似网管系统
  - 网元节点的信息收集，拓扑视图呈现
  - 网元的配置与管理
  - 网络流量的监控与管理，实时网络告警管理
  - 网络实时分析与呈现
  - 支持SNMP Get/Set/Trap
- 挂载在EBS下
  - 事件生产者：自动收集网元信息，将收集的信息同步到RC
  - 事件消费者：响应网络操作，自动部署网络
- 系统配置
  - 系统基于openNMS + Twaver？
  - 提供Set接口？

系统架构-OSS（续）

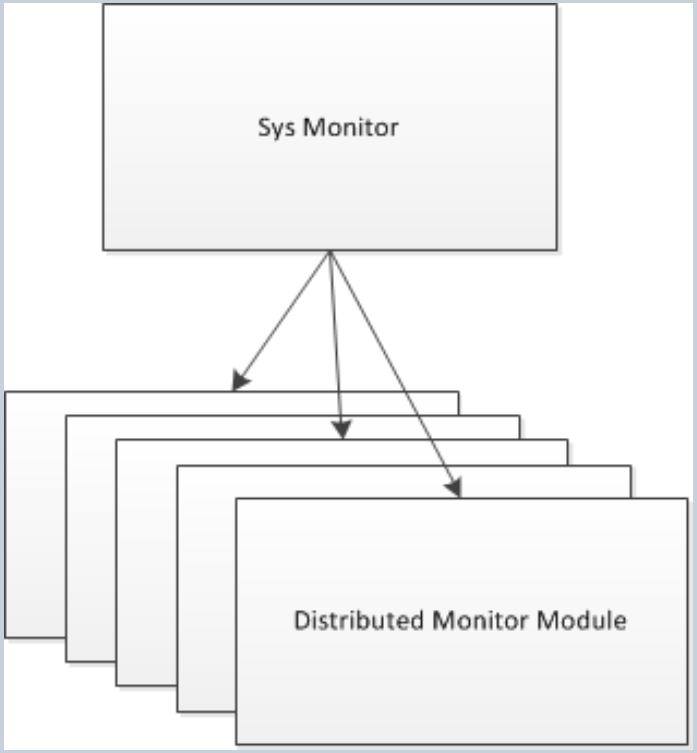
- 分层设计
  - OSS基于SNMP：地层技术简单SNMP；VAL用以屏蔽多个HW Vendor对不同功能的实现
  - OSS基于Open-Flow：底层为OpenFlow Control？
  - Logic层提供符合平台需求的逻辑实现
  - 上层为OSS API层和展示层，可以作为独立服务存在
- 简单设计
  - 先期只提供EMS
  - 只对有限的Object开发
  - 只提供对中国市场主流Vendor的支持，如H3P，Cisco，Dell



系统架构-Monitor

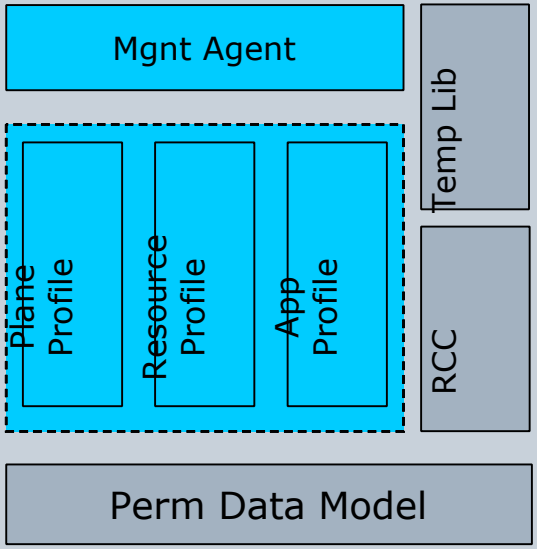
- 功能描述
  - 提供系统全局视图
  - 监控系统运行状态
  - 系统事件触发

- 分布式Monitor 架构
  - 基于内核的状态监控
  - AN节点有Monitor Module
    - 主机状态
    - VM状态
  - 集中的Sys Monitor
    - 负责系统全局信息的收集
    - 提供系统全局视图
    - 提供Alert，事件接口
    - 提供系统交互的接口



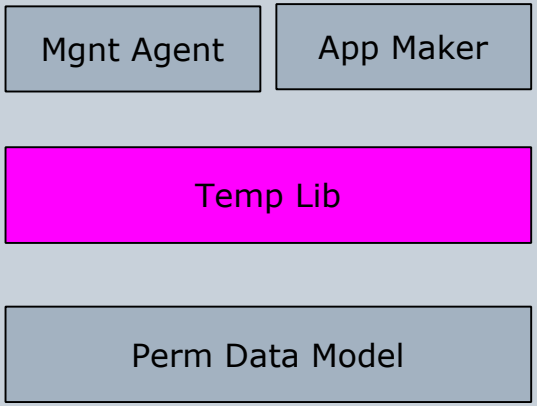
系统架构-Mgnt Agent

- 提供平台管理接口，不生产App
  - Plane 配置
    - 系统网络接口
    - 网络属性配置，外部网络接入策略，负载均衡策略
    - 系统备份策略
    - App 调度策略，如是优先级，调度算法，可抢占等
  - Resource 配置
    - 网络资源；存储，VM资源
    - 系统资源配置和策略（如资源预留），capability 配置（x VM per host？）
  - App 配置
    - App template
    - Component template & image
    - 访问方式：域名 or IP
    - App Resource配置
    - App 备份策略，冗余策略， 生命周期
    - 访问策略，安全机制
    - QoS， Policy，优先级



系统架构-Temp Lib

- App&Component Templent的管理与维护
  - Component Templent
    - 类似Virtual Appliance
    - 一般为一个Image 映射
  - App Templent
    - Component 的组合
    - App 即为多个Comp 的组合
    - CT与AT为一对N的关系， N>=1
  - CT遵循OVF or AMI？
    - 遵循标准，降低App部署，许可，认证管理等的复杂度
    - 为后续的开放API 提供可能性
    - 可以编辑，Platform Admin可以管理这些Templent
    - App Maker 可以直接使用Templent组成App

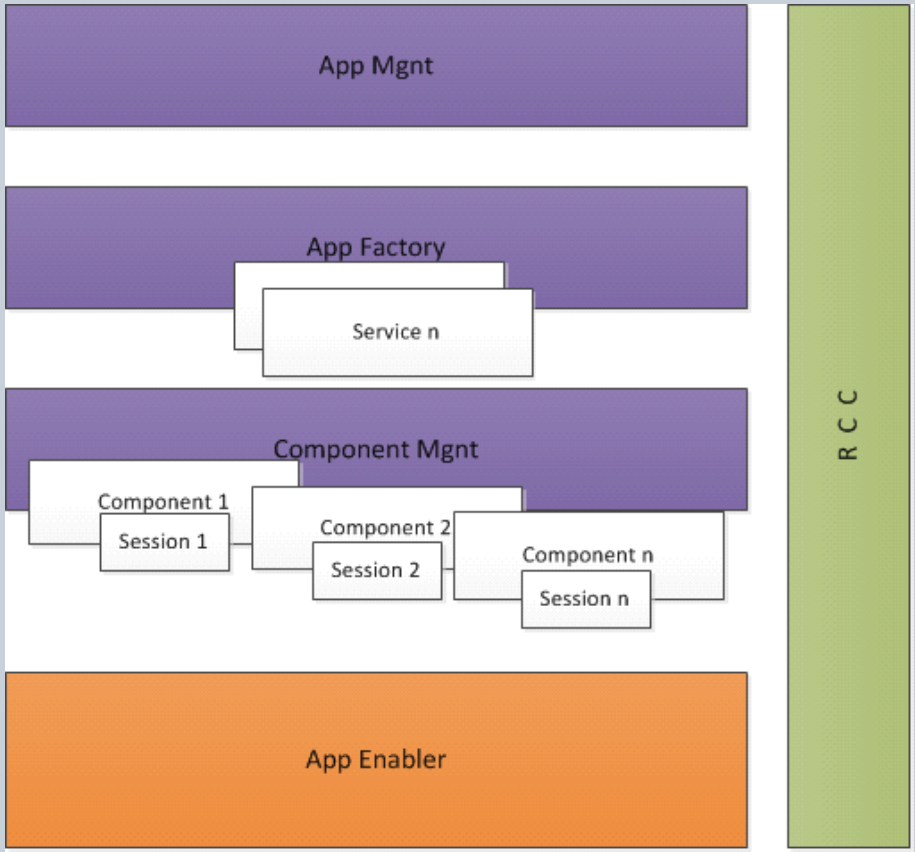


系统架构-App Maker

- App界面与接口
  - App事件代理，创建，删除，修改与维护，编辑
  - 支持App整体控制，提供基于App的管理接口
  - 支持单个Component的控制，提供App 各Component的管理接口
- App Mgnt功能
  - App component组合
  - 管理/配置App；开发新的App等功能的模块
- Session 监控与管理
  - Session监控，创建，维护，删除等

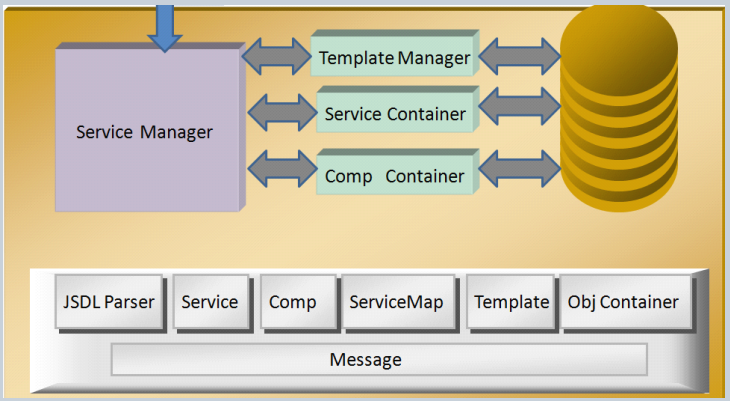
  Session 起点，终点？
- 用户为App Admin，对小型业务平台
  - Mgnt业务流一般并发性低，故不需要Broken
- 对大型业务平台（未决定实现）
  - App繁多，Mgnt前置Broken
  - 提供App级别并行处理与Loading-Balance(基于Control Session)
  - 与分布式App Enabler协同，可以提供两级并行处理

系统架构-App Maker（续）



系统架构-App Factory

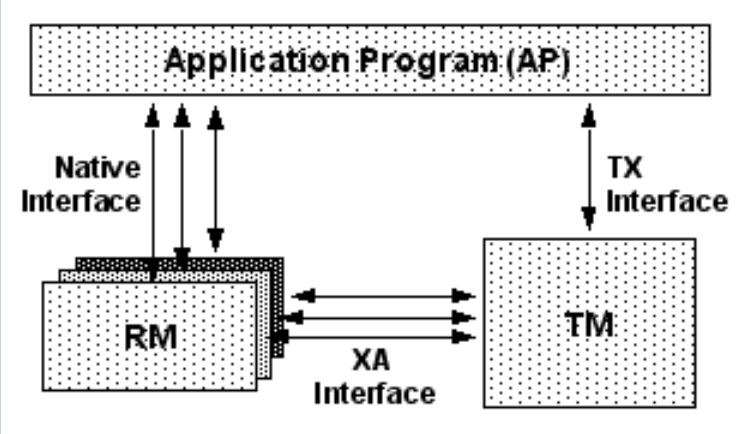
- App Factory
  - App 容器
  - Component 管理
  - App 信息管理与容灾备份
  - App生命周期管理
  - App保存/加载
  - App状态/配置信息等的收集与维护
  - App信息查询

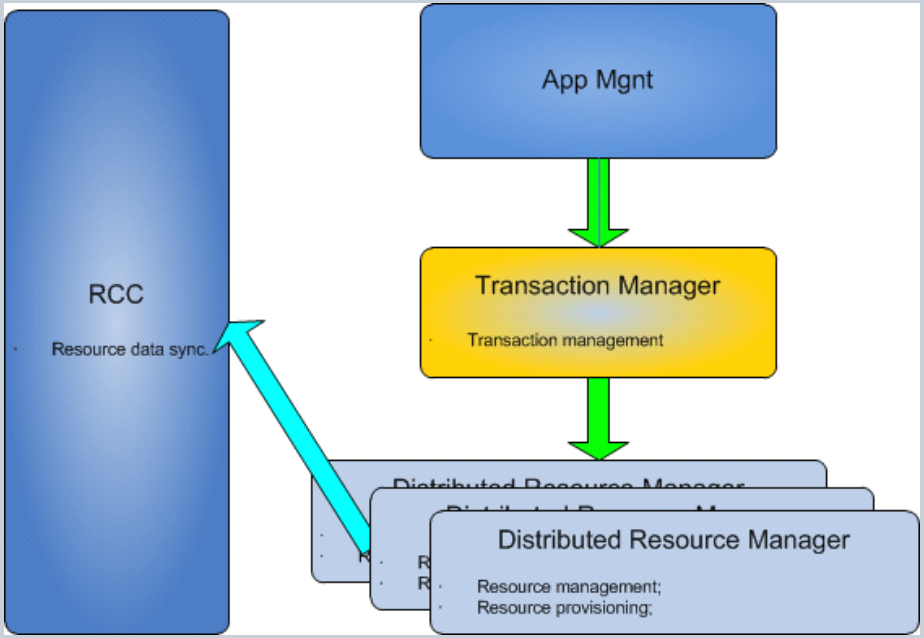


系统架构-App Enabler

- App 使能执行体
  - 无状态执行体
  - 负责事务的处理与分发
- AE 参考X/open DTP分布式事务处理模型：
  - TM 集中处理事务的管理与分发
  - RM 分布在 Distributed App上，处理各系统的事务
- TM 可实现分布式事务处理（未实现）
  - 各Component使用不同的TM处理事务
  - 提供二级平行处理
  - 集中式的RC保证资源的原子性(有点像新的Hadoop？)

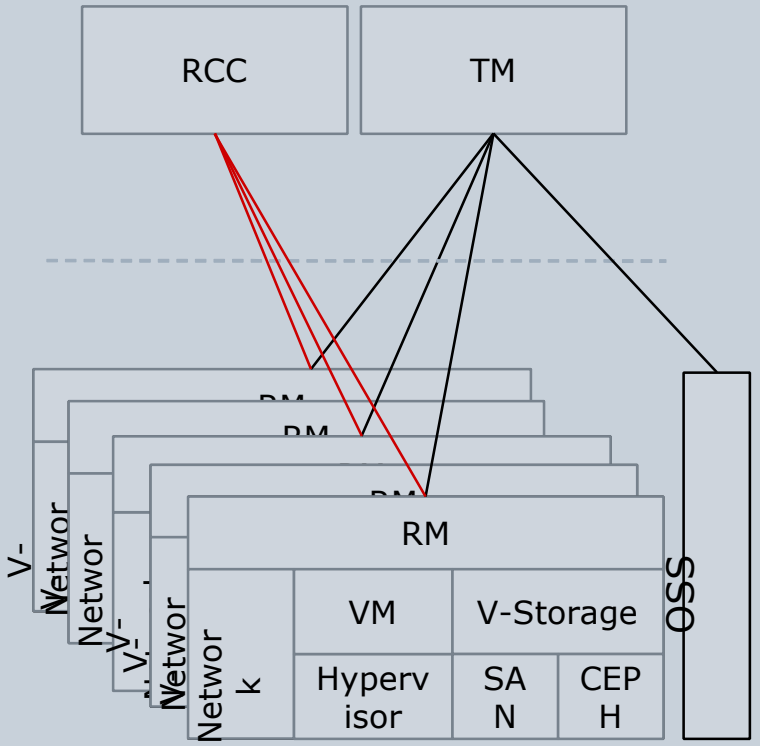
系统架构-App Enabler（续）





系统架构-DAI

- TM集中的事务处理
  - 事务分发
  - 事务调度
- 分布的资源控制和事务处理RM
  - 处理资源请求，与RM交换，资源收集
  - 处理事务
- V-Network
  - 使用open v-switch
- VM & Hypervisor
  - 需要重可用文件系统， IO等
  - 支持KVM+QUEM / XEN
  - 使用Libvirt ？
- V-Storage
  - 支持分布式文件系统CEPH
  - 支持IP-SAN HBA, 使用GFS or Lustre

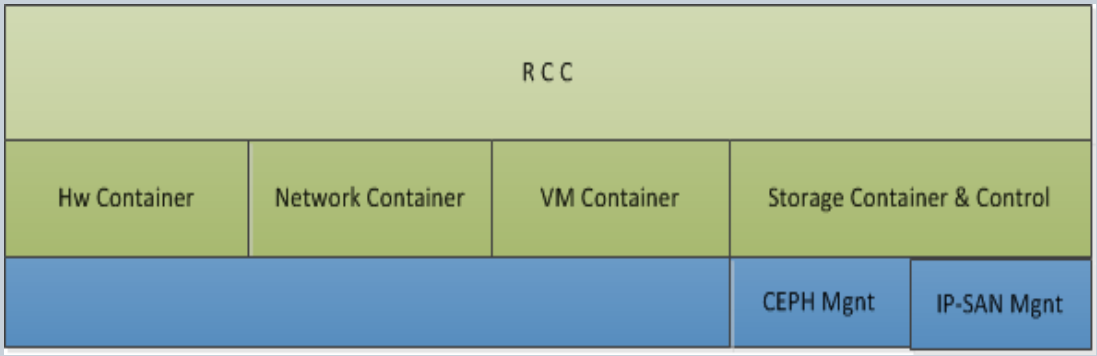


系统架构-RCC

- 集中的RC-Resource Container
  - 资源集中收集
    - CPU，内存，网卡，存储等硬件信息
    - 系统，版本信息，物理位置等
  - 资源自动发现
    - 自动发现硬件引入，即插即用
    - 响应系统Monitor事件，实时处理资源变化
- 采用资源预留方式
  - 可为App动态预留资源
  - 为App服务
- 考虑与Monitor合并
  - 两者有相似的架构？

系统架构-RCC(续)

- 集中的资源策略执行-Resource Control
  - 集中执行资源分配策略
    - 根据App资源需求和物理机状态
    - 根据App优先级，同一物理机分配给不同优先级
    - 根据App运行时间状态，分散时间高峰
    - 根据App调度策略
  - 体现系统的调度策略
    - 体现App的优先属性
  - 保证资源的原子性
    - 集中式锁，非抢占



系统架构-App Broken

- App Broken为User访问代理
  - 提供访问代理服务和负载均衡服务
    - 通过App profile配置
    - 提供外网，内网的转换
    - 提供用户负载均衡策略
    - 可是使用开源软件替代， 如LVS
    - 可是使用F5
  - Internal-User
    - 访问App Factory获取App信息
    - 获取Component访问方式
    - 访问控制和认证
    - 用户直接访问或者负载均衡
  - External-User
    - 黑盒用户
    - 提供域名访问服务？

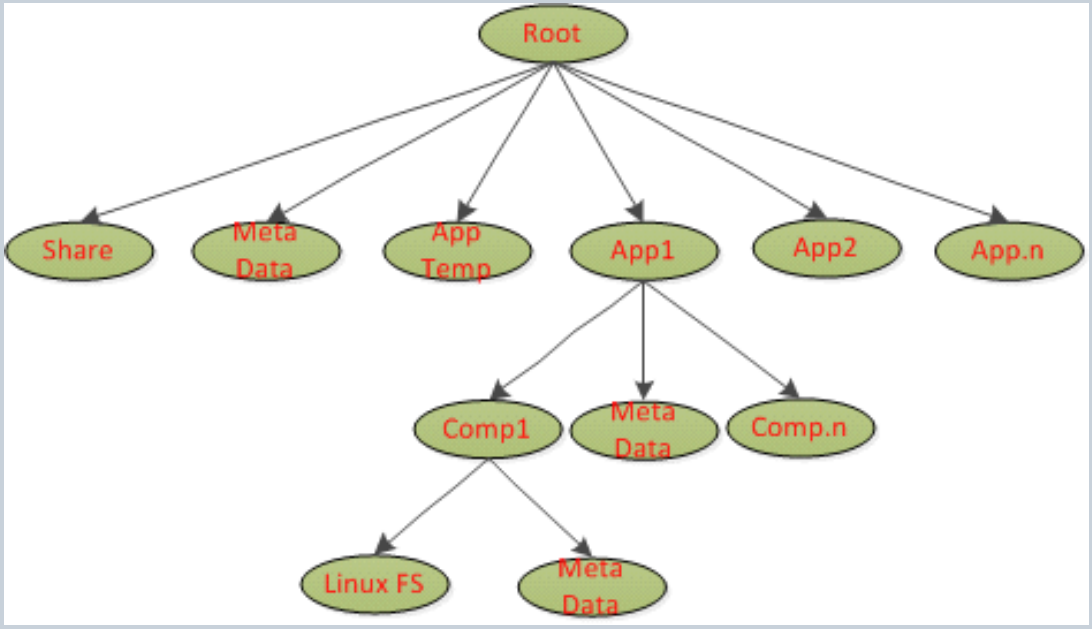
系统架构-存储

- IAAS需要集中存储方案
  - 方便VM动态部署，用户数据与位置无关
  - 方便Image管理，集中管理App，Comp模板
  - IP-SAN/iSCSI-SAN or other？
    - 可以使用HBA卡提高性能（待考虑？）
  - 分布式数据库
    - CEPH or HDFS？

- 部署方式
  - 部署在管理平面
    - 应用平面安全
    - 带宽要求高，系统有瓶颈，不能随App扩大而Scale-out
  - 部署在应用平面
    - 离应用最近，更高性能
    - 随App扩大而Scale-out扩展
    - 资源利用率高

系统架构-文件系统

- 文件组织方式
  - 以全局方式管理
  - 一个App对应一个名字节点
  - 提供安全访问机制，App不能访问不同App的节点(不存放用户数据？？？)

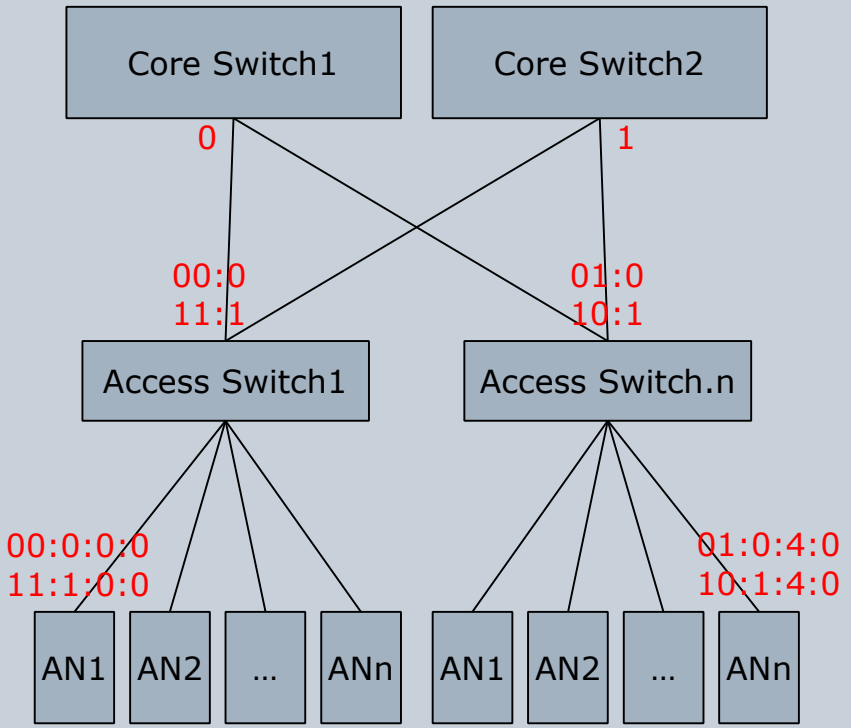


系统架构-硬件环境

- 只部署在机架式服务器集群中
  - 刀片Switch难以控制
  - 采用大管道网络模型，控制点都在OVS/Core 上，可在刀片上部署
- 硬件加速
  - 使用HBA卡提高IP-SAN能力
  - 可是使用SSD提高存储能力（到100M/S ？）
  - 使用IOV卡（VT-C）提高虚拟机交换能力
- 硬件异构策略
  - IP-SAN与CEPH不共存
  - 在使用IP-SAN情况下，HBA卡与普通存储网卡不共存
  - Open V-Switch 与 IOV卡不共存
  - 简化系统Resource 管理

系统架构-统一位置标识和名字空间

- 系统自动部署需要统一的Location标识和名字服务系统（DNS？）
  - 采用多位置信息描述符
  - 每个位置信息描述包含双重信息
    - 上行信息
    - 下行信息
    - 采用16进制
- 举例：
  - AS1有两个位置信息：
    - 00标识设备处于第二层
    - 00:0标识上行core 1的0号端口，使用自己的0号端口
    - 11:0表示上行core 2的1号端口，使用自己的1号端口
  - AN1也有两个位置标识：
    - 00:0:0表示设备处于第3层
    - 00:0和11:1集成自上行设备
    - 0:0标识上行0号端口和自己的0号端口
    - 可以由AN1节点位置找到所有路径



系统架构-开发环境

- 分层设计
  - 分布式App Node使用C， C++
    - 减少多应用环境资源的消耗
    - 提升系统性能
    - 如Rc， Monitor，v-swtich
  - Mgnt Node使用OO语言Java，C++
    - 重用现有系统框架
    - 更好的匹配web界面
  - 多平面接口
    - 减少环境复杂度；保持语言中立
- 尊重架构， 同时尊重人员能力和喜好
  - 保持架构一致性
  - 降低人员需求

网络面临的挑战与应对

Ronnie.shi79@gmail.com

VM动态迁移下的网络架构

- VM的动态迁移的优势
  - 系统故障恢复，提高冗错能力
  - 动态平衡网络/计算热点，优化计算资源
  - 优化电源管理，降低能耗

- VM迁移要求No-Stop-App，用户零感知
  - 网络对用户透明，无变化
    - IP/MAC/QoS/安全策略等VM网络资产和属性不变
  - 网络等其它资源跟随迁移
    - VM对应的大部分网络属性和资产跟随动态迁移包括

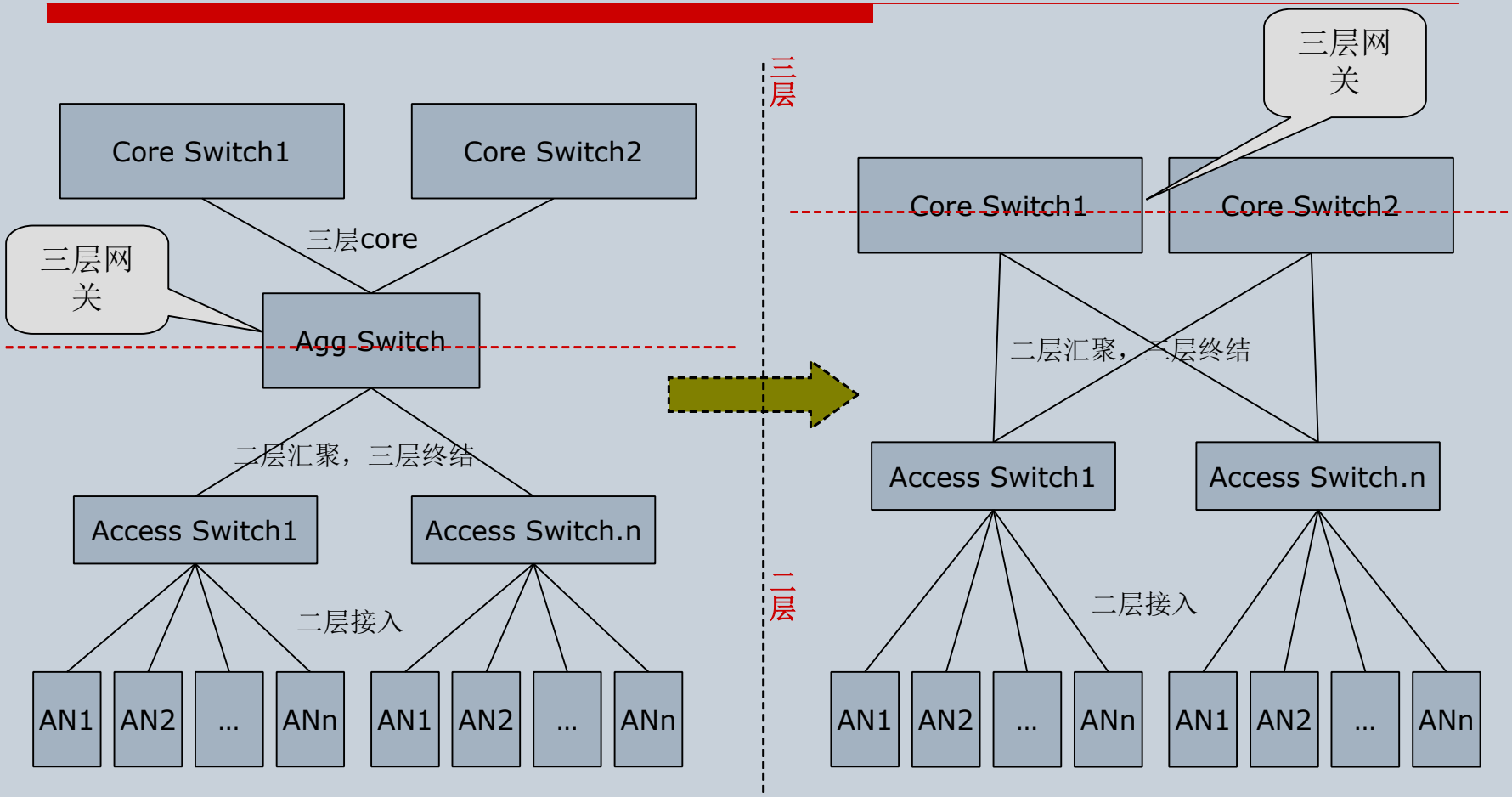
- 企业数据中心里二三层网络之困
  - 二层网络：使用简单，方便，缺乏安全性和扩展性，适合小网络和边缘网络。部署在接入和汇聚层
  - 三层网络有更好安全性，扩展性，冗余能力和恢复能力，但更复杂，更高成本。部署在核心和汇聚层

VM动态迁移下的网络架构

- 三层网络因其安全，网络隔离，扩展性而部署在网络汇聚层和核心层，意味着：
  - VM迁移其网络属性不能发生变化，故迁移只能发生同一子网段内。意味着只能在接入层迁移
    - 迁移范围小，灵活性差
    - 不能整个数据中心动态迁移，冗余能力和负载均衡能力差
    - 网络部署复杂，扩展性差

- 使用大二层架构
  - 减少网络层次，去除中间的汇聚层
    - 突破汇聚层约束，扩大VM迁移范围
    - 简化网络架构，去除不必要的二三层复杂性
  - 使用大二层架构
    - 提高收敛比，增加内部带宽
    - VM整个数据中心内动态迁移

VM动态迁移下的网络架构



VM动态迁移下的网络架构

- 大二层架构的要求
  - 接入交换机
    - 上行高带宽
  - 核心交换机
    - 高密度
    - 高带宽
    - 支持复杂的三层功能和安全策略
- 已有网络设备的限制条件
  - 使用既有设备，降低成本
  - 使用LACP提高上行带宽
  - 控制App 数目，降低密度要求

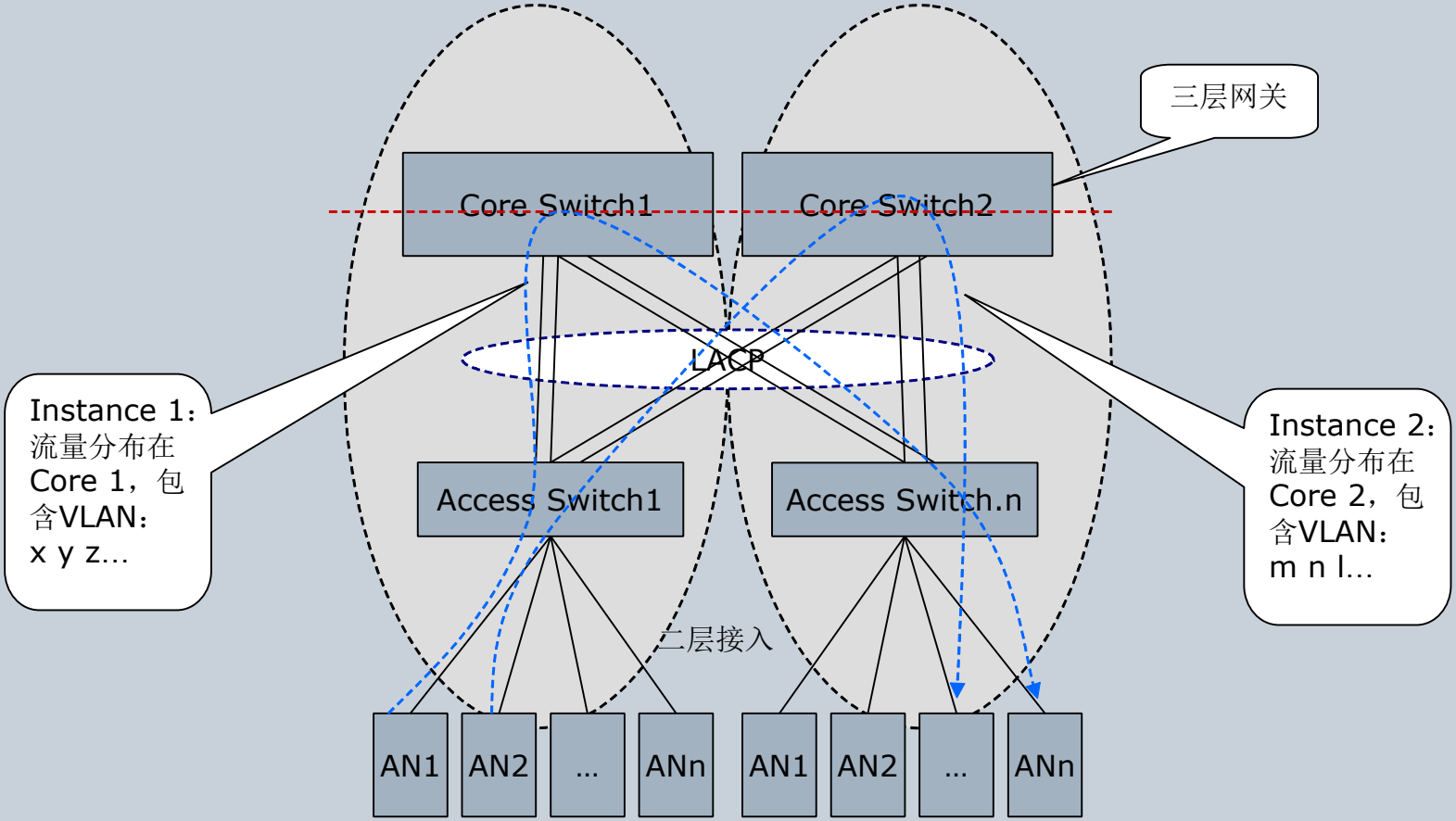
横向流量 & 二层多路径

- 横向流量问题
  - 传统网络的CS模式
    - In-Out 网络流量模式
  - 私有云平台的App间交互流量大增
    - 分布式处理需要App间的通讯
    - VM迁移下的内部拷贝需要高带宽保证
    - VM动态迁移需要集中式存储或者集群文件系统，App App对集群文件系统的访问，极大地增大了内部数据访问的需求
- 解决之道：二层多路径技术
  - 提供Active-Active多路径技术
    - 增大S2S 的带宽

横向流量 & 二层多路径

- 二层多路径的困境
  - 大二层网络模型下，经典的三层多路径技术ECMP失效
    - VPN等建立在三层路由基础之上多路径技术，在大二层网络架构中同样实效
  - 二层冗余技术STP未能充分利用网络带宽
    - Active-Standby模式提供网络冗余
    - 网络切换慢
  - 在新型数据中心里，二层多路径是个新兴热门技术
    - 各厂商分歧大， Trill Vs SPB，没有统一标准
    - 新技术成本高昂
- 我们的解决方案：MSTP+LACP+VLAN规划
  - 充分利用现有技术，降低设备成本
  - 利用MSTP实现二层多路径
  - 利用VLAN分布提供粗粒度负载均衡能力
  - 合理规划网络，使负载尽量均衡

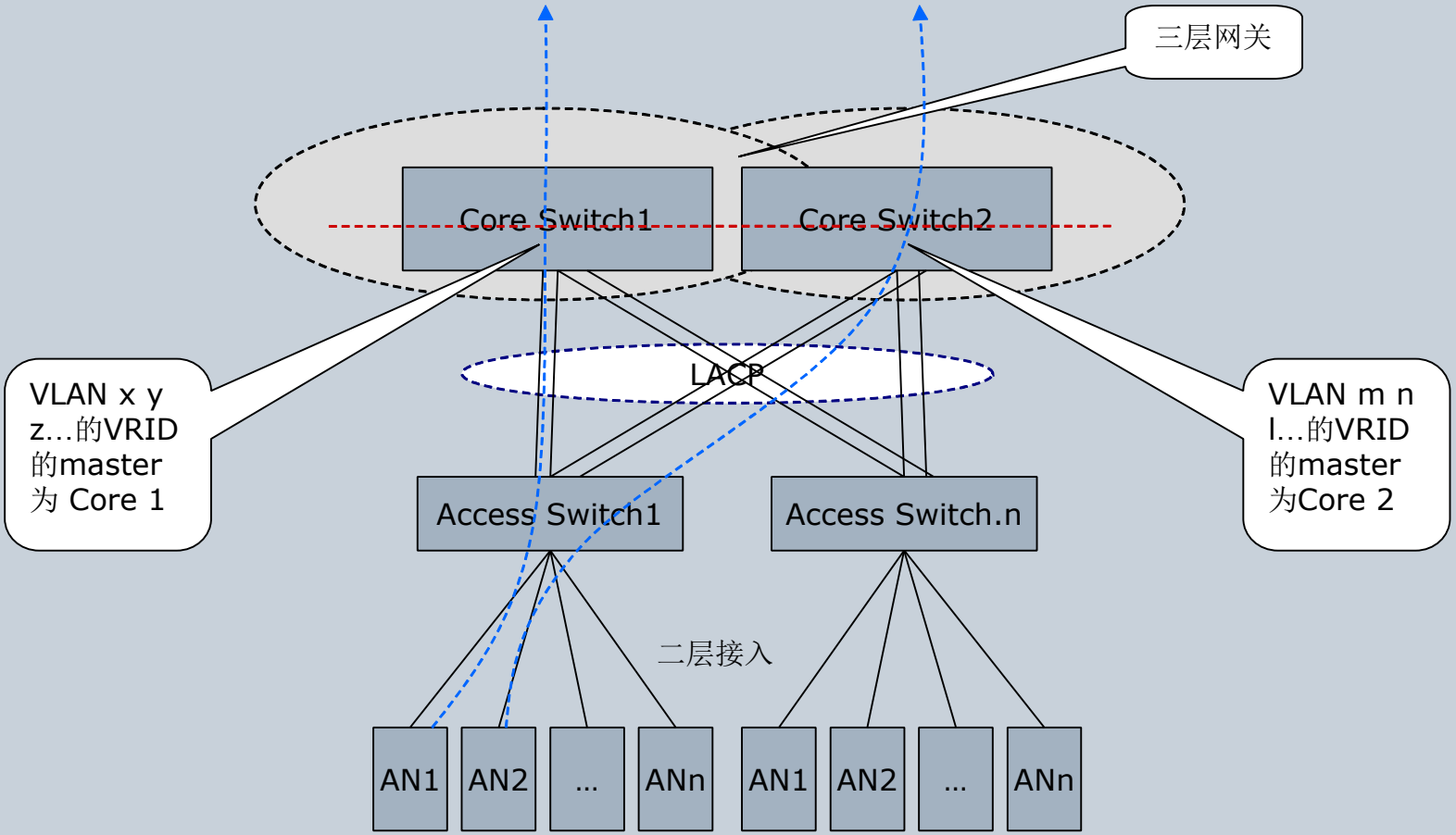
横向流量 & 二层多路径


三层网关
Core Switch1
Core Switch2
LACP
Instance 1：
流量分布在
Core 1，包
含VLAN：
x y z...
Instance 2：
流量分布在
Core 2，包
含VLAN：
m n l...
Access Switch1
Access Switch.n
二层接入
AN1
AN2
...
ANn
AN1
AN2
...
ANn

网关冗余和负载均衡

- 外部网关负载均衡和冗余
  - 私有云承担各种应用，高并发性和大规模用户要求负载均衡
  - 系统可用性要求网络冗余
- 跨网段负载均衡与冗余
  - 系统内部可靠性要求系统冗余
  - 系统内部不同应用的高速交互需要跨网段负载均衡
    - 对Ceph文件系统的访问
  - 关键组件的交互需要提供冗余
    - 文件系统，数据库
- 解决方案
  - 放弃VSS，IRF2，user均衡等复杂技术
    - 更精细的均衡粒度， scale-out的扩展性
    - 更快故障恢复时间
  - 使用VRRPE/HSRP成熟基础
    - 提供粗粒度，低成本的负载均衡和冗余能力
    - Per VRRP/Per VLAN，扩展性差，难管理

网关冗余和负载均衡


三层网关
Core Switch1
Core Switch2
LACP
VLAN x y z...的VRID 的master 为 Core 1
VLAN m n l...的VRID 的master 为Core 2
Access Switch1
Access Switch.n
二层接入
AN1
AN2
...
ANn
AN1
AN2
...
ANn

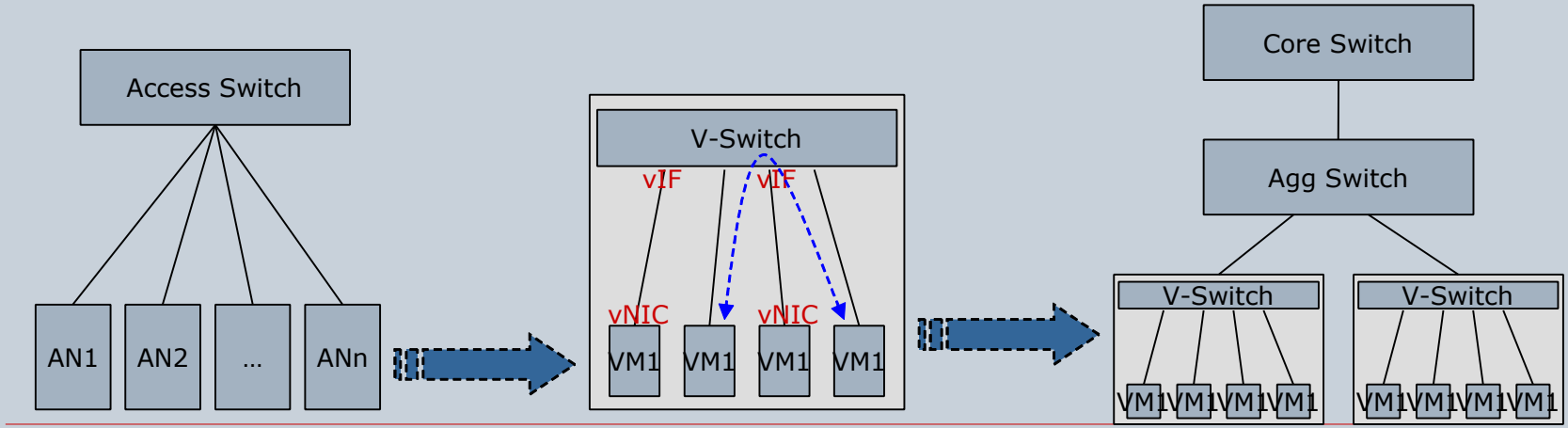
QoS 模型

- Plane Network QoS 模型特点
  - Plane对用户的交付为App App，Network作为App支撑系统，用户零感知
  - Network QoS模型用户不可见
  - Network QoS模型丰富，怎么定义与选择？其与manage model的关系是什么？

- Network QoS 与 App 匹配
  - 根据网络设备情况，定义固定QoS模型
  - App QoS模型可以参考Network QoS 模型定义成固定模型，生成几类模板
    - How？需要大量用户场景数据
  - Network QoS和App QoS模型间保持固定关系
  - 不同的App 可以选用不同的App QoS模板
  - 待定？？？

VS问题

- Access下移；可管理/可视性，一致性差
  - 内部流量，状态信息不可见
  - 与其他网元难保持一致
  - 硬件虚拟化VT-C(IOV网卡)不可见



VS Solution

- Access下移，VS内置ANMP Agent（preferred）
  - VS提前部署，将其做为Access物理机看待
  - 网络统一管理，明确清晰
  - VS需要支持较完善的二层功能
- 构建DVS（distributed VS）
  - 各网元统一考虑
  - 非常复杂
- VN-Tag or VEAP or QinQ?
  - 需要更新硬件
  - 标准未定，价格昂贵

网络Capability

- 系统扩展性
  - 两台Core Switch系统，在采用48口交换机情况下，可以接入960台物理App，可以应付目前大部分的需求
  - 最多可以扩展到4台Core，同样配置下，系统可以接入1920台物理App
  - 大二层网络，系统平滑扩展能力差，管理配置无自动扩展能力，需要手动规划
- MAC地址表
  - 按最大2000个物理App，12个VM计算，系统最大MAC地址数目为24K，超出目前主流交换机能力
  - 考虑负载均衡，子网MAC隔离等因素，假设4：1的收敛度，每个交换机6K MAC地址，可以接受
  - 系统RC，Data Model负载均衡需要充分考虑网络均衡
- VLAN数目
  - VLAN可以用来隔离不同业务，单系统App App数目应该很难超过4K

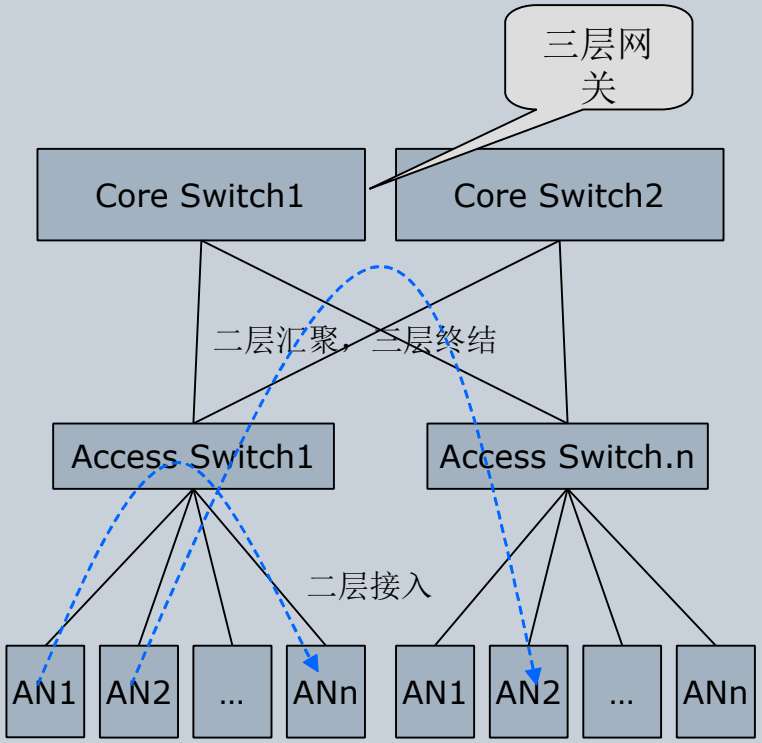
网络自动部署

- 原则
  - Simple Designed
    - 降低系统复杂度和系统开销
  - 静态配置最大化，减少动态部署与迁移
    - 初始态：一次配置
    - 运行时：有限自动迁移
- 网络架构的自动部署
  - 网络异构情况下，使用ANMP完成网络部署
  - 采用网络配置模板，完成大部分网络的首次部署
  - 新网络设备的加入/退出，手动修改 or 新的模板？
- 全静态策略
  - App预先规范与配置？
  - 网络预先规划好，做网络资源预配置
  - 非动态分布与部署
  - 存在路由黑洞

网络动态部署

- VM部署/迁移情况
  - App内迁移（意义？）
  - 交换机内迁移
  - 跨Access Switch迁移

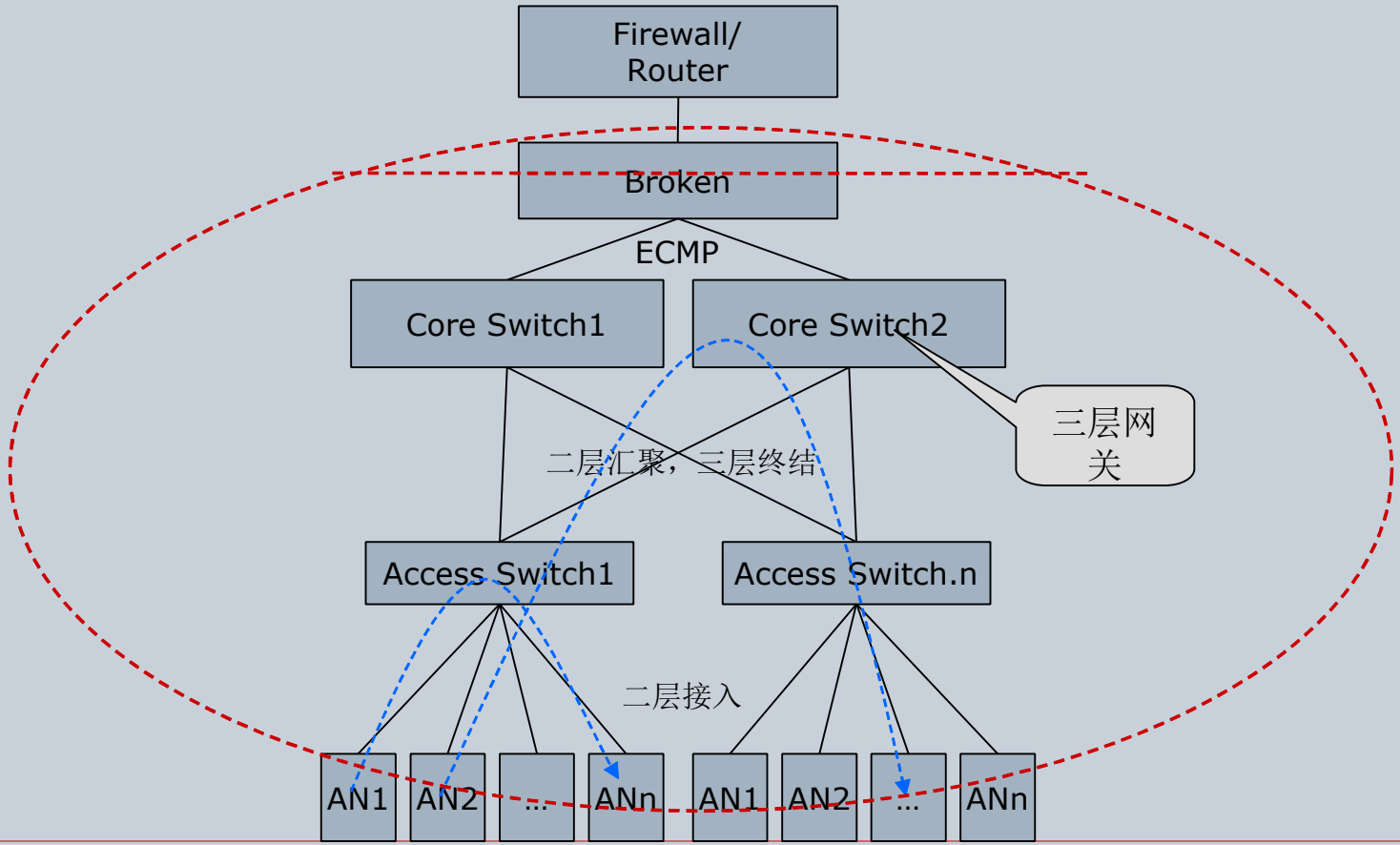
- 问题：
  - 显然，VM的MAC需要刷性，VLAN等需要同步迁移
  - 不同情况，需要刷新和迁移的物理路径不同，需要有系统有全局view



网络动态部署/迁移

- MAC地址刷新
  - 通过App Enable 触发OSS系统刷新MAC表
    - 复杂，效率低
  - VM自动刷新（preferred）
    - 借助VM的DHCP or GARP自动刷新网络MAC
- VLAN部署/迁移
  - 通过App Enable 触发OSS系统配置VLAN路径
    - 复杂，大系统下，必须自动更新VLAN路径
  - 大二层Hub模式
    - 接入交换机之上Trunk all VLAN， 做大管道模式
    - 网络隔离，VLAN控制交给VS，通过App Enable/OSS实现
    - 简单，部分解决硬件交换机自动配置问题
    - 网络广播风暴等，缺乏安全
    - 在小规模网络情况下使用
- 三层网关和出口
  - 需要OSS在硬件Core动态部署
  - 有可编程控制的Core吗？ Or 三层再次上移到可编程的Broken？

系统网络接口

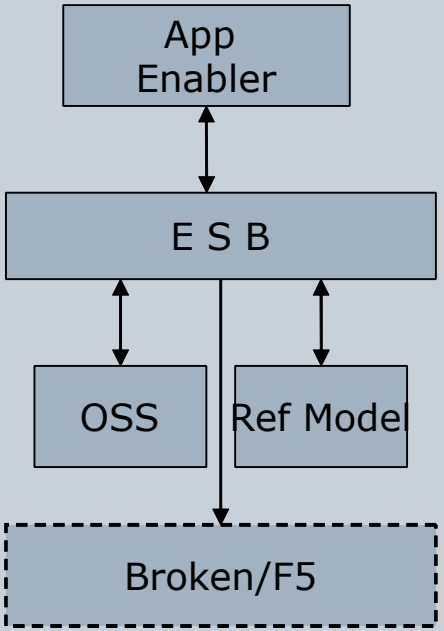

Firewall/
Router
Broken
ECMP
Core Switch1
Core Switch2
三层网
关
二层汇聚，三层终结
Access Switch1
Access Switch.n
二层接入
AN1
AN2
…
ANn
AN1
AN2
…
ANn

系统网络接口

- Broken 与外网Firewall / Router接口
  - 必须使用三层策略
  - ECMP or LACP多链路技术
  - 外部路由引入和内部路由发布
  - 对公网业务，使用IP访问代理
  - Plane Profile 控制路由策略
  - App Profile控制IP路由，访问隔离策略，代理策略
- Broken 与Core 的内部互联
  - 三层OSPF互联
  - 至少两条路径， ECMP多路径技术
  - 由Plane Profile & App Profile控制
- Core的三层属性
  - 三层网关终结
  - ECMP 与 F5互联
  - 由App Profile控制网关路由

IP/VLAN/Broken管理

- Plane平台采用静态IP/VLAN管理策略
  - Plane可用IP地址段由用户规划，实现与用户现有网络的无缝接入
  - App规模由用户规划，需要使用的IP地址池用户最清楚，且保证在系统整个IP Pool里获取，方便规划网络路由
  - Component的IP采用静态策略，在App Pool里获取
- IP对应的路由，VLAN配置
  - App Enabler驱动
  - 通过OSS自动配置完成
- Broken的管理与配置
  - 网络设备，与Switch一样，可以通过ANMP-OSS配置，提供一致性的网络架构？
  - 如果使用F5，提供单独接口管理（不支持）



MAC地址管理

- 动态MAC地址策略
  - MAC地址由RC集中分配
  - 管理复杂
- 可变MAC地址策略
  - MAC地址由位置决定（见系统位置标识）
  - VM迁移，MAC随着变化
  - 很好的体现VM的位置信息
- 静态MAC地址策略（Preferred）
  - MAC地址前缀可配置
  - 与IP地址绑定
  - 简单实用

其它问题

- 有没有其它的网络模式？
  - 绕开大二层架构；将二三层负载均衡和多路径技术直接透到可编程的Broken 来实现， 中间为大管道模式
  - 有没有支持可编程接口的Core 设备来避免OSS的复杂引入？
- 存储网络问题
  - 怎么与App网络隔离？ 物理 or 逻辑？
  - 如果使用独立网络，怎么负载均衡？（可是使用多个网段）
- IOV卡问题
  - 网络可编程能力？