GOPS2016 Beijing

2016 全球运维大会

DevOps 2.0：重塑运维价值

北京站

会议时间：12月16日 - 12月17日

会议地点：北京国际会议中心

主办单位：高效运维社区 GreatOPS Community　开放运维联盟 OOPSA Open OPS Alliance

十 问

--- 银行私有云建设若干问题

周海鹏 中信银行 云计算架构师

目录

1 银行IT架构中，云平台怎么定位

2 云平台建设中，厂商能贡献什么

3 私有云建设失败的原因

4 “自主掌控”到什么程度

5 大规模云的维护问题

6 私有云建设的技术风险有哪些？

7 私有云是否向开发开放自服务

8 数据库是否能运行在虚拟机上

9 云的最后一公里，分布式数据库

10 PaaS，你准备好了吗

About Me

- **周海鹏**，中信银行数据中心云平台架构师，负责云平台、虚拟化、存储管理、灾备建设。曾供职于IBM和华为，做过开发、测试、现场支持、800电话支持、系统运维、应用运维。

- 不但要做技术，更要做技术管理

1

银行中，云平台怎么定位

吴伯凡的“云战略”思维

01 格局，决定结局

02 多一个维度看世界

03 体系化竞争

04 善良比聪明重要

银行中，云平台怎么定位

云平台不仅仅是虚拟化，

也不仅仅是openstack平台

云平台是新型的**数据中心基础架构**，涉及到服务器采购、存储资源池建立、网络双活和SDN的实施、虚拟机的自动安装部署、应用的双活部署等**基础架构的重新设计**

云平台涉及到从**实体机的运维模式到云化运维模式的转变**，影响到我们平时的配置管理、变更管理、应急管理等等方方面面工作

2

银行云平台建设中，厂商能贡献什么

“罗辑思维”的逻辑

01 做时间的朋友

02 年轻人在大城市，能获得的最更重的东西是，见识

03 大部分实践工作中，行动力都比思考力重要

04 当一个事情复杂、新奇和不确定的时候，我们说：有趣；

而当一个事情简单、稳定和确定的时候，我们说：愉悦

云平台建设中，厂商能贡献什么

咨询

- 核心产品、眼界、信心、定力

产品

- 稳定、高可用、一键式接入、预防网络风暴、开放

实施

- PMO、洗脑、定制化、落地、汇报

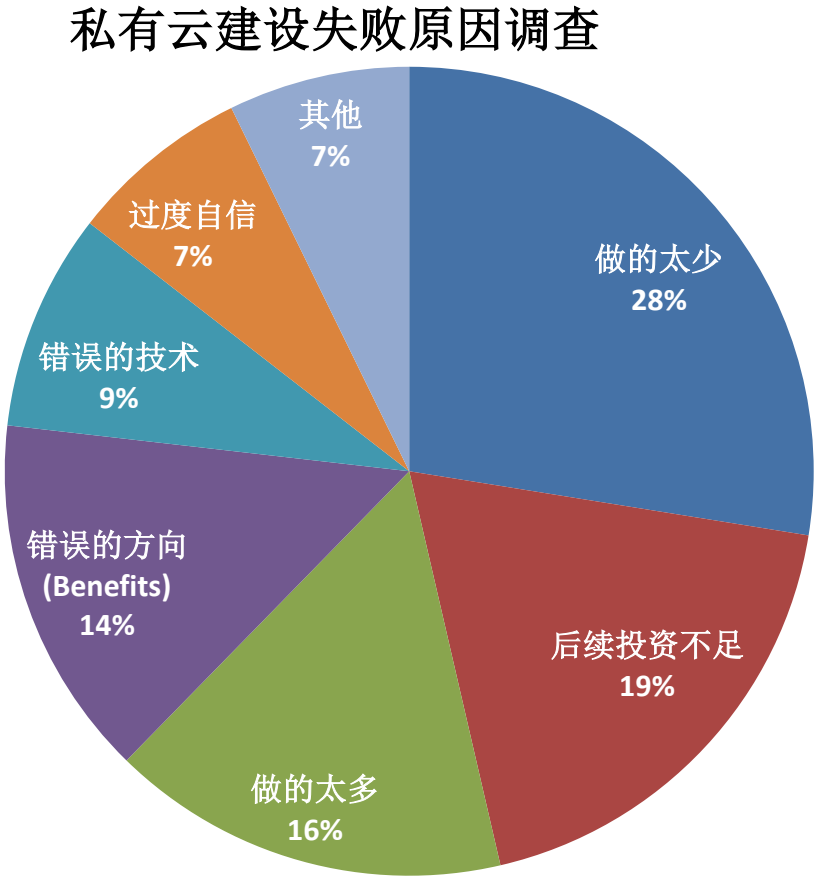
运营

- 收益、持续改进、版本本级、IaaS与PaaS对接

3 私有云建设失败的原因

私有云建设失败的原因

失败原因	比例	
做的太少	19%	不足
后续投资不足	13%	
做的太多	11%	过度
错误的方向(Benefits)	10%	
使用的错误的技术	6%	
过度自信	5%	
其他	5%	



云平台建设中的“五毒”

- **贪**：设定了过度的目标。**对外**，对于云计算的技术特定、发展规律、应用场景没有的**充分的理解**；**对内**，对于自身组织特点、人员技能、规章制度没有充分考量。造成目标的“too big”。

- **嗔**：当实施过程中，**达不到既定目标，心生怨恨**。不是反思、修正、改善；而是动摇、否定、迁怒，不断降低设施目标，造成云平台的“too little”。

- **痴**：对于技术过度迷信，没有把握技术的发展规律，没有在更宏观的层面去把控技术路线。例如openstack、SDN、分布式存储、kvm、docker这5个云平台高度依赖的技术，技术本身还存在明显的缺陷。作为云平台的建设者，需要把控技术风险、跟踪技术发展确实，**有所有“取”，有所“舍”**。

- **慢**：私有云的建设要和公司内部的战略、组织架构、IT工具等等结合，是一个高度个性化的项目。这样，对于同业的云平台的实施经验，容易出现以“傲慢”的态度对待，不能虚心学习。就像《傲慢与偏见》这本名著中所诠释的，傲慢将产生偏见，“**只看见自己想看见的**”，妨碍了进步。

- **疑**：以自身的云平台经验、技术路线为圆心，自以为是，画地为牢，怀疑、否定其他技术、**自我封闭**。对于不同建议想当然地下结论“**见取见**”认为只有自己的建设方法、见解是对的。

4

“自主掌控”到什么程度

自主掌控的监管要求

飞龙在天

深度参与某个开源项目，并回馈社区，比如华为对于openstack的贡献，在国内是第一的。这种情况，一般都有强烈的商业目的，有非常大的决心，非常大的投入。开发人员投入至少要**三百人以**上，而这个项目，要放到公司的**核心战略**中。例如2014年intel放弃了**自研的大数据**。

八部天龙

深度定制某个开源项目，作为自己的基础平台。比如BAT自研的PaaS，用以支撑庞大的IT体系。这种公司，做IT的人应该在千人以上，才有这么大的需求，才值得这么大的投入。

双龙聚首

深度定制某个开源项目，自己有大量的人员投入，同时有一个**实力雄厚**的厂商支持。例如三大运营商使用openstack做自己的公有云平台。这种项目，一般都用于该公司的一个**核心业务**上。

猛龙过江

将某个开源项目，作为**IT转型的契机**，大量的人员投入，构建IT基础平台。例如恒丰的IT的全面云化，招商和兴业的行业云项目。但风险也较高。

自主掌控


尽在掌握
技术方向
IaaS：openstack，kvm，ceph，Hadoop
PaaS：docker与cloudfoundry
风险把控
网络：网络风暴、广播风暴
高可用：脑裂、大面积宕机
数据：数据丢失、数据不一致
运维需求
高可用，自动化，灾备，权限控制，堡垒机对接，系统升级，
项目进度
范围、时间、进度、质量

5 大规模云的维护问题

基础设施的池化

按需采购

系统隔离

配置各异

扩容困难

云计算改变了基础架构的建设思路

云化采购：整体规划、按照服务器计算资源、存储配比，按池化的目标采购，减少扩容成本

池化部署：在保证安全的前提下，合并网络、简化网络、池化网络

标准配置：硬件设备标准化为5个套餐，云平台提供4个级别的套餐

动态调整：系统实现弹性伸缩、快速扩展，应对秒杀

云数据中心

绿色、开放、安全、弹性

- **云数据中心**
  1. **机房、机柜**、设备选型
  2. 计算资源池
  3. 存储池
  4. 网络
- **平台化运维**
  1. 集中管理，统一入口
  2. 统一的展现、监控、报表
  3. 变更自动化
  4. 与服务流程平台集成
- **团队建设**
  1. 开源产品的选型、运维
  2. 架构优化指导
- **持续运营**
  1. 同城灾备方案的制定和测试
  2. 初步建立同城灾备集群

服务器选型

云主机

1、2路高配置服务器

2、4路服务器：

联想R680： 4C8核256G

联想RQ940 ： 4C8核512G

外来考虑1T内存

3、超融合一体机： Nutanix，华为（X6800：4U4节点，无交换机），沃趣

- 刀片服务器 ： 华为E9000（12U、16个2路节点或8个4路节点：5000W）（传统刀片散热问题）

- 高密服务器

4、模块化机房/微模块

天蝎，电信和互联网一体化机柜

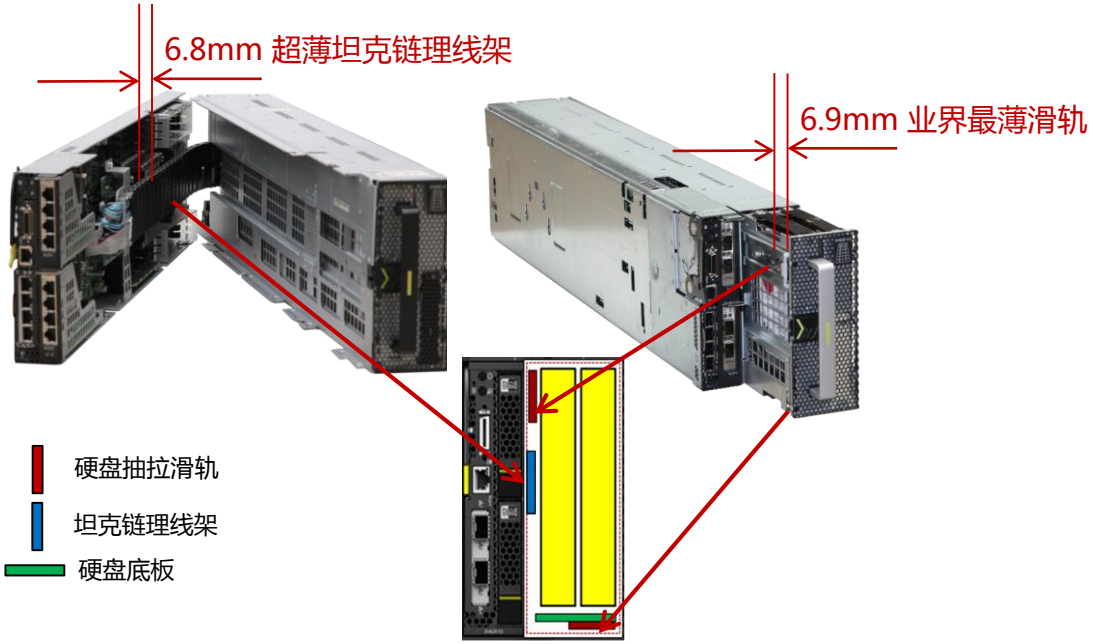
数据库

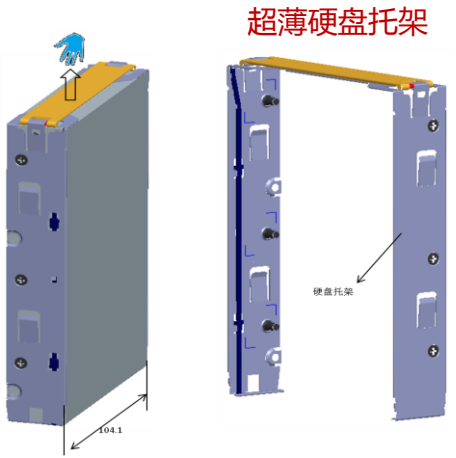
1、高配置4路服务器，4C10核心；

2、8路服务器

3、16、32路昆仑服务器

高密设计，比传统机架省空间50%





●超薄坦克链理线架与业界最薄滑轨，理线架与滑轨空间复用
●滑轨采用硬盘槽位单侧（左侧）与底侧布局，整框节省4个滑轨的空间

●超薄硬盘托架，比业界硬盘托架宽度减小3mm以上，硬盘密度比同类产品提升20%-50%

机房问题

云数据中心的机房设计问题

1、**机柜规划**：虚拟化的服务器集中放置。

2、**机房设计**：高密机房、微模块、整体机房。

3、信息点需求：每台设备至少6个信息点，需要大量的接入设备。

3、**万兆网**：虚拟化的密度会增加，例如采用万兆接入。

4、**托架**：每台虚拟化服务器，至少4块硬盘，存量服务器有8块硬盘，采用目前的隐形托架方式，承重是一个问题。

5、**供电**：每台虚拟化服务器，4C8核，256-512G内存，CPU使用率超过30%，耗电会比普通4路服务器大很多。

6、**散热**：虚拟化服务器集中放置，冷通道、热通道设计好。

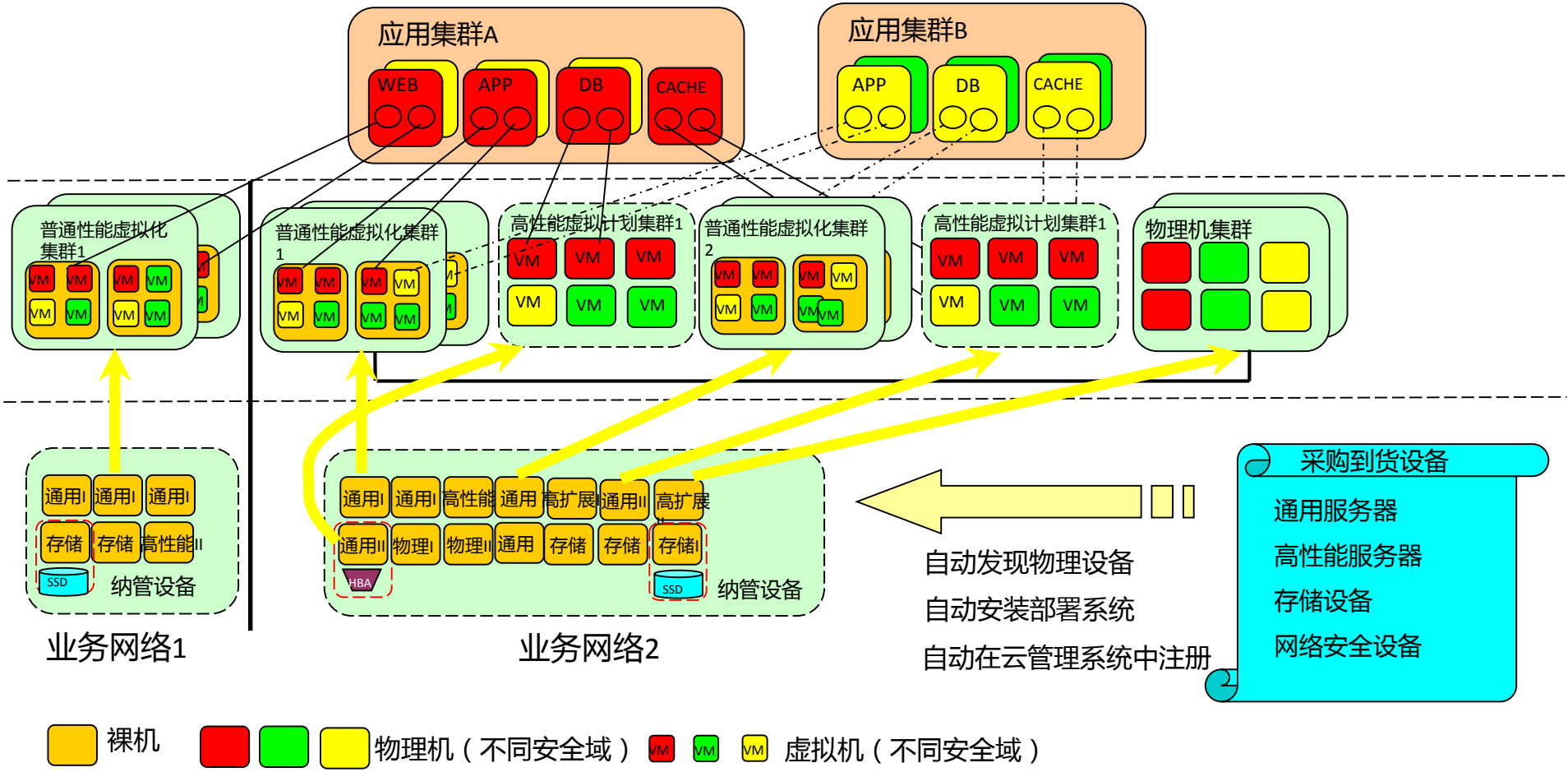
7、**互联网公司的经验**：例如大规模虚拟化定制的服务器，有的互联网公司有无机箱设计。

虚拟化管理

云数据中心的虚机管理问题

1. **过度创建**：范围蔓延，需要严格控制需求。
2. **过度依赖**：对于P2V，克隆技术的过滤依赖。
    1. P2V，用于从实体机迁移到虚拟机。但由于调用底层操作系统接口，失败率较高。同时迁移速度不可控。从最带1M/s，到自最高50M/s无法控制。
    2. 克隆，用于系统扩容、灾备建设时新建虚拟机。但超过300G的虚拟机，存在快照无法删除等风险。
3. **高可用**：vmware实现了集群级别的高可用，但操作系统还是单点。需要有完毕的备份机制和大量的带库投入。
4. **资源消耗**：1个512G的主机， 需要5T存储(目前有400T的缺口)。每台主机需要6-10个信息点，服务器集中放置后，需要大量的接入层交换机。

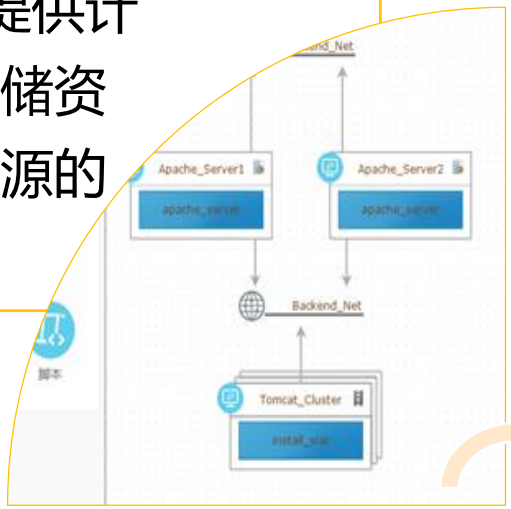
服务SLA



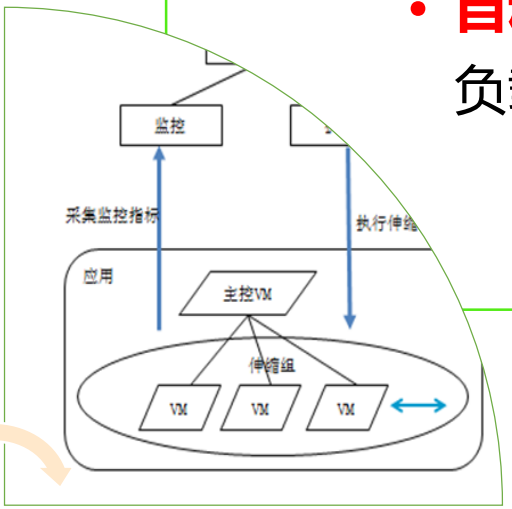
提供普通资源池、高性能资源池、物理资源池

自动化

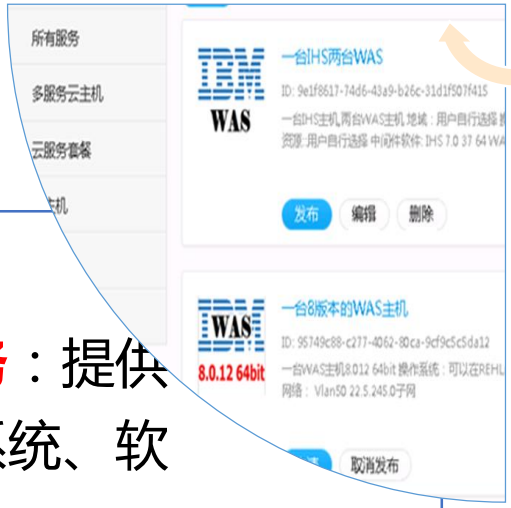
- **服务编排** 提供计算资源、存储资源、网络资源的组合



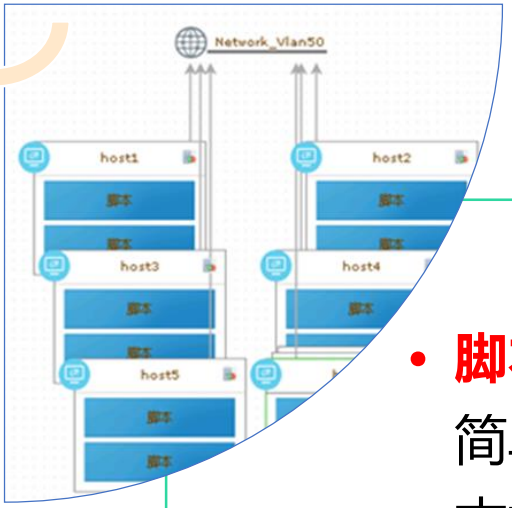
- **自动伸缩**：根据负载自动调整



- **丰富的云服务**：提供**50+种**操作系统、软件的组合安装



- **脚本接口**：提供简单、高效的脚本接口



自动部署

图形化的环境部署

6

私有云建设的技术风险有哪些？

边界问题

功能边界

1、系统纳管

① 纳管VMware、开源虚拟机

② 纳管存储

2、系统管理

① 哪些功能在云平台实施，哪些在vcenter实施

3、操作风险

① 哪些操作会造成系统影响

数据边界

1、数据导入

① 导入哪些数据，哪些数据只读，哪些数据双向修改

2、数据维护

① 哪些操作会导致数据不一致

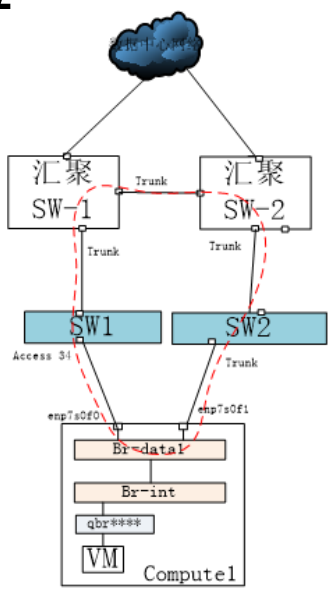
3、数据风险

① 如何保障数据安全

② 数据备份问题

网络风险

网络风暴



分析：

1、设备问题：网络设备，尤其是新出网络设备增加了大量新功能，要对应虚拟化、大二层等大量新情况，防环技术不完善。

2、边界问题：网络设备不能识别服务器内部的虚拟交换机，造成STP不能识别和处理虚拟交换机造成的环路

3、维护问题：虚拟化和刀片服务器使用了虚拟交换机，但做系统的不懂网络，网络的人不懂系统，造成虚拟交换机维护的困境。

解决和预防

一、小机层面

1. 交换机下连接虚拟交换机的接口启用trunk allow VLAN，仅允许必要VLAN经过
2. 交换机打开生成树协议，不开启portfast
3. 交换机上启动storm control功能，限制广播流。

二、KVM层面：要求厂商屏蔽通过修改配置文件的方法配置虚拟机交换机。

三、网络层面：

1、思科设备的脚本监控功能，预先防环

2、交换机虚拟化环境的智能防环技术

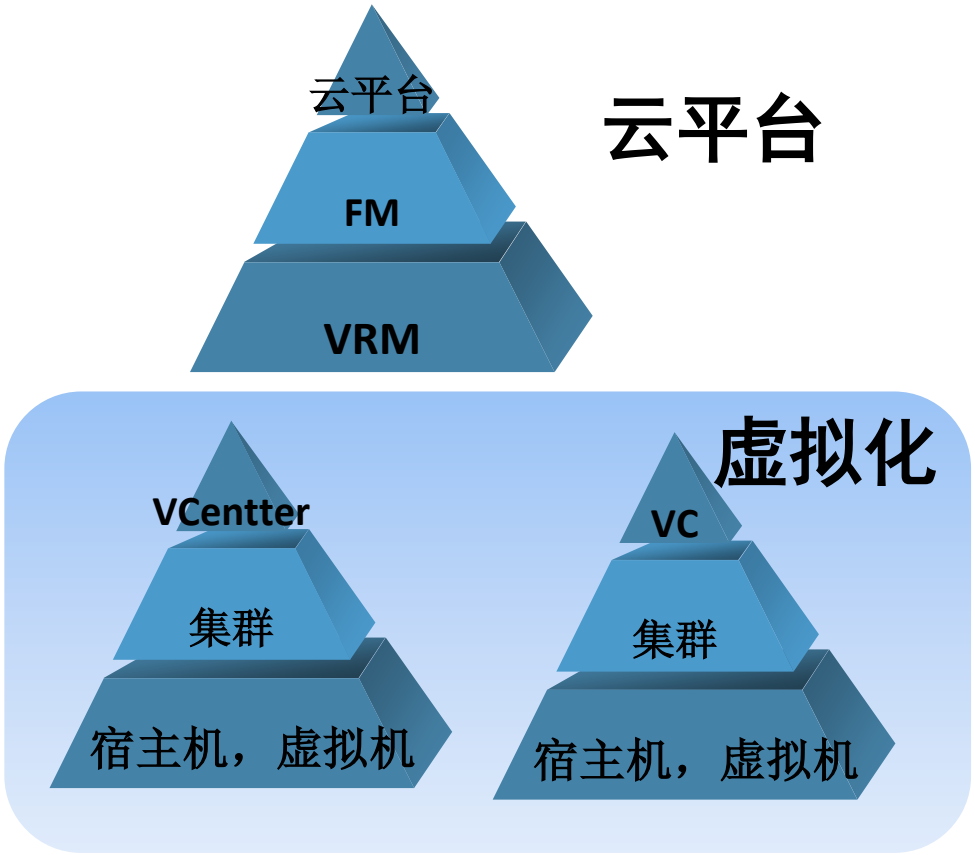
四、管理层面：

1、除了总行值班外，机房也安排网络部人员值班，预防网络中断问题。

2、某行的vmware虚拟机是网络部负责。

数据风险

数据丢失



- **管理问题**
  - 管理复杂，尤其是删除操作

  不能实现完全的自动化
  - 虚拟化与云平台的界面问题不清晰

- **数据问题**
  - 数据双向同步
  - 数据关联大，对于删除等操作，可能出现数据不一致

7

IS or NOT，银行私有云是否向开发中心开放自服务

向开发开放“自助服务”- 理想状态

用户自助

服务申请

服务变更

服务维护

服务释放

服务审批

运维管理

用户自助完成

- 查询VDC的各配额指标的使用率、总量、已用量、未用量。
- VDC按类型统计已申请资源数量。
- 申请VDC配额扩容。
- VDC用户管理。
- VDC服务目录管理。
- VDC网络环境配置。
- 审批业务用户的服务申请。

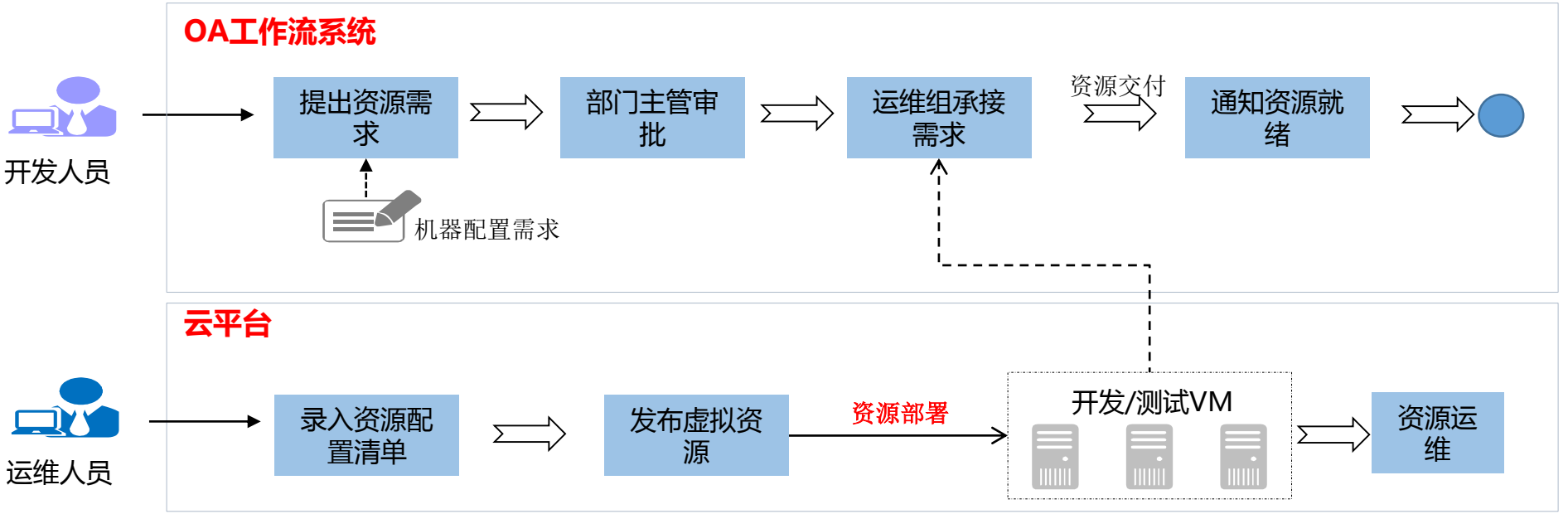
实现了流程再造

- 数据中心将系统服务通过“云服务”的方式发布出去。
- 开发中心、测试中心可以通过服务目录自助完成资源申请。

风险

- 没有公有云的“付费”约束
- 开放倾向于要“最大”的资源

变通方案1 - 线下

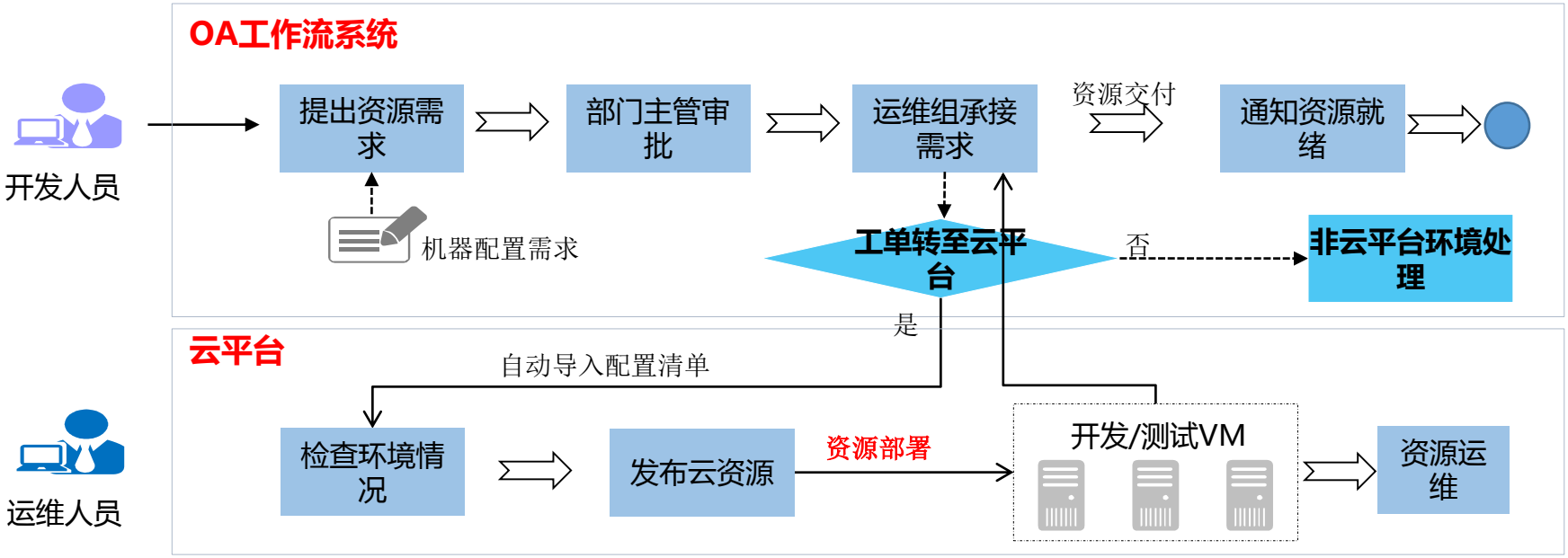


【现状】开发人员通过ＯＡ系统提出ＶＭ需求，主管审批过后，由运维组登录云平台线下准备资源。

【问题】OA工作流没有和云平台进行联动，资源需求单无法自动传递至云平台，资源部署需要线下准备，无法发挥工作流的自动化能力。资源利用率低，无用资源无法有效回收，无法有效统计各个开发处的资源使用情况。

变通方案2 – 工单对接

第一种：结合传统的OA工作流，使用者提出申请，由运维组负责资源的发放和管理

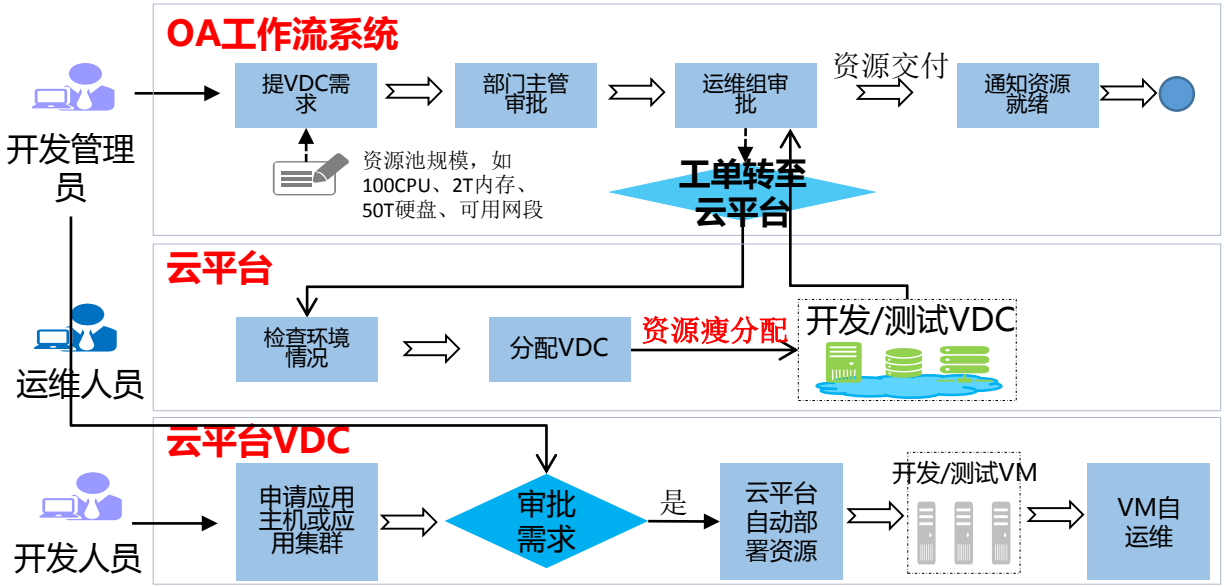


【优势】符合传统的资源申请和管理流程，对资源的使用更为安全受控，云平台自动化资源部署。

【劣势】没有充分发挥云平台价值，没使用多租户、自服务的功能，较多的运维工作。

变通方案3 – 分层管理

第二种：使用云多租户能力，给开发处分配资源池，自服务自动化部署资源。



【优势】各个开发处拥有自主可控的自服务云资源池VDC，一次申请无限使用，节省繁琐的审批流程，资源自动化部署，很少量的维护工作。资源瘦分配所需即所得，节省开发测试资源。按照资源使用量进行计量，更有效的管理部门KPI。

【劣势】开发处室在VDC内自服务、自运维，要求掌握简单的云平台使用能力。

- **申请资源池**：开发部管理员申请资源池；由运维部审批申请，并交付可用资源池
- **自助服务**：开发人员提交资源申请，由开发管理员审批并在自有资源池内进行分配部署。实现自有资源池的自服务自运维

8 数据库能否上云

服务器选型的标准化

U2VL

从传统的小机向云平台转型

MIS系统

小型机 → 应用-》linux → 应用-》虚拟机 → 数据库-》linux → 数据库->虚拟机 → 数据库-》分布式

交易系统

小型机 → 应用-》linux → 应用 》VM → 数据库->Linux → 数据库->虚拟机

核心系统

AS400 → 8路、16路X86服务器 + linux

解决关键问题

1. **Unix->linux：** 应用的兼容性问题，X86服务器、Linux的稳定性问题
2. **实体机->虚拟化：** 虚拟机的稳定性、高可用问题
3. **数据库->虚拟化：** 宕机后的数据一致性问题，IO性能问题
4. **分布式数据库：** 大规模应用问题，数据一致性问题

虚拟机上运行数据库

- 使用NIC teaming 实现可用性和负载均衡
- 使用VMXNET3 准虚拟化网络适配器提高性能，
- 磁盘规划 ： 数据库文件=》Datastore=》LUN的1:1:1配
- 虚拟机上运行数据库，Redhat系统建议调整IO的调度算法为noop（在Redhat系统仅做小IO整合，无其他的优化处理，IO的优化处理交由EXSI主机/hypervisor进行。）

9

云的最后一公里，分布式数据库

分布式数据库设计目标

基于主流的开源关系型数据库，实现数据库技术自主掌控

支持节点级容错，任何节点故障不影响业务运行，数据多副本，完善的数据备份恢复机制，支持双活数据中心

性能和容量可以在不中断业务情况下在线增加机器扩展

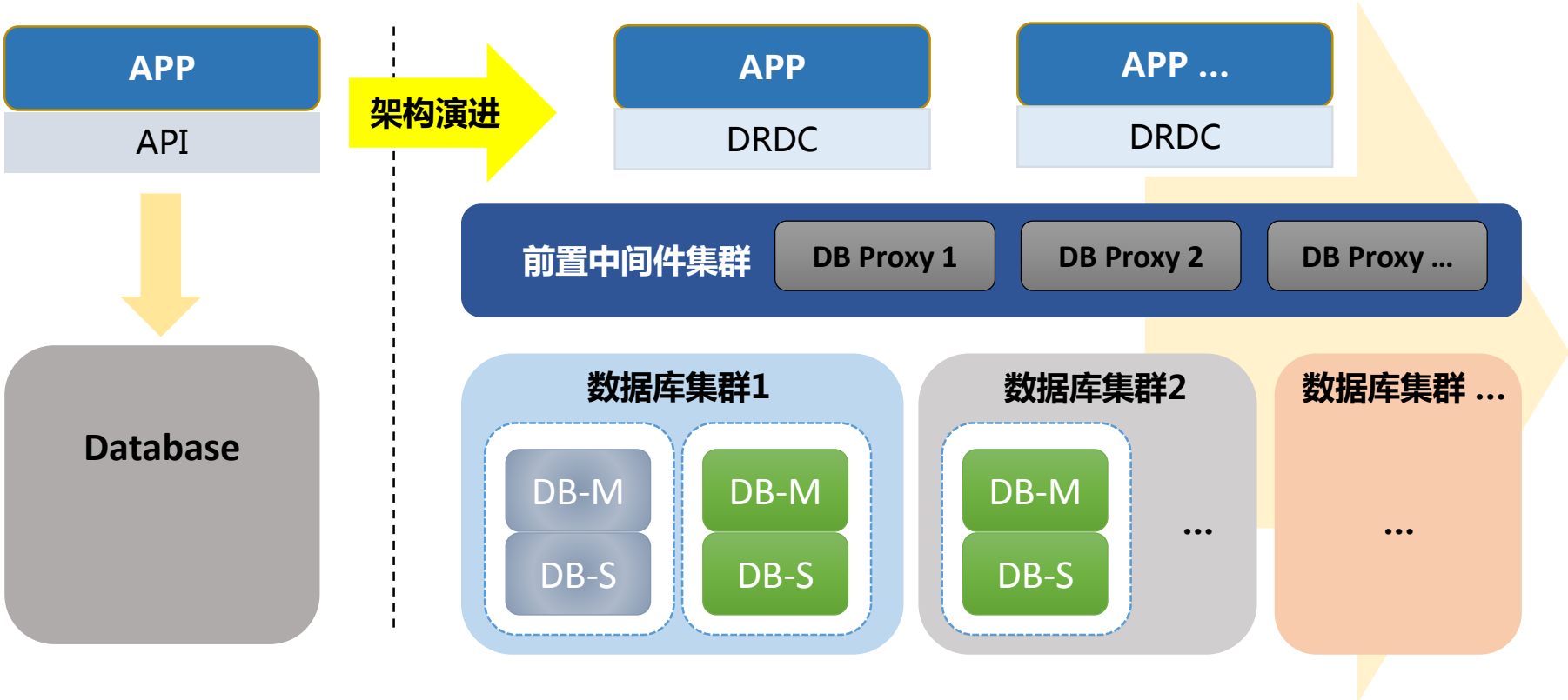
分布式数据库

支持分布式事务，采用分布式柔性事务机制，事务失败有完整回滚补偿机制

完善的性能及告警管理，实时监控慢SQL，完善的操作维护工具

兼容常用SQL99语法，针对分布式特性进行扩展

关键技术 - 线性扩展



客户端接入层、中间件层，DB层均可横向线性扩展，满足性能及容量的各种处理需求

- 中间件层根据业务需要可灵活配置中间件节点数量，实现处理性能线性扩展
- 数据在DB层根据策略自动重分布，存储容量随DB节点数线性扩展

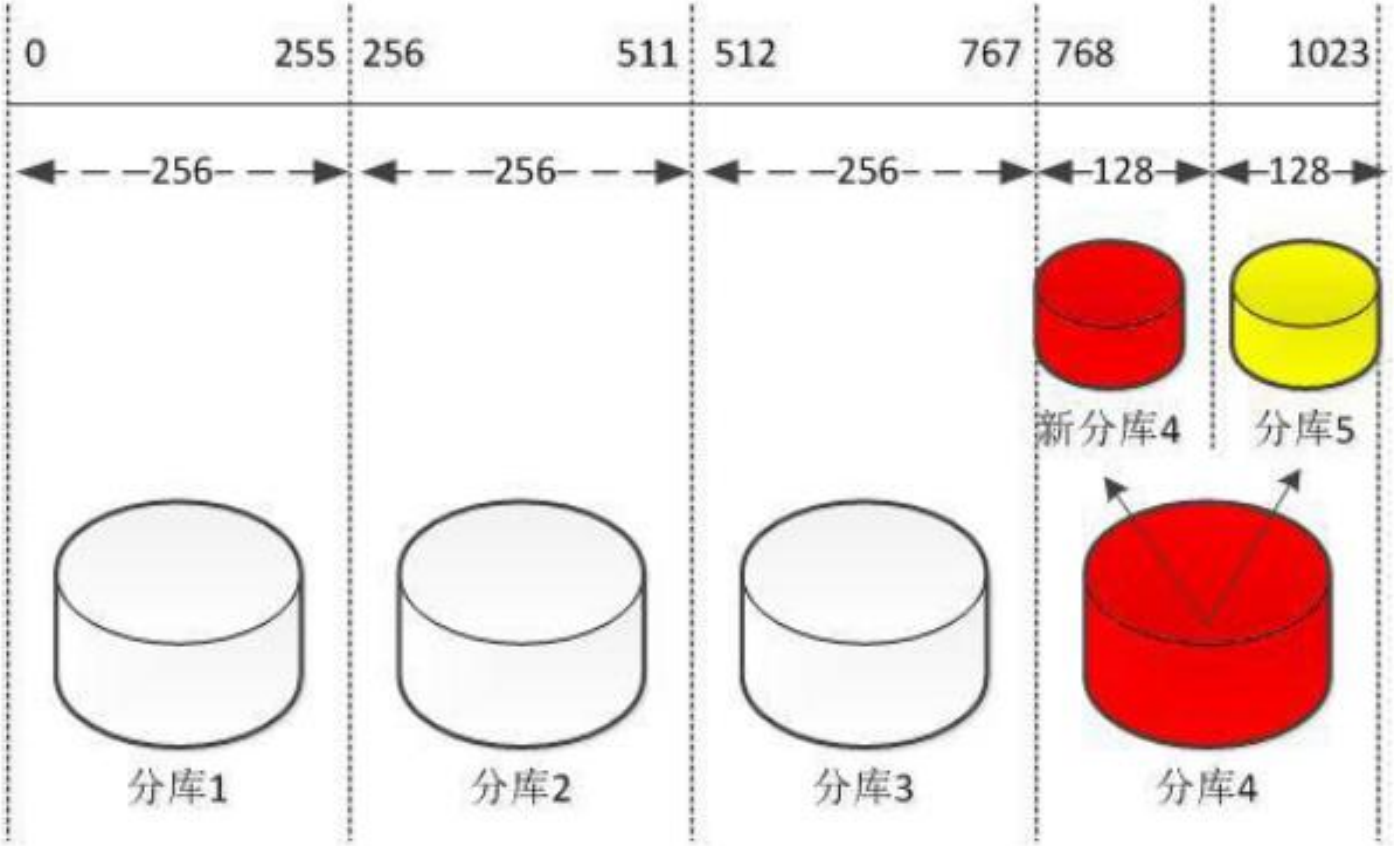
关键技术 - 线性扩展（续）

场景	Select	Insert	Update	Delete
单节点	27980	2238	2640	2422
2分片	24157	4531	4919	4931
3分片	24584	6825	6759	6626

- Select操作在高并发情况下，网络成为瓶颈，增加1个proxy之后，Select操作TPS最高达5W
- 典型写操作基本实现性能随group数量线性扩展
- 并发数增加以后，TPS可以保持在一个稳定的最大值的态势。

关键技术 - 数据重分布

在线数据重分布技术可支撑分布式数据库平台动态伸缩，弹性调度



- 数据迁移过程不停服务，服务暂停分钟级
- 支持部分重分布和完全重分布
- 对Range、List和Hash三种分片方式都支持

关键技术 - 智能管理台

自动化安装、可视化监控、故障自动修复为管理大规模数据库集群提供有效保障

- ✓ 智能化、可视化的运维，大幅提升易用性、可服务性体验
- ✓ 一键式安装，升级和巡检；在线扩容
- ✓ 集群内故障的自动检测，触发告警并进行自愈修复

10 PaaS，你准备好了吗？

容器出现的背景

服务互联网化

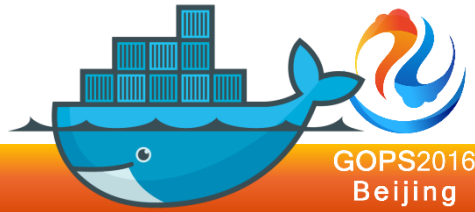
- **互联网公司**：凭借着强大技术平台，渗透能力，改变了零售业、物流业。而余额宝等产品，冲击着传统银行的业务领域。
- **银行**：感受到“金融科技公司”的竞争压力，纷纷进行“互联网+”改革
- **业务特点**：双十一的促销，抢购、秒杀成为新常态

系统虚拟化

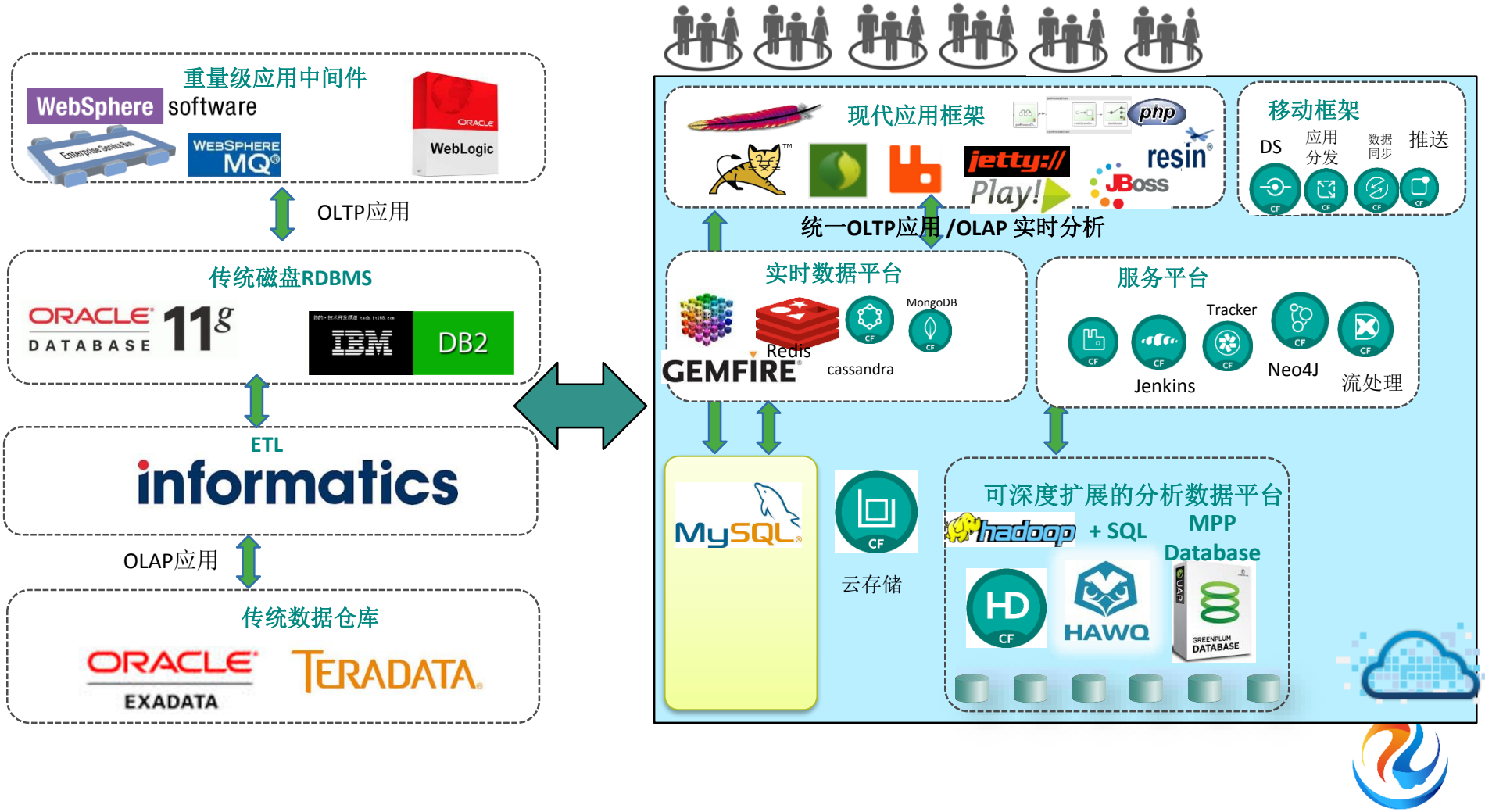
- **U2VL**：去IOE的监管要求，促进U2VL的技术转型，大量系统从Unix平台转化为Linux平台，进行使用虚拟化平台。
- **虚拟化**：凭借着隔离性、硬件无关性、高性能等优势，虚拟化技术在银行业广泛使用。
- **IaaS云平台**：各行分行搭建自己的IaaS平台，并进行大规模的虚拟化改造。

应用容器化

- **互联网公司**：google等互联网巨头通过容器技术来获得更高的资源利用率、更快的发布速度。
- **DevOps**：开发、运维一体化，持续集成、持续发布成为一大技术趋势。容器大大加速了这个过程。
- **PaaS平台**：发展一直缓慢的PaaS平台，凭借容器技术开始大规模应用。

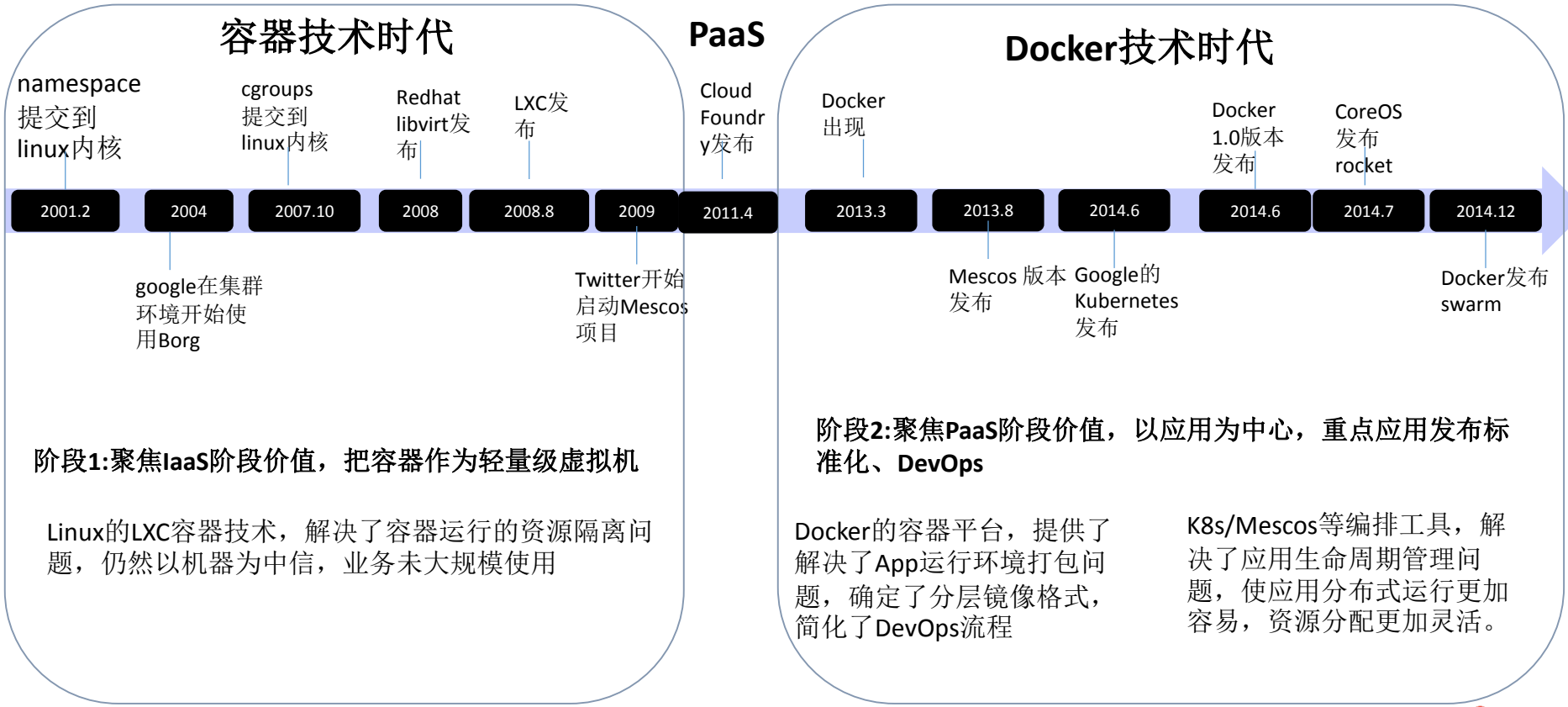


传统应用架构和互联网应用架构



容器的历史

容器技术从2001年开始出现，PaaS平台从2006年开始出现，而2013年的Docker出现才引爆了市场。



- Cloud Foundry，openshift传统的PaaS受到Docker的冲击

- 三大容器编排工具 K8s/Mescos /swarm相继形成
- Docker的竞争对手出现

系统运维角度，容器的收益

		虚拟化在我行状况	容器的特点	使用容器的收益
日常维护	占用磁盘空间	100-200G	100M-200M	解决了虚拟化存储瓶颈问题
	占用内容	固定值，8G-32G	小，为应用所占内存	解决了虚拟化内存瓶颈问题
	CPU占用率	低，由于磁盘与内存的瓶颈问题，一般在10-30%	高	提升了CPU使用率
	服务器利用率	一台宿主机5-10个	可以达到上百个	
	启动速度	分钟级，包括操作系统的启动时间	秒级，只有应用的启动时间	对于出现问题的容器，可以快速重启
动态	新系统上线	测试环境和生产环境隔离，新系统上线需要准备新的基础环境后，发布应用	可以实现测试环境和开发环境统一标准，将应用和依赖的包一起发布	提升上线速度
	横向扩展	需要部署虚拟机，通过中间件的集群技术进行扩容	通过容器的集群技术进行扩容	实现秒级扩容、在线扩容、自动扩容

使用容器

- 使用容器后，资源率利用率会大幅度提升，但运维难度加大

容器的问题 - 技术路线之争

容器之争： Docker阵营分裂

- Docker ： 发现Swarm后，与google的k8s和Mesco出现竞争关系，docker可能通过限制器API接口来封杀其他的编排工具，docker阵营分裂，
- LXC ： 原生的容器，支持cgroup,namespace
- Rocket： Docker最早的支持者CoreOS，创建了容器，得到了google大力支持
- LXD： Ubuntu 是支持Docker最好的linux，提出了LXD
- Mesos Containerizer： Mesos 提供

编排工具

- Mesos
- k8s： Google
- Swarm :Docer

容器标准：

- runC 与OCI： Linux基金会与Docker建立OCI（ Open Container Initialtive）组织，提出OCF（开放容器格式标准）
  - runC： Docker提出
- AppC： CoreOS 提出
- CNCF： Cloud Native Computing Foundation云原生计算基金会，google倡导

编排工具之争

编排工具：容器编排工具提供**调度和集群**的技术，提供用于基于容器应用可扩展性的基本机制。这些工具使用容器服务，并编排他们以决定容器之间如何进行交互。解决容器运行的**高可用问题，管理问题**。

Mesos
- 诞生于2009年，由Mescoshere主导
- 推出基于Mesos和Marathon的DC/OS
- 引入Universal Container摆脱Docker的依赖定位于数据中心操作系统
- 支持hadoop、spark、docker等多种应用框架
- Mesos仅提供资源抽象，由各种framework提供应用的管理
- marathon是管理docker的framework

类似于 **Xen**

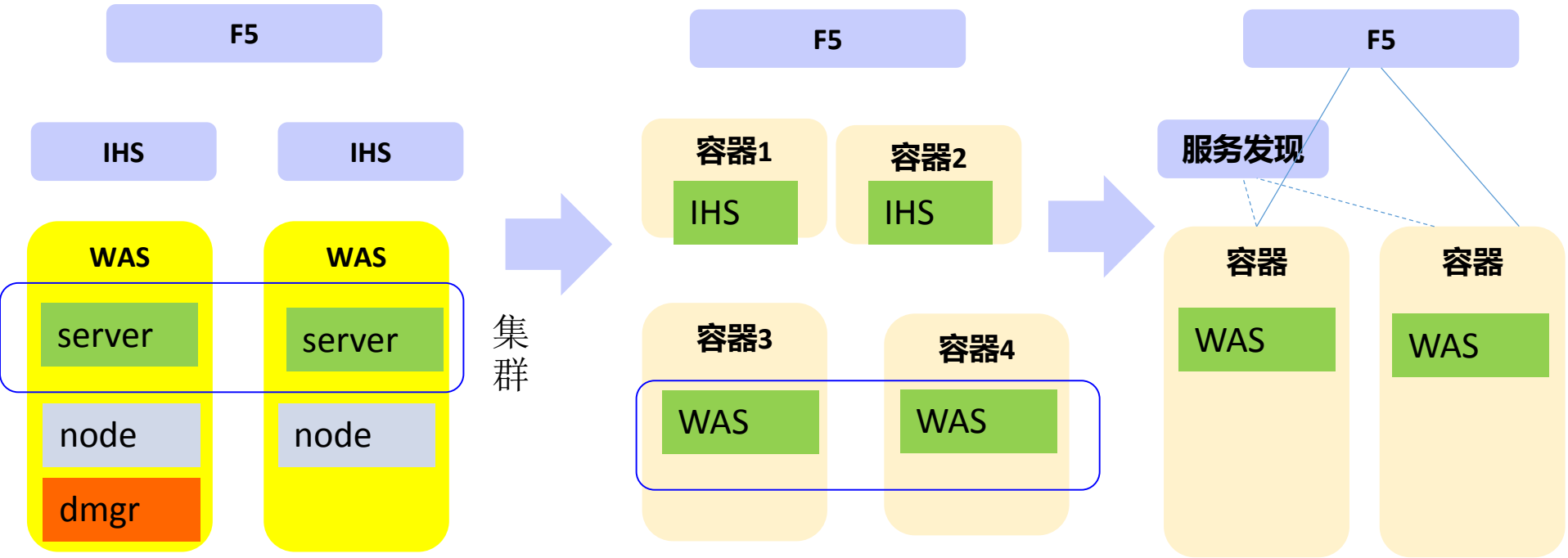
Kubernetes，简称K8s
- 源自google borg，诞生于2014年
- 原生为docker而设计
  社区最为活跃
- Google开始支持rocket已摆脱对docker的依赖

类似于 **KVM**

Swarm
- docker官方的产品
- docker推出了完成的产品线，支持公有云、私有云、混合云

类似于 **vmware**

容器化改造 – 中间件改造

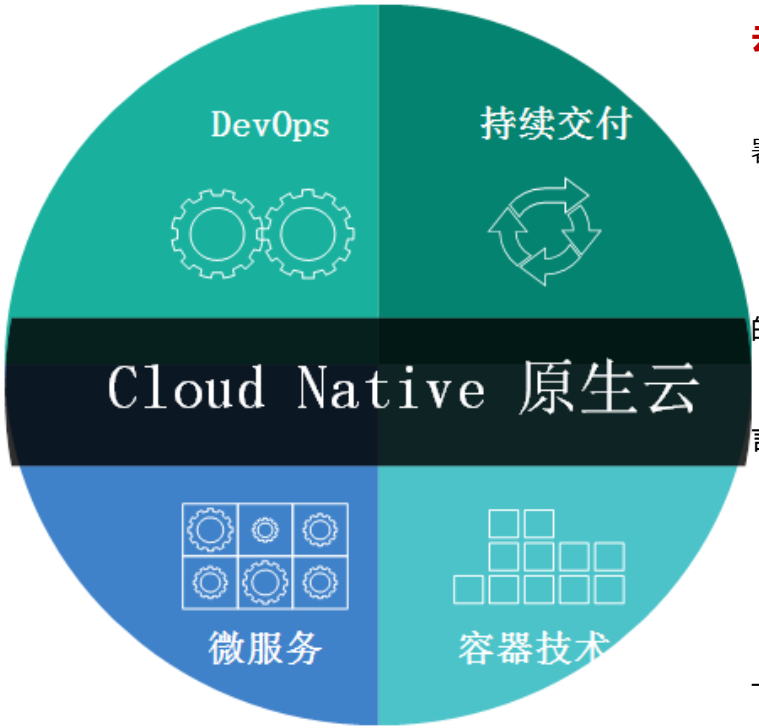


传统的中间件：依赖中间件自身的集群功能，扩容困难

1、中间件的简单封装，容器太“重”了，无法发挥容器的优势
2、与容器的无状态的管理模式“水土不服”

1、替换为容器模式
2、WAS替换成Tomcat，或Liberty
裁剪掉WAS的Ejb等高级功能

容器化改造 – 应用改造



1 分布式架构

2 应用无状态

3 应用数据分离

云应用平台Heroku创始人Adam提出12要素应用 宣言

1. 基准代码：一份基准代码，多份部署。基准代码和应用之间总是保持一一对应的关系。所有部署的基准代码相同，但每份部署可以使用其不同的版本。

2. 依赖：显式声明依赖关系。应用程序一定通过依赖清单 ，确切地声明所有依赖项。

3. 配置：在环境中存储配置。将应用的配置存储于环境变量中。环境变量可以非常方便地在不同的部署间做修改，却不动一行代码。

4. 后端服务：把后端服务当作附加资源。应用不会区别对待本地或第三方服务。对应用程序而言，两种都是附加资源。

5. 构建，发布，运行：严格区分构建，发布，运行这三个步骤。

6. 进程：以一个或多个无状态进程运行应用。应用的进程必须无状态且无共享。

7. 端口绑定：通过端口绑定提供服务。应用完全自我加载而不依赖于任何网络服务器就可以创建一个面向网络的服务。

8. 并发：通过进程模型进行扩展。开发人员可以运用这个模型去设计应用架构，将不同的工作分配给不同的进程类型。

9. 易销毁性：快速启动和优雅终止可最大化健壮性。应用的进程是可销毁的，意思是说它们可以瞬间开启或停止。

10. 开发环境与线上环境等价：尽可能保持开发、预发布、线上环境相同。应用想要做到持续部署就必须缩小本地与线上差异。

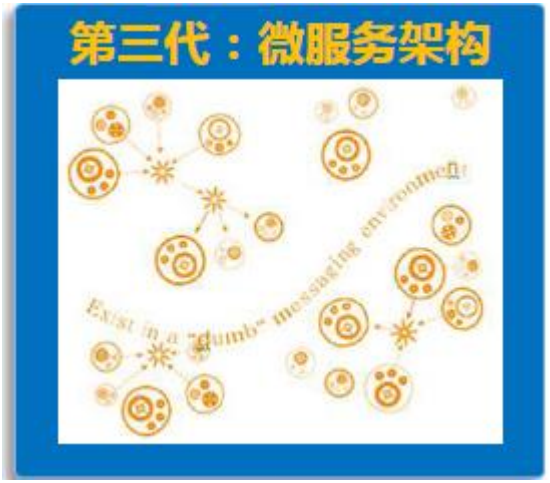
11. 日志：把日志当作事件流。应用本身考虑存储自己的输出流。不应该试图去写或者管理日志文件。

12. 管理进程：后台管理任务当作一次性进程运行。一次性管理进程应该和正常的常驻进程使用同样的环境。

容器化改造 – 微服务化

- 紧耦合，
- 系统复杂、错综交互，动一发而牵全身
- 重复制造各种轮子：OS、DB、Middleware
- 完全封闭的架构

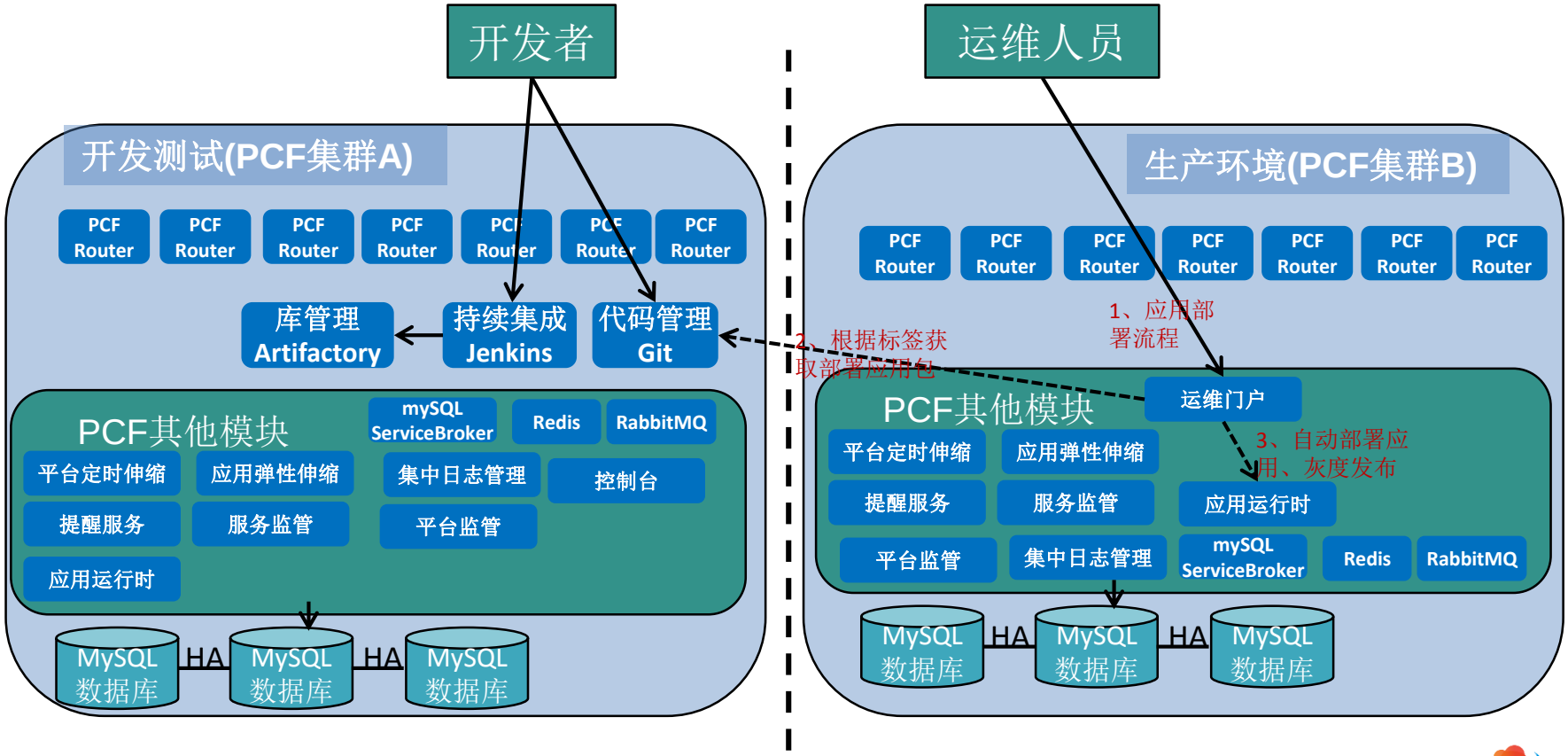
- 松耦合
- 在大型、超大型企业中仍然流行
- 通常通过ESB进行系统集成，有状态
- 大团队：100~200人
- TTM: 1年、半年、月
- 集中式、计划内停机扩容



- 解耦
- 互联网公司、中小企业、初创公司
- TTM: 按天、周进行升级发布
- DevOps: CI, CD, 全自动化
- 可扩展性：自动弹性伸缩
- 高可用：升级、扩容不中断业务

容器化改造 – 发布流程

PaaS开发测试到生产环境



结束语

- **格局**有多个大， 云平台就能做多大
  - 不但要做技术，更要做**技术管理**
- 六度：布施、**持戒**、忍辱、**精进**、禅定、**智慧**

周海鹏@中信云平台

北京 东城

推荐

- 伯凡日知录，冬吴相对论
- 罗辑思维
- 老梁看世界

DevOpsDays 即将首次登陆中国

DevOps 之父 Patrick Debois 与您相约

DevOpsDays 北京站 2017年3月18日

门票早鸟价仅限前100名，请从速哟

http://2017-beijing.devopsdayschina.org/

想第一时间看到
高效运维社区公众号
的好文章吗？

请打开高效运维社区公众号，点击右上角小人，如右侧所示设置就好

Thanks

高效运维社区

开放运维联盟

荣誉出品