

微博DCP

基于Docker的混合云架构与应用实践

@it_fuwen 微博平台研发中心

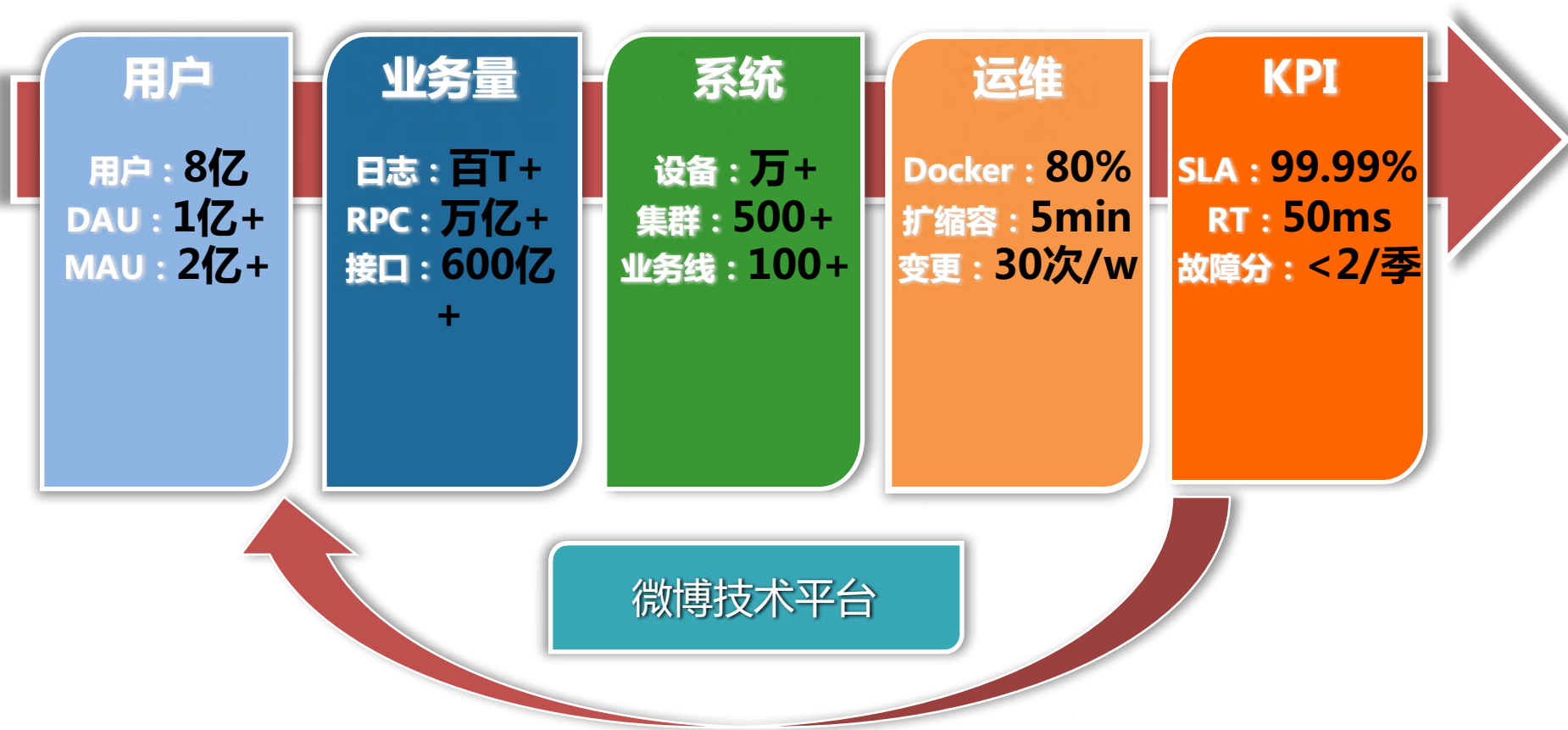
- 一、背景、挑战与实现
- 二、基础设施
- 三、弹性调度
- 四、Weibo DCP编排
- 五、春晚实战与总结



Part 1

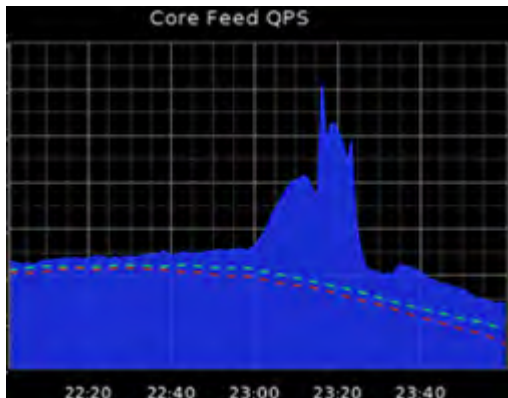
Weibo DCP

背景、挑战与实现



十亿级PV、千亿级数据、2000台以上的服务器规模、20个以上的大小服务模块、百亿级数据HBase存储、千台以上的Docker混合云集群，持续不断的技术挑战；

- 春晚峰值流量应对
 - 机架位不足，上千台服务器库存不足
 - 千万级采购成本巨大
 - 采购周期长，运行三个月只为一晚
- “宝宝”娱乐事件等热点突发峰值应对
 - 突发性强无预期、无准备
 - PUSH常规化，短时间大量设备扩容需求



如何10分钟内完成1000节点扩容能力？

服务扩缩容流程繁琐

项目评审
设备申请

入CMDB

装机
上架

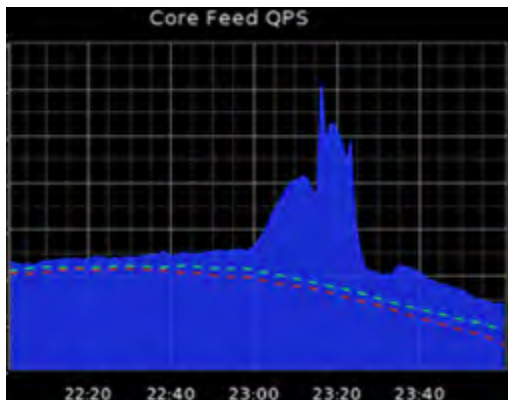
初始化

服务部署

报修
下架

微博业务现状与解决方案

弹性快速扩缩容



混合云弹性调度可伸缩业务成本节省数倍

场景	私有云方案	公有云方案	预估总成本对比 (私/公)
短期峰值应对	部分调度, 部分采购	按小时付费	数十倍
日常常规部署	按月摊销	包月付费	1.0~1.2
离线计算	按月摊销	按小时付费	数十倍

配置	按量 (元/小时)	包月 (元/月)
4核4G, 20G磁盘	1.13	302
4核8G, 20G磁盘	1.77	402
4核16G, 20G磁盘	3.05	602
8核8G, 20G磁盘	2.25	598
8核32G, 20G磁盘	6.09	1198
16核16G, 磁盘20G	4.52	1207
16核64G, 磁盘20G	12.17	2390
备注	小时计费按月为每月3~4倍 (84G必须选择16核)	

极端峰值

成本

扩展性

业务快速迭代

产品更新迭代快, 系统变更代码指数增长

扩！扩！扩！

平滑 快速 无缝 高效？？？

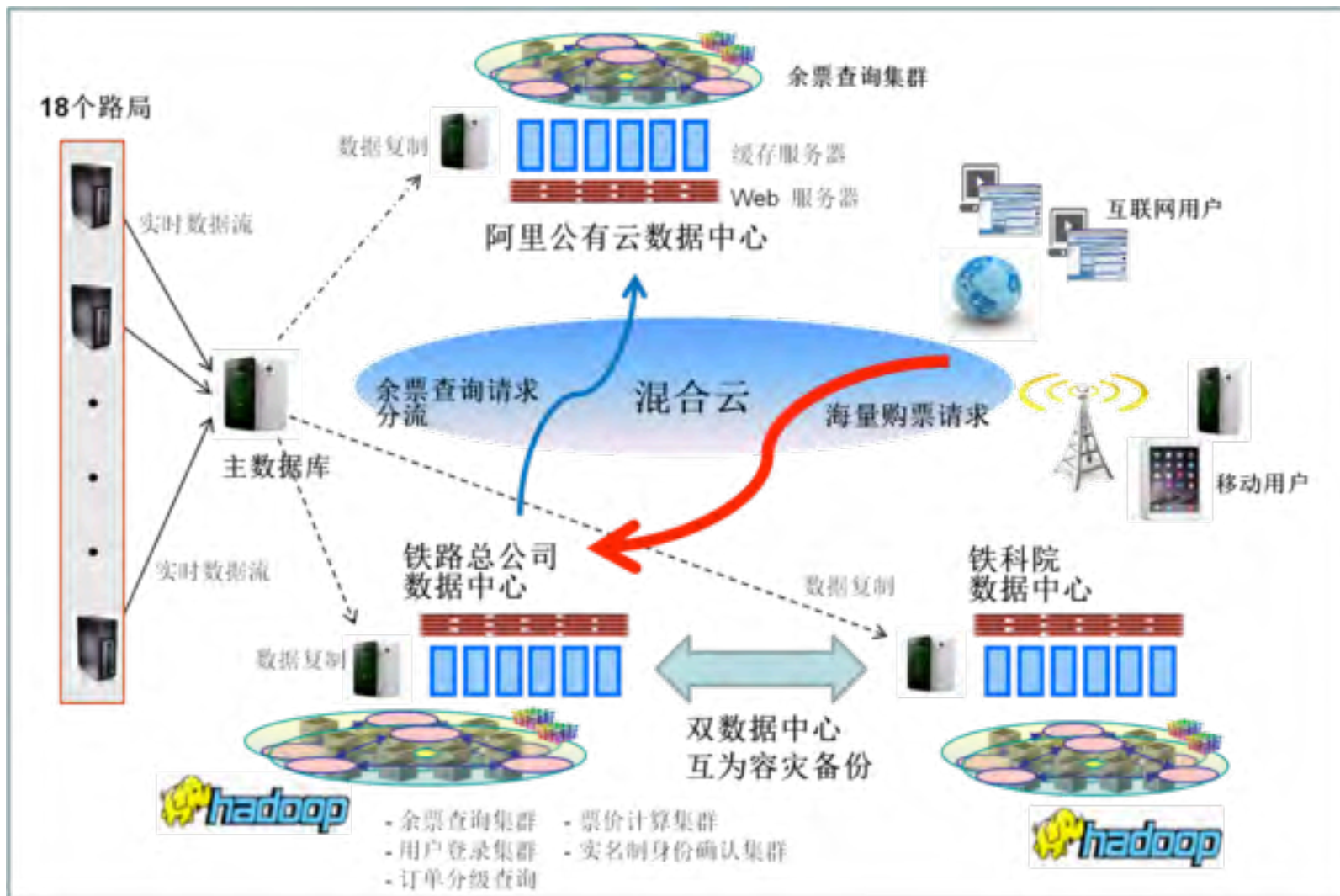


混合云趋势：安全、可扩展性、成本...

- 阿里云、AWS等公有云平台趋于成熟
 - 国外Zynga、Airbnb、Yelp等使用AWS进行部署
 - 国内阿里云12306、高德、快的已部署，陌陌等部署中
 - 12306借助阿里云解决饱受诟病的春节余票查询峰值问题
- Docker、Mesos等容器新技术使大规模动态调度成为可能
 - 京东618大促借助Docker为基础的弹性云解决峰值流量问题



12306混合云案例



12306 两地三中心 混合云架构

● 混合云进展:

➢ 容器数: 3000+

● 春晚实战:

➢ 10分钟混合云扩容1000节点技术能力

➢ 春晚峰值历史新高，两天内完成1375台阿里云ECS扩容，实现无降级平滑过渡，高峰支持微博50%主体流量。Feed流、红包飞、手机微博不同业务方均完成支持

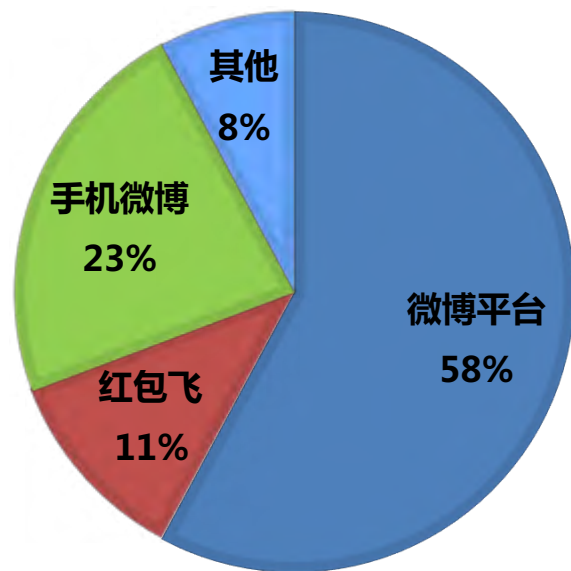
● 晚高峰及多次峰值流量应对:

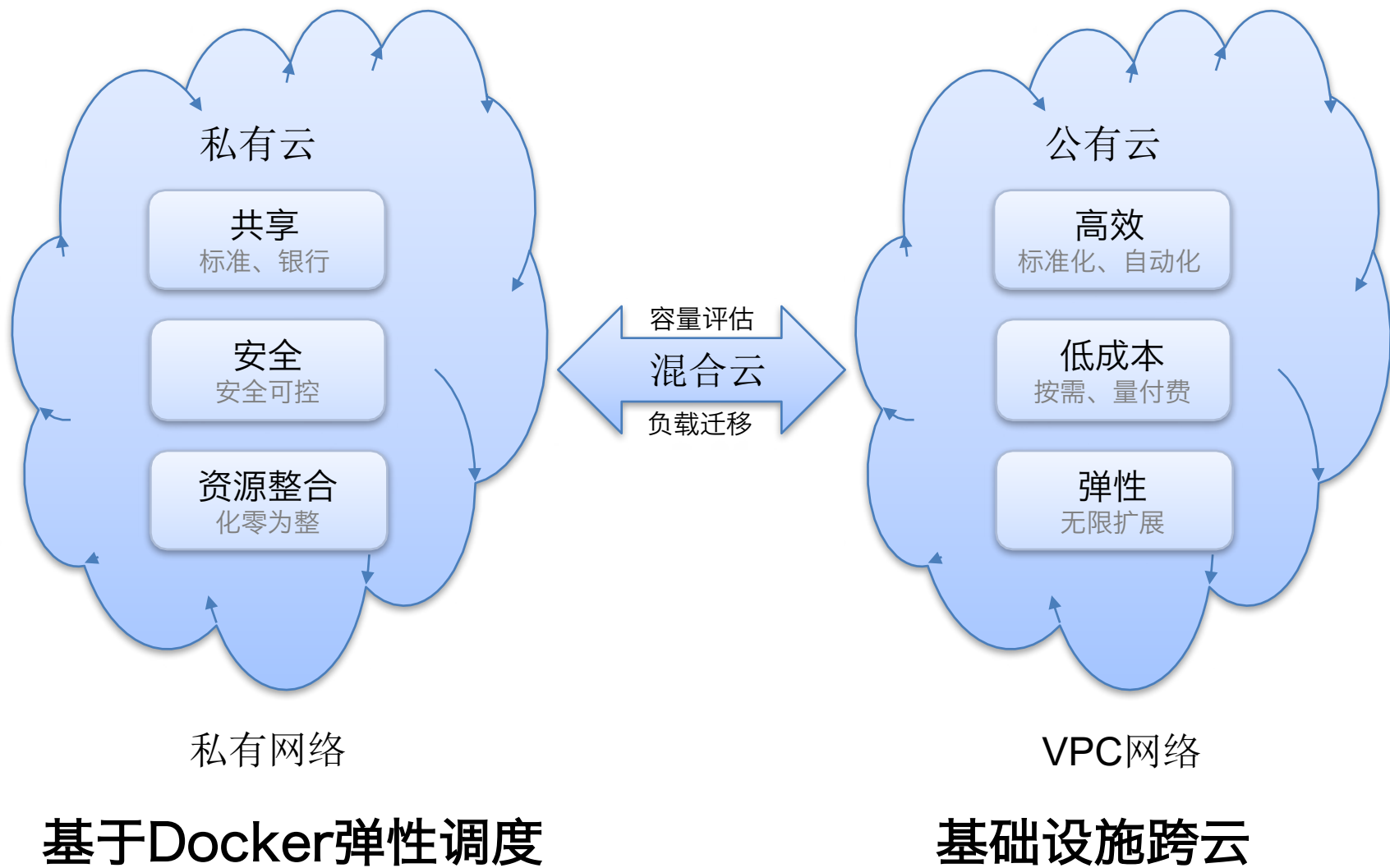
➢ 晚高峰弹性扩容500+节点

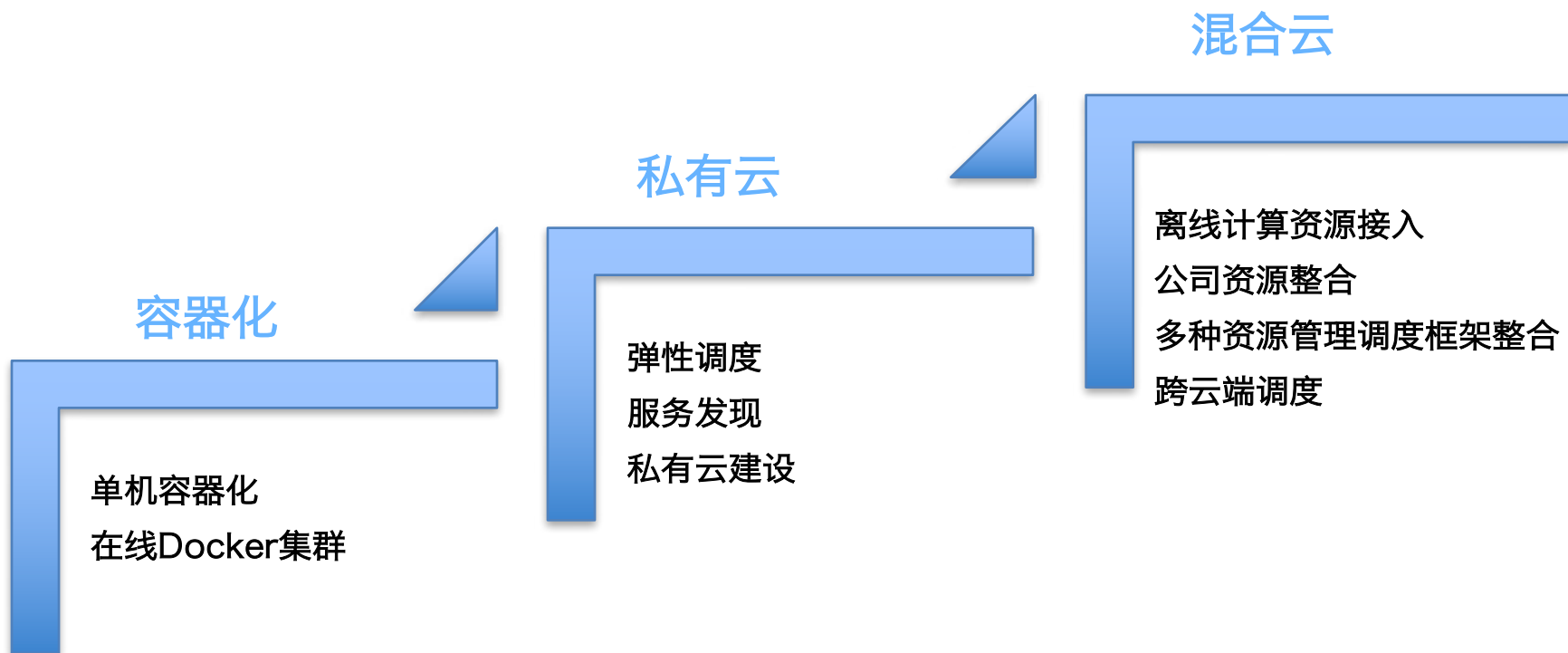
➢ “宝宝”事件弹性扩容

主要业务方

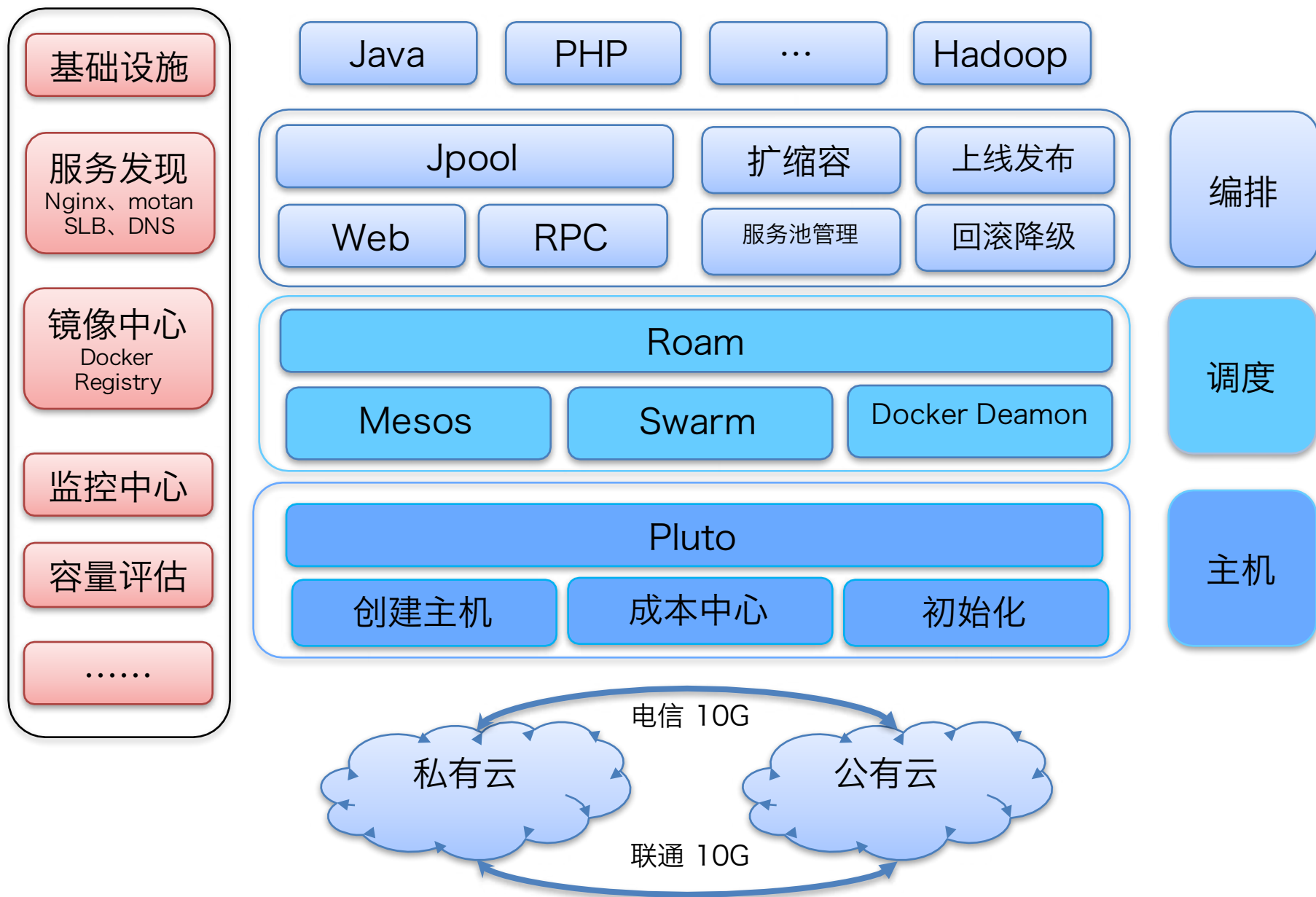
■ 微博平台 ■ 红包飞 ■ 手机微博 ■ 其他

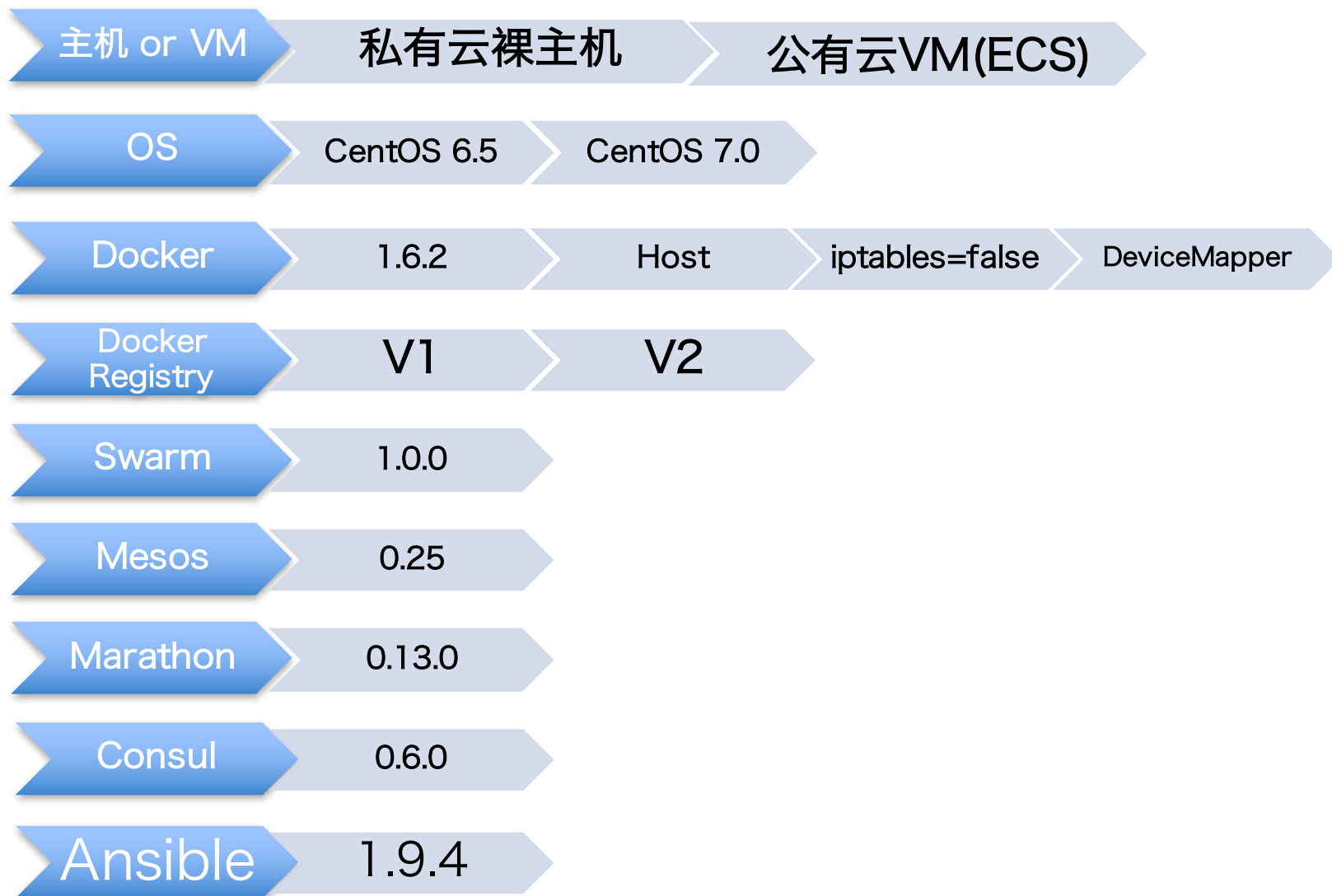






混合云DCP技术架构

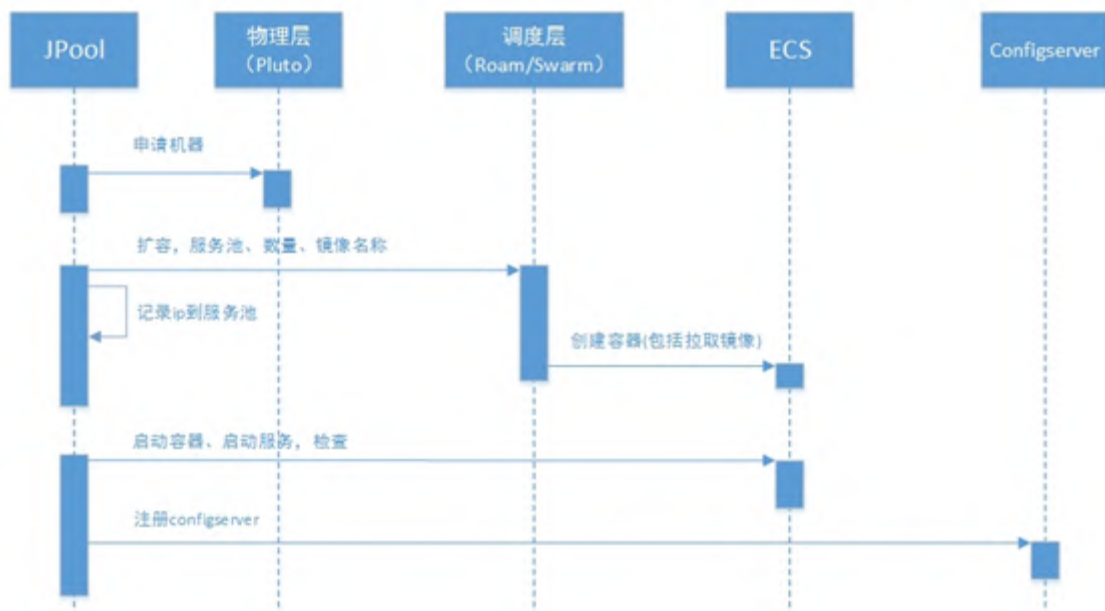
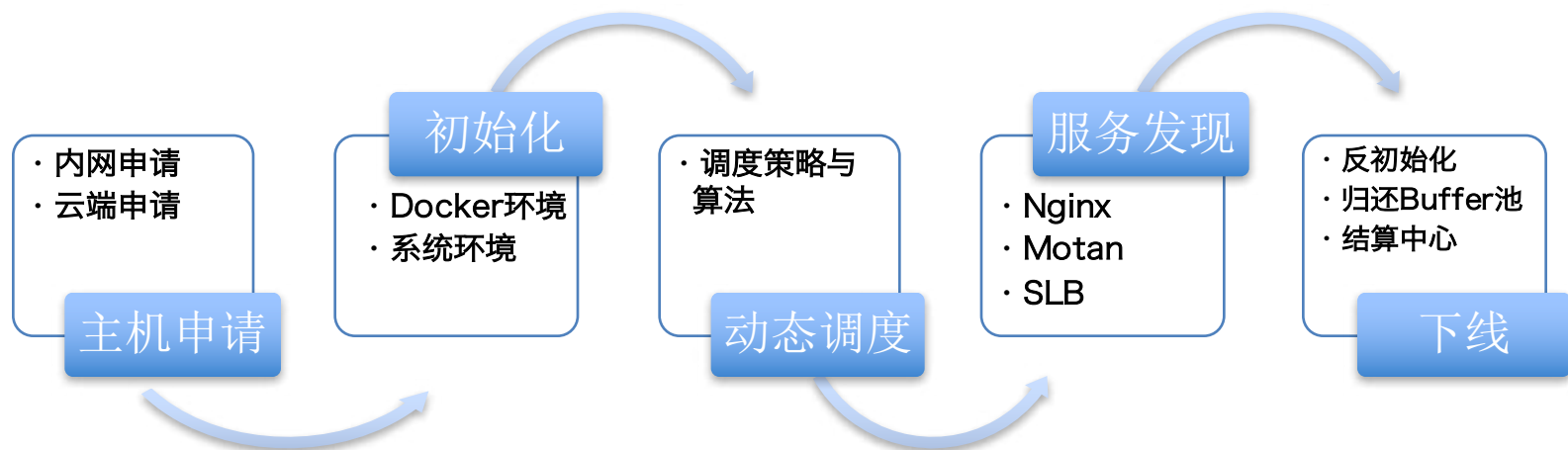




混合云DCP功能模块

业务方	红包飞	MAPI	广告	有信	Feed
	用户	通讯	平台架构		
P A A S	Swarm	Docker 调度策略	Mesos 调度管理	容量评估	Docker Register
	容器监控	调度监控			Docker 镜像市场
I A A S	共享池 管理	成本核算	ECS管理	SLB 管理	Consul 工具管理
	四七层解 决方案	专线保障	公有云 流量管理	审批流程	OS升级 自动化
基础 框 架	软件安装	工程框架	安全保障	账户体系	Docker 工具体系
	监控体系	DNS管 理	配置管理	阿里云 Yum/日志	

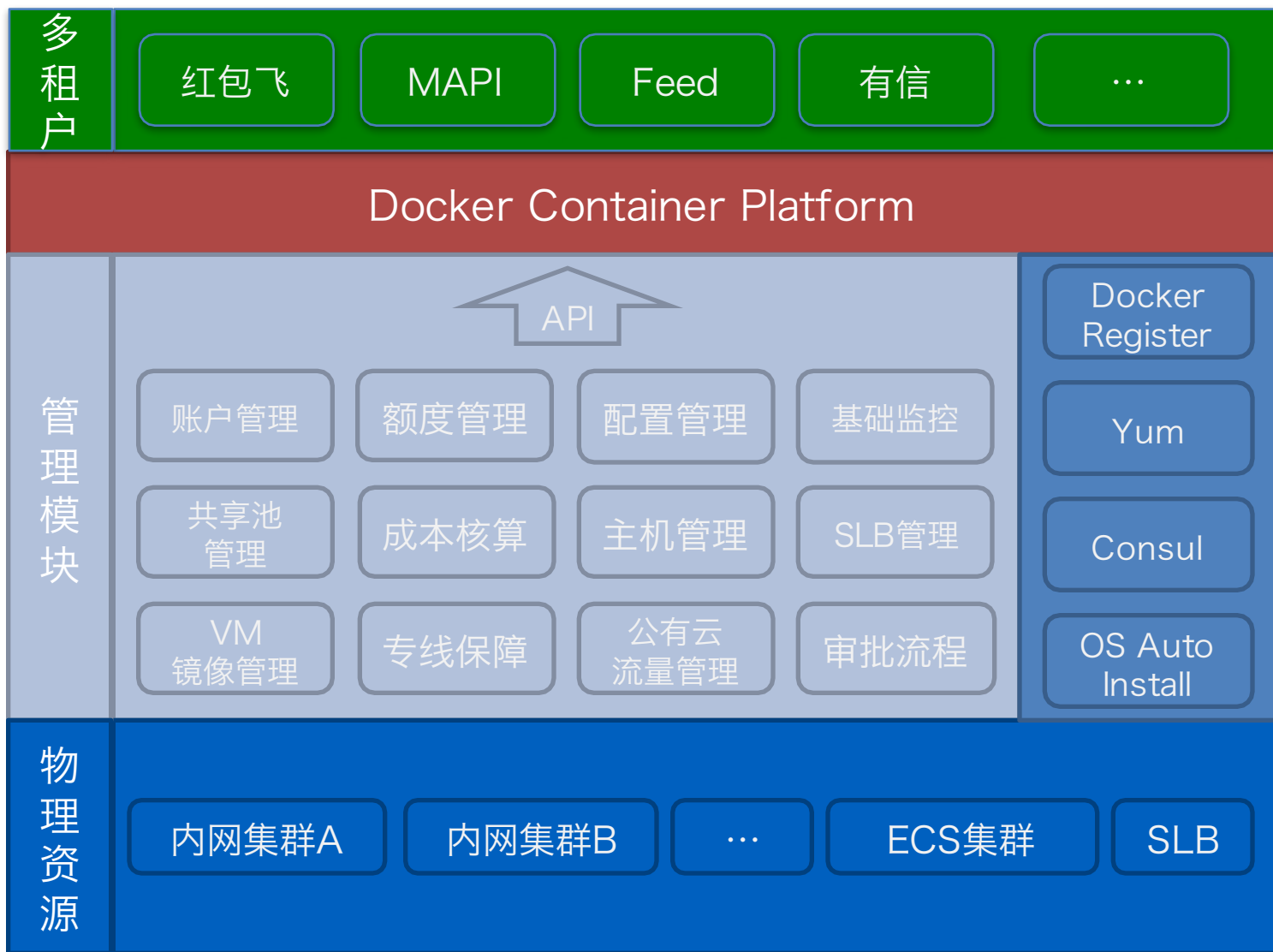
混合云DCP流程



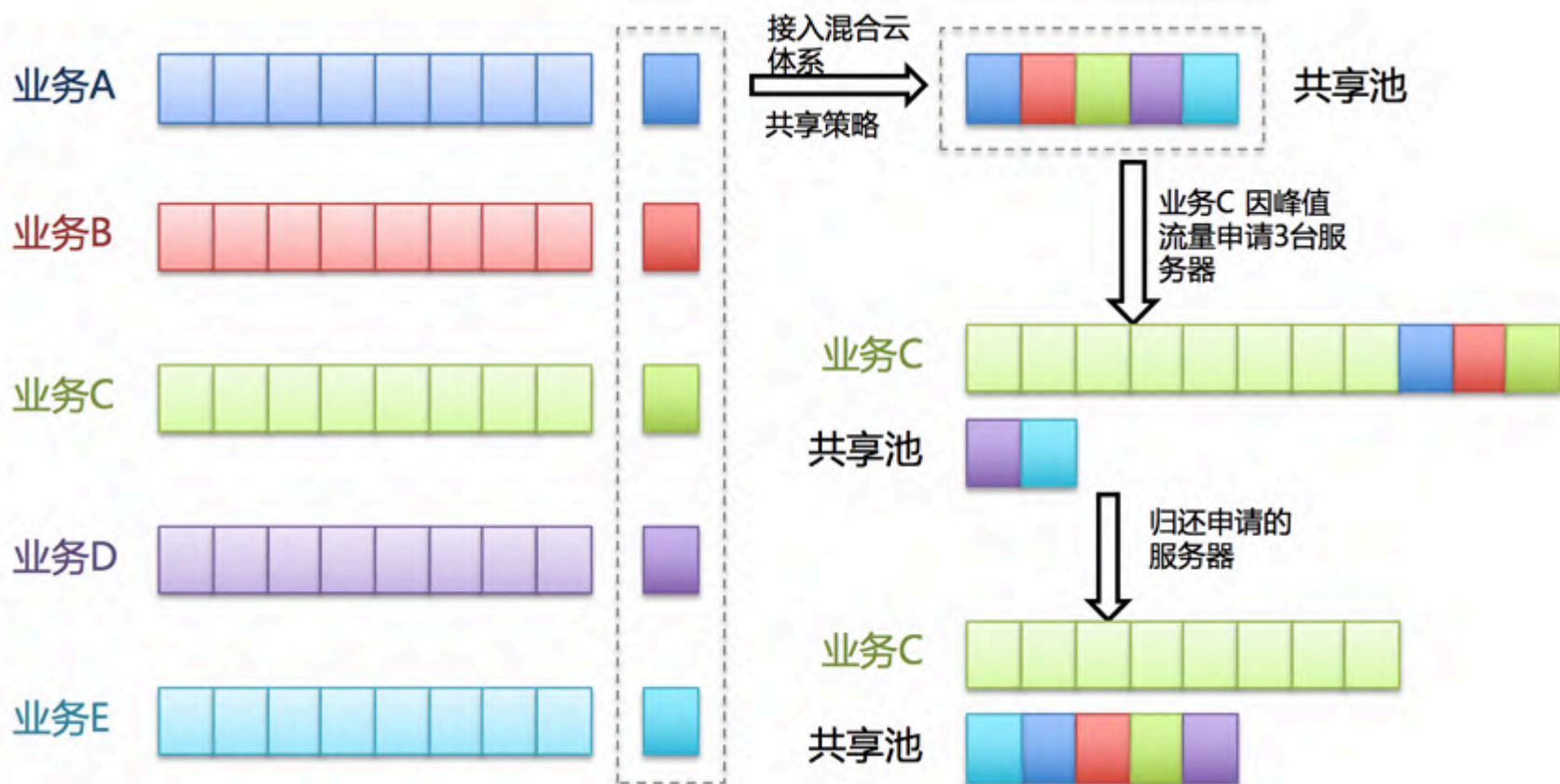


Part 2

Weibo DCP 基础设施



私有“云” 化零为整



- 封装阿里云接口

- 阿里云golang sdk
- goroutines并发
- chan异步调用
- websocket和前端同步状态

- 遇到的问题

- ECS前后端（瑶池和后羿）状态不同步，sleep解决
- 并发数限制，单vpc下有锁，50秒拿不到锁即超时

- 各业务方管理自己的VM镜像，加快构建环境的速度



Java		php	
cAdvisor 0.7.1.fix			
Mesos 0.25			
Swarm 1.0.0			
Consul 0.5.2			
Docker 1.6.2	Registry v2	Daemon Wrapper	Registry v1
devicemapper-direct-lvm		Docker 1.3.2	
		devicemapper-loop-lvm	
CentOS 7.1.1503/3.10.0-229.el7.x86_64		CentOS 6.6/2.6.32	



● 微博内网

- 已有模块稳定
- 无法SSH



ANSIBLE

● 阿里云

- 依赖少
- 易于编写插件
- 集成SSH登录权限
- 高并发下水平扩容

- ECS创建完成后，自动执行初始化



● 1.9.4

- 模块能够满足使用
- 多线程执行并发任务
- CPU占用低

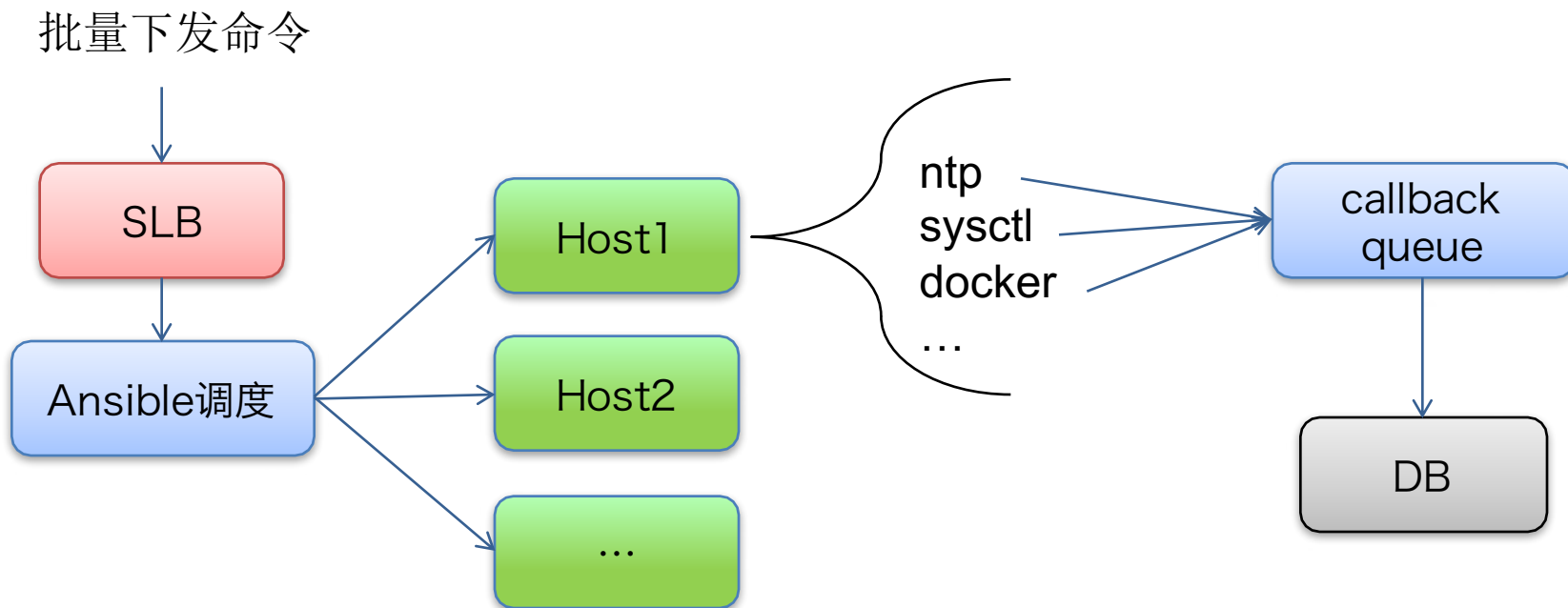
● 2.1.0

- 支持更多特性，如curl模块
可设置header参数
- 多进程执行并发任务
- CPU占用高

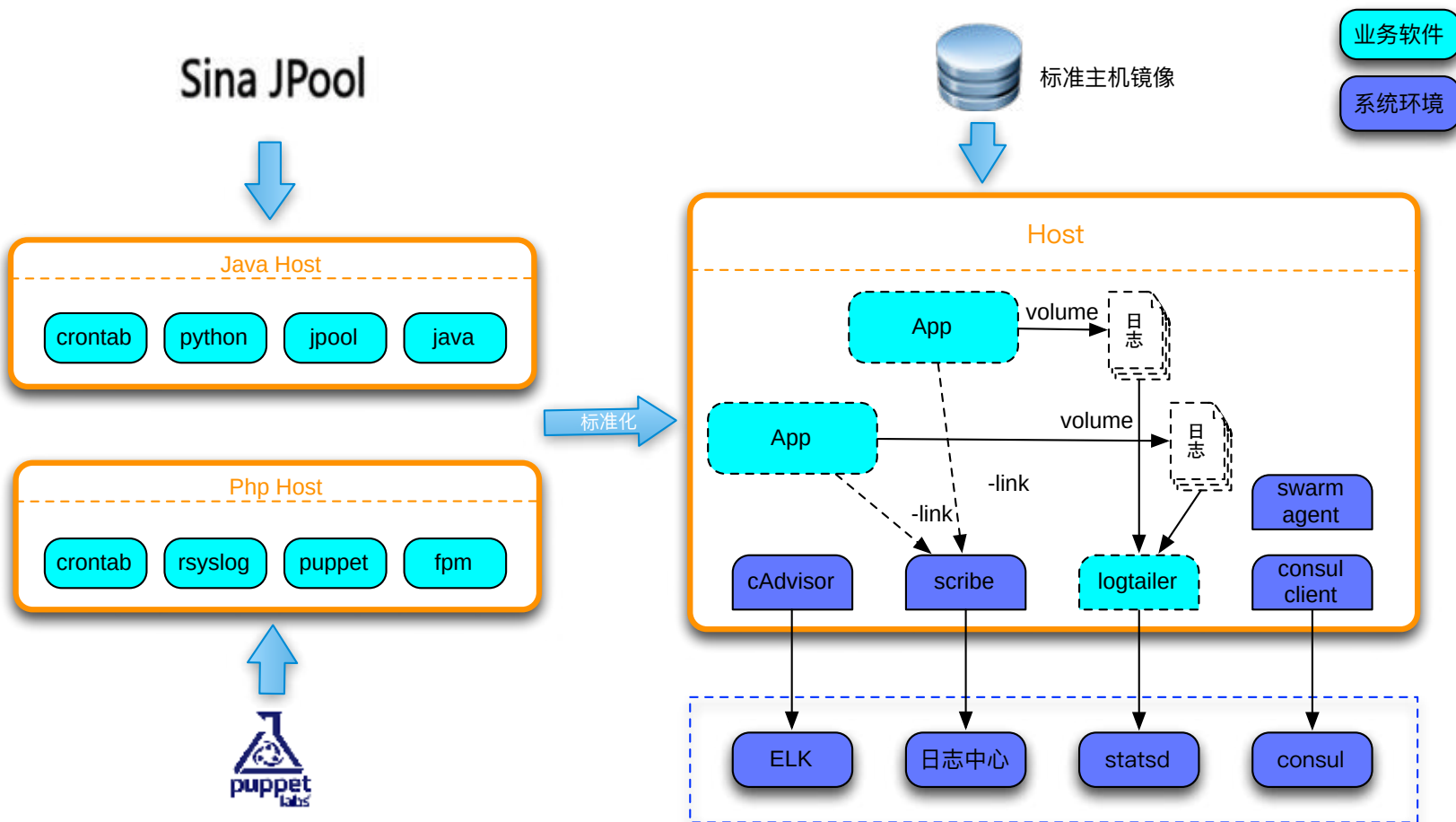
```
while True:
    try:
        result = self._read_worker_result()
        if result is None:
            time.sleep(0.0001)
            continue

        # send callbacks for 'non final' results
        if '_ansible_retry' in result._result:
            self._send_result(('v2_playbook_retry', result))
            continue
```

- SSH开启pipelining和ControlPersist
- Ansible前端增加调度队列，单机控制并发数
- 在VM镜像中预先安装部分软件，如dnsmasq等
- 自定义callback，异步向队列中写入结果

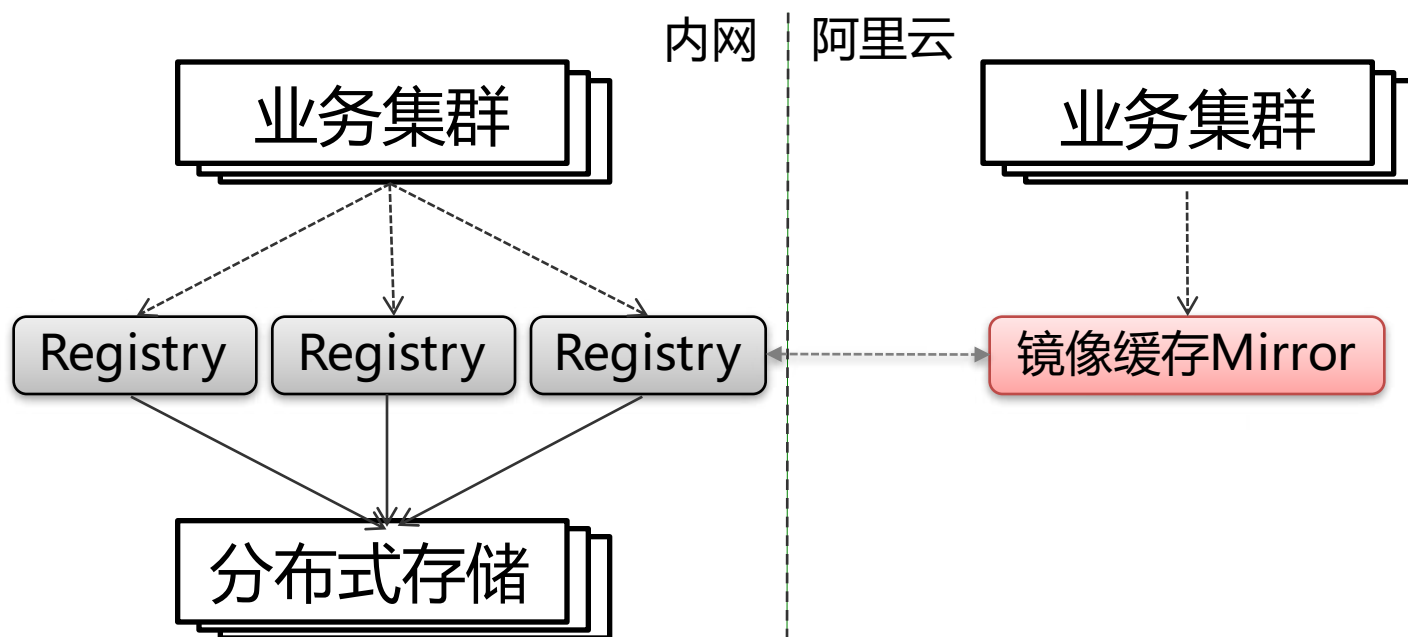


打破差异 - 标准化运行环境



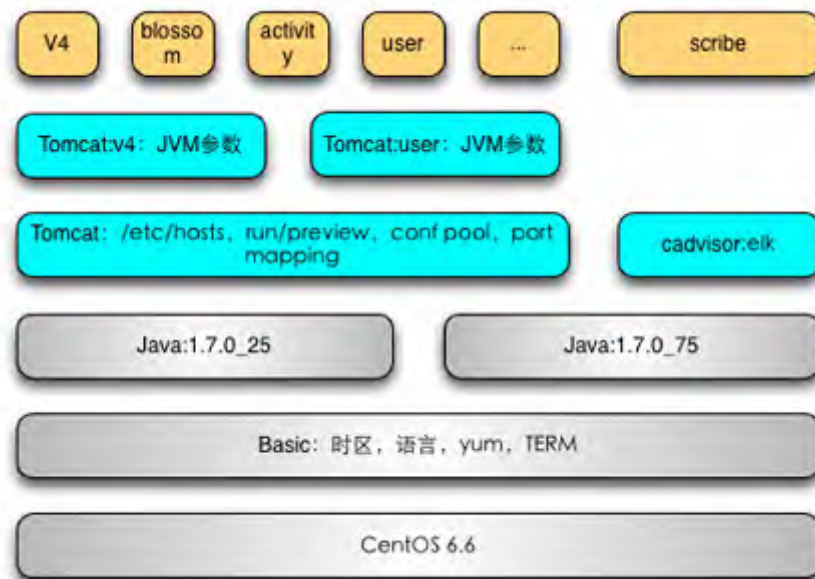
● 构建私有Registry Hub

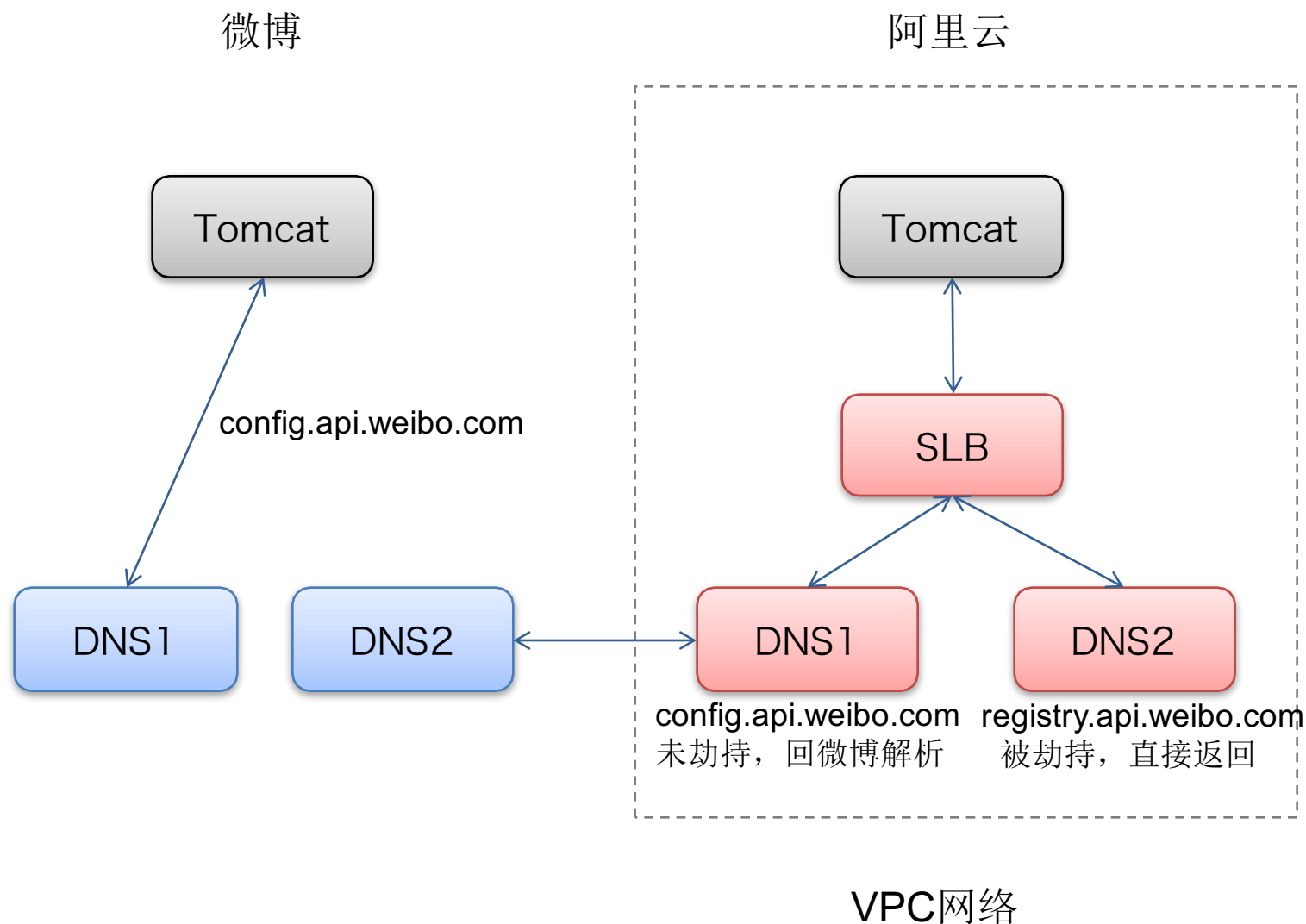
- docker-registry : V1 -> V2
- storage driver : Ceph
- docker-registry-frontend : Nginx



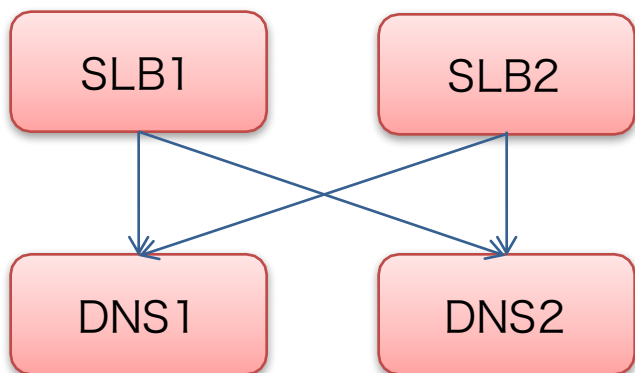
● 镜像服务

- 分层设计，逐层复用
- 基础环境/运行时/容器/业务
- 优化大小，dockerignore
- 禁止使用latest

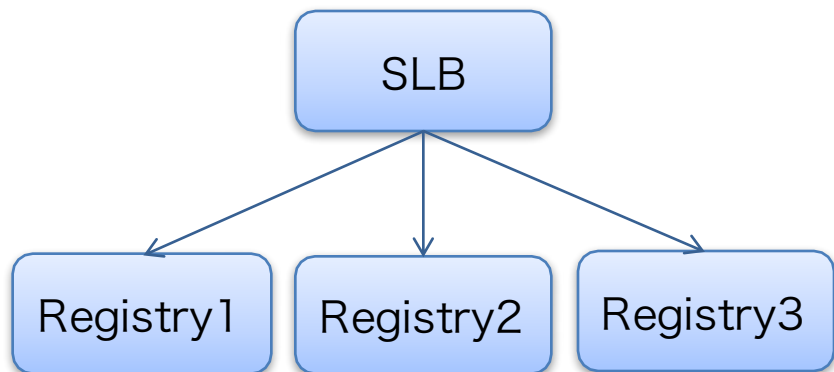




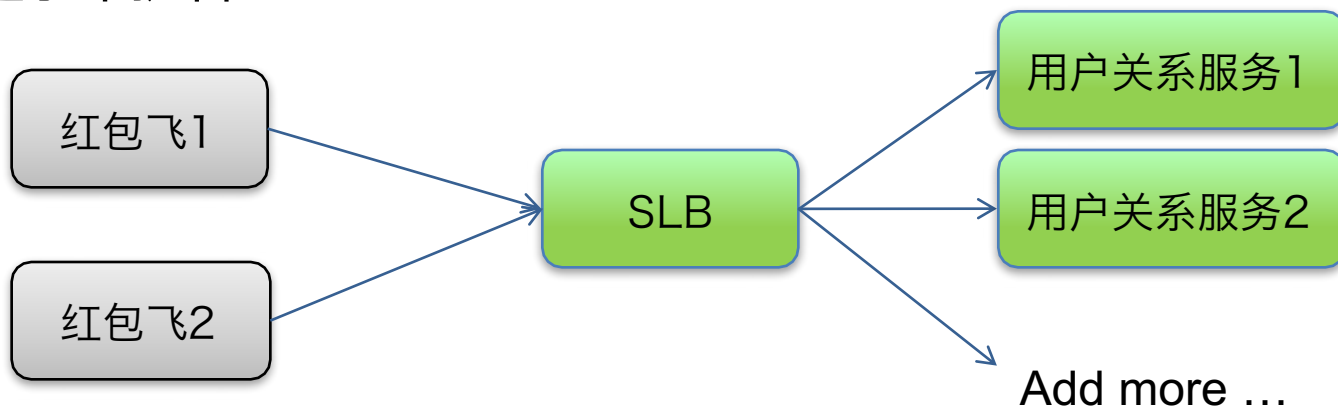
- DNS高可用



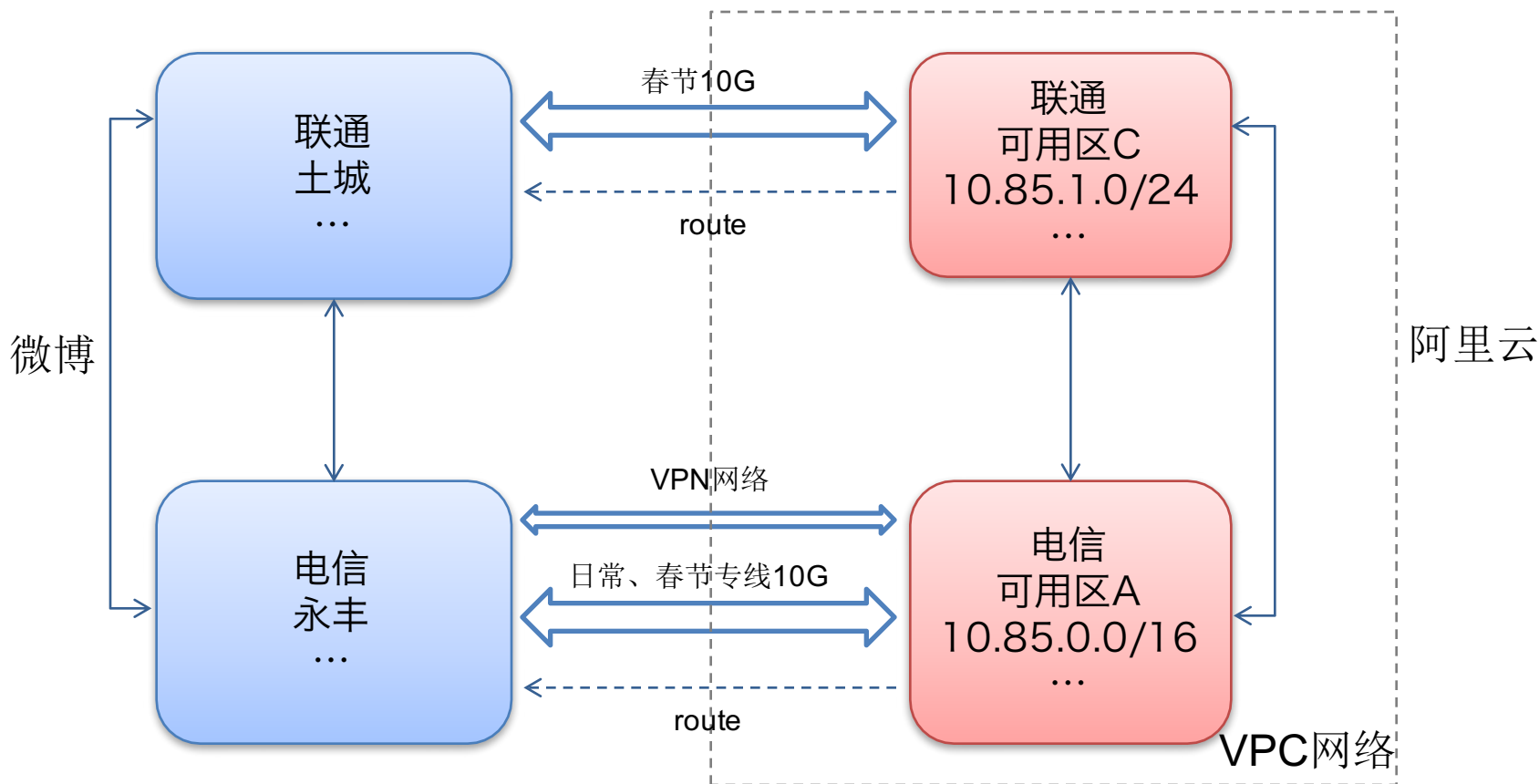
- Registry负载均衡

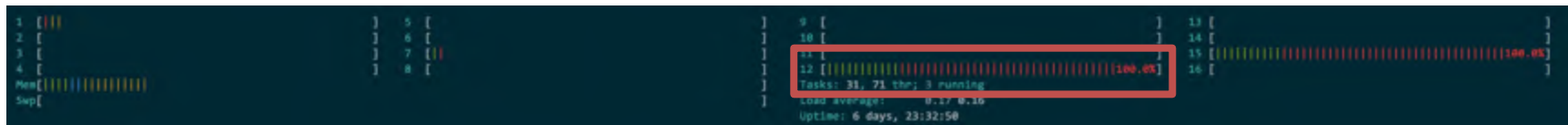


- 快速水平扩容



- 通过路由配置分散两条专线压力，可随时切换
- VPN做备用
- 不同业务划分网段，便于监控专线带宽使用情况





测试CPU：

`sysbench --test=cpu --cpu-max-prime=10000 run`

`sysbench --test=threads --num-threads=64 --thread-yields=2000 --thread-locks=2 run`

测试磁盘：

`fio -direct=1 -iodepth=64 -rw=randwrite -ioengine=libaio -bs=16k -size=10G -
numjobs=1 -runtime=1000 -group_reporting -name=/path/testfile`

测试内存：

`mbw -q -n 10 256`

测试带宽：

`netperf -H host -l second -t [TCP_STREAM|UDP_STREAM|TCP_RR|TCP_CRR|UDP_RR] -s
localBufferSize -S remoteBufferSize -m localPackageSize -M remotePackageSize -D
TCP_NODELAY`

测试业务



Part 3

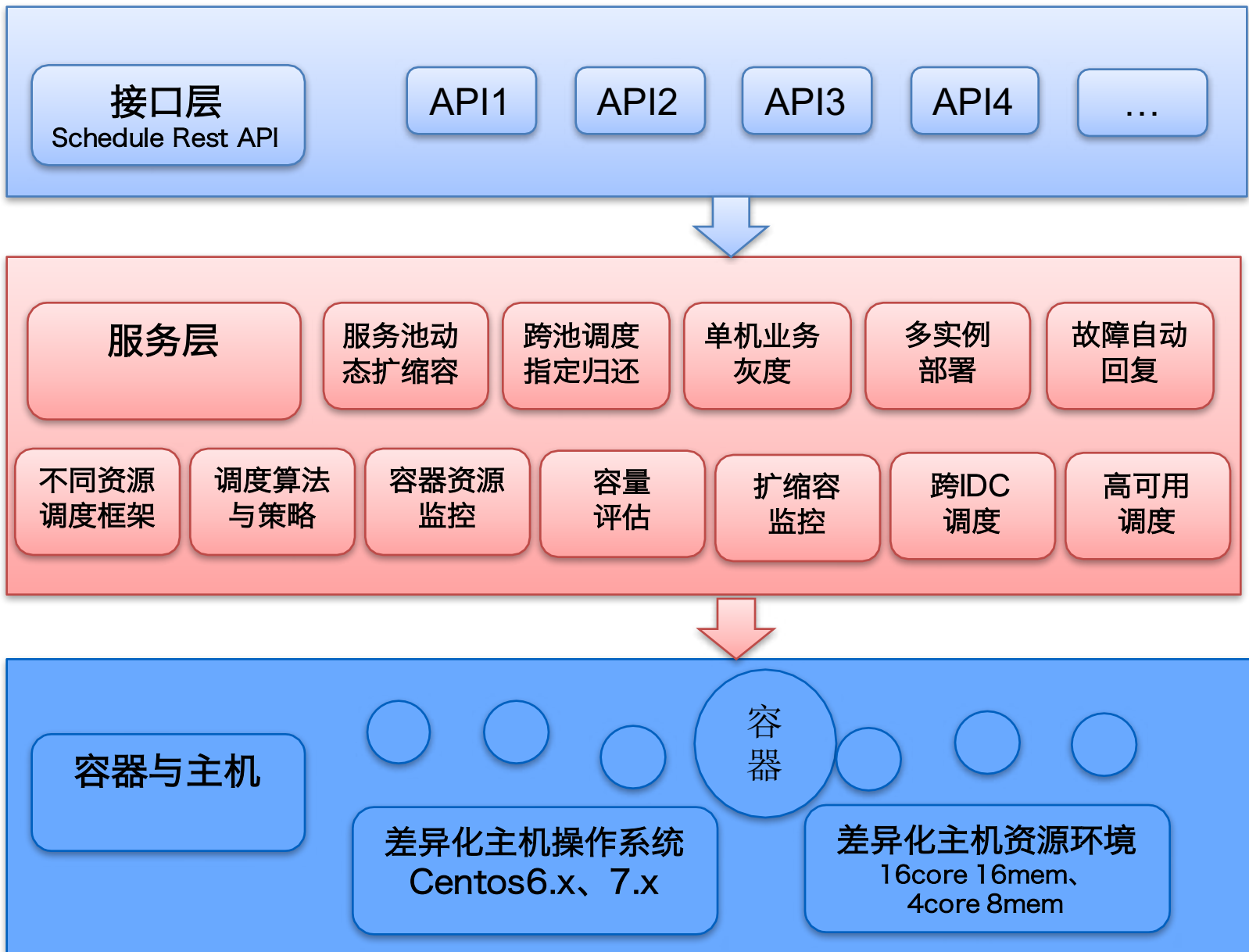
Weibo DCP 弹性调度

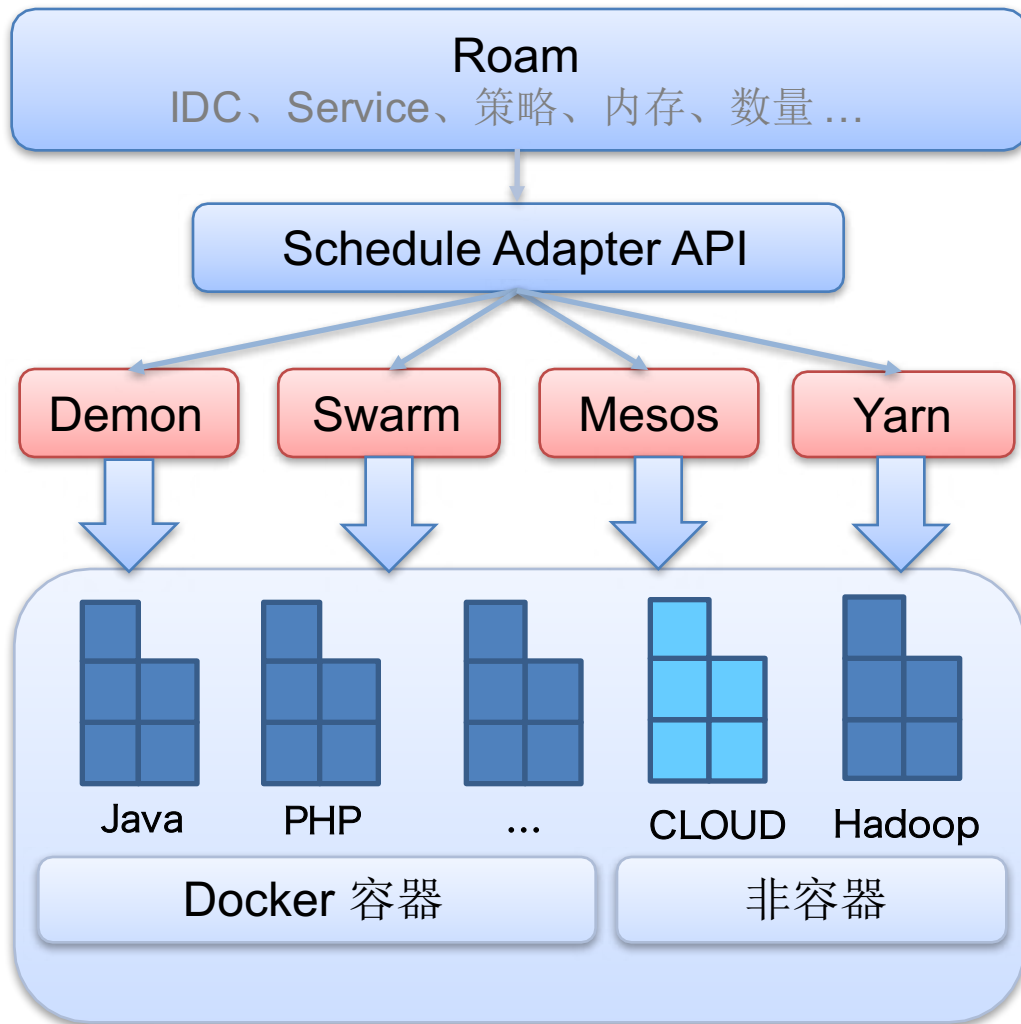
弹性调度 - 选型

-	Swarm	Mesos	k8s
架构特点	针对Docker体系	借鉴Borg理念	源自Borg, 原生Docker
与Mesos结合	社区完善中	成熟	社区积极推荐
隔离机制	Docker	Mesos/Docker/其它	Docker
资源类型	内存 CPU 端口	内存 CPU 端口 Ulimit	内存 CPU 端口 Ulimit
调度非Docker	No	Yes	No
主机分组	Docker进程标签	Slave配置	Slave配置
打分策略	资源使用情况	资源使用情况	资源使用情况+应用节点均衡
端口编排	No	Yes	Yes
网络模式	Docker原生	支持自建	支持自建
主高可用	双主切换	zk	etcd
扩缩容	Roam二次开发	API修改实例数	API修改实例数
应用节点健康检测	No	Yes	Yes
应用节点自动故障转移	No	Yes	Yes
服务发现	Consul	zk	etcd
负载均衡	No	Haproxy	Kube-proxy
DNS	No	No	skydns
使用复杂度	低	中	中
集群规模	小	中	中, 在提升
生产使用	尚无大规模使用	业界大规模使用	业界大规模使用

弹性调度 - 选型

需求：
快速迭代
实现内网
计算资源
统一管理
调配，公
有云上获
得计算资
源，快速
自动化资
源调度与
应用部署





服务发现

调度框架

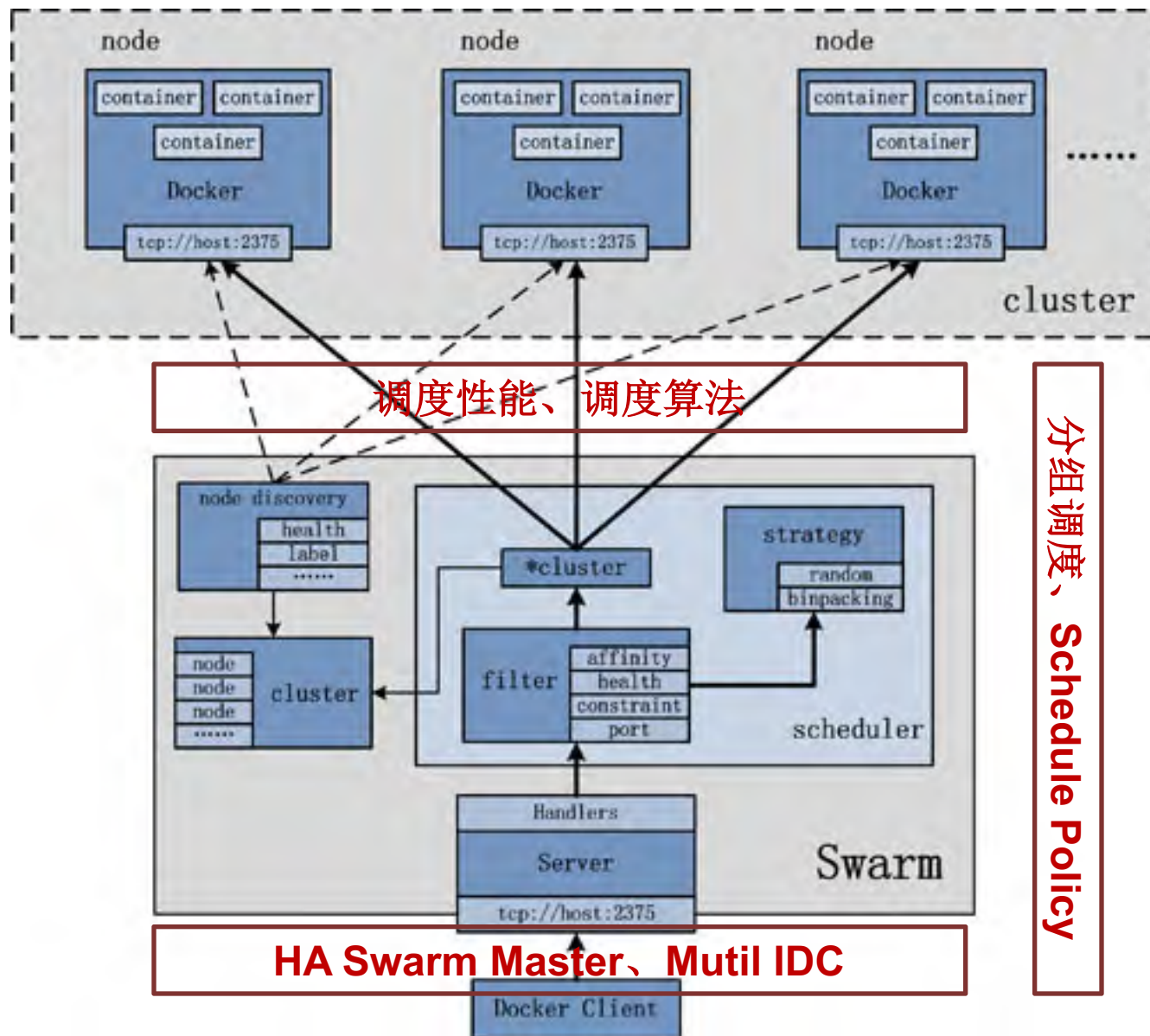
自定义调度

资源管理

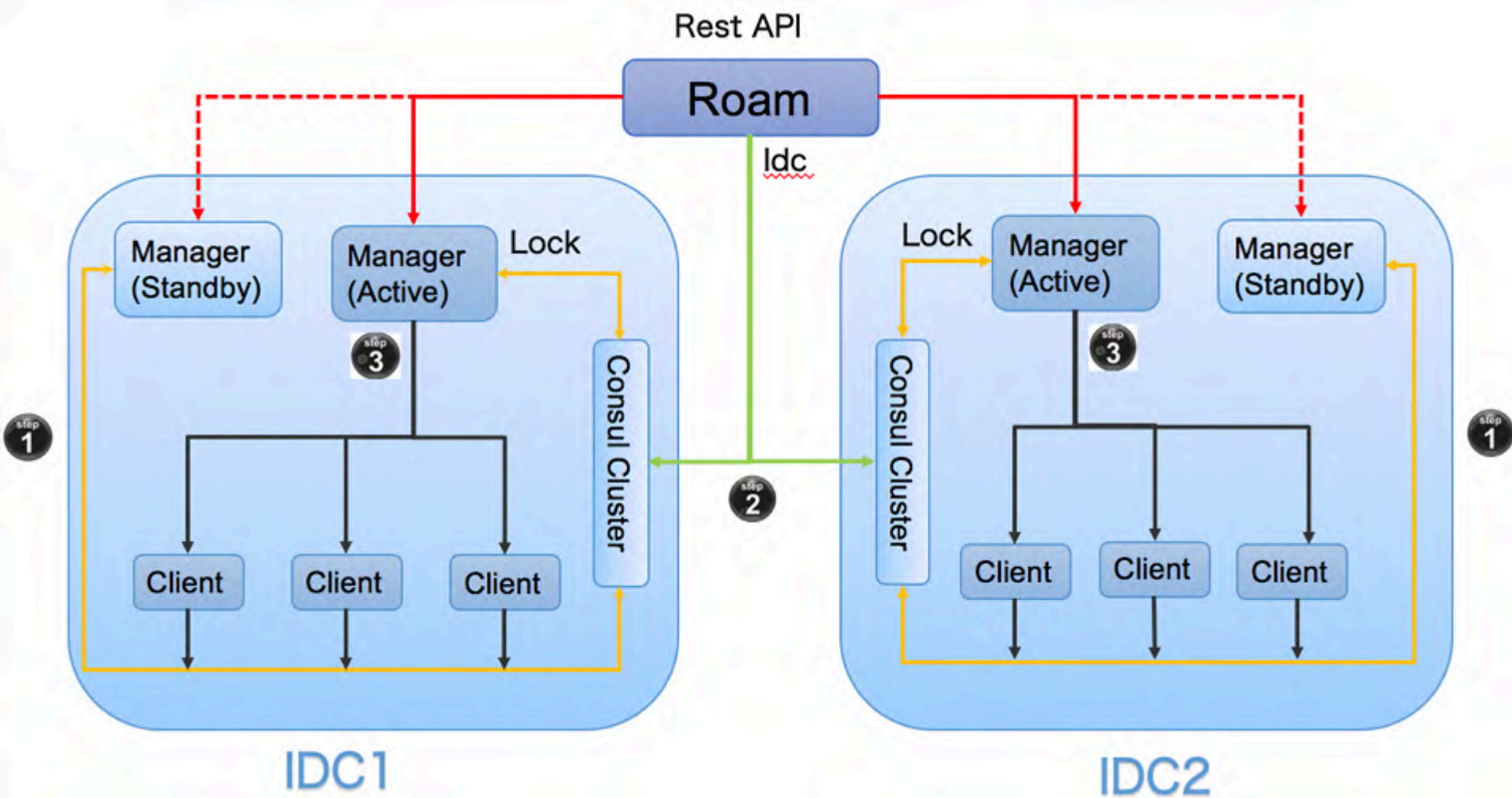
容量评估

监控报警

动态调度 - Swarm架构



动态调度 - 多IDC、高可用、可扩展



- 调度=主机 or 容器过滤 + 策略选择
- 过滤器filter
 - Node Filters: health (会根据节点状态进行过滤, 会去除故障节点)、
constraint (约束过滤器、Label分组调度)
 - Container Configuration Filters: affinity (亲和性过滤器)、dependency (依赖过滤器)、Port (会根据端口的使用情况过滤)
- 调度策略
 - 根据各个节点的可用的CPU, Mem及正在运行的容器的数量来计算应该运行容器的节点进行打分, 剔除掉资源不足的主机, 然后策略选择: spread、binpack、random
 - Binpack: 在同等条件下, 选择资源使用最多的节点
 - Spread: 在同等条件下, 选择资源使用最少的节点
 - Random: 随机选择一个
- 调度颗粒度
 - Memory: `docker run -m 1g ...`
 - CPU: `docker run -c 1 ...`

```
if config.CpuShares > 0 {  
    cpuScore = (node.UsedCpus + config.CpuShares) * 100 / nodeCpus  
    //cpuScore=（物理机已用CPU+本次需用CPU）*100/物理机CPU  
}  
if config.Memory > 0 {  
    memoryScore = (node.UsedMemory + config.Memory) * 100 / nodeMemory  
    //memScore=（物理机已用内存+本次需用内存）*100/物理机内存  
}
```

```
if cpuScore <= 100 && memoryScore <= 100 {同时满足可用内存、CPU  
    weightedNodes = append(weightedNodes, &weightedNode{Node: node, Weight: cpuScore + memoryScore})  
}
```

资源只与容器Create时配置有关，与运行时实际使用资源情况无关。无论容器是否由Swarm创建，无论容器处在何种状态，只要配置了资源限额，调度时均会计算在内！

- 提供通用HTTP API, 适配不同分布式资源调度框架
- 不同调度策略与算法
- HA Swarm Master高可用
- 多机房自动适配
- 单IP Docker Deamon下发执行机制
- 容器资源监控
- 扩缩容监控
- 容器资源评估
- ...



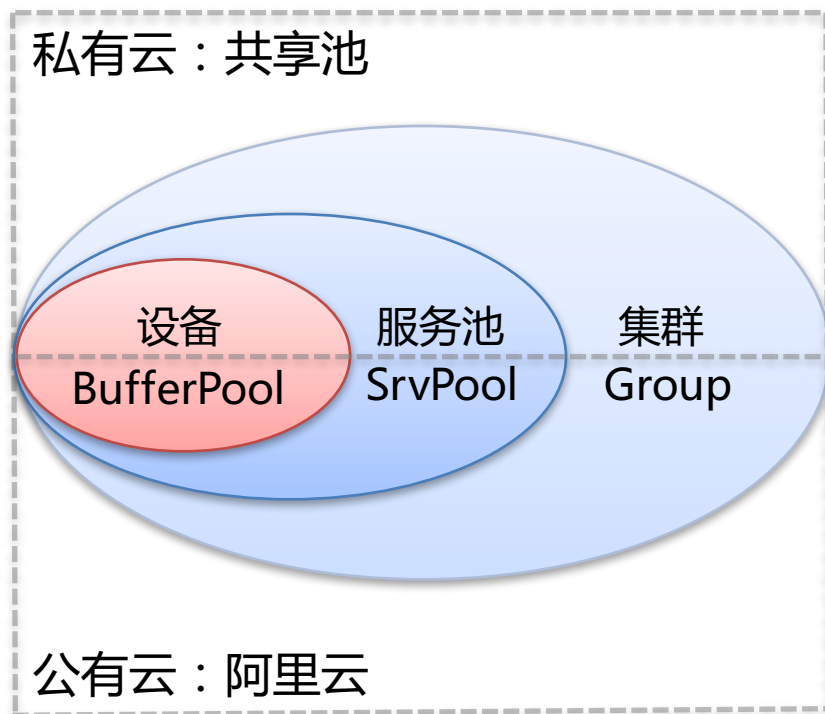
Part 4

Weibo DCP 编排与实现

DCP-功能模块图



- 核心问题：设备从哪来？
- 设备方案：内网共享池 + 公有云=BufferPool
- 服务：IP + Port

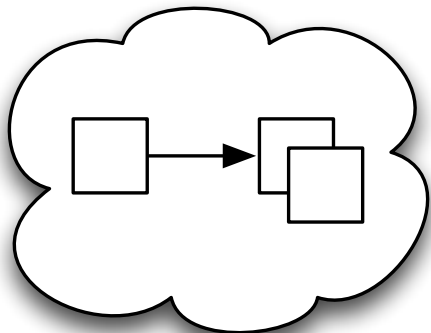


层级关系

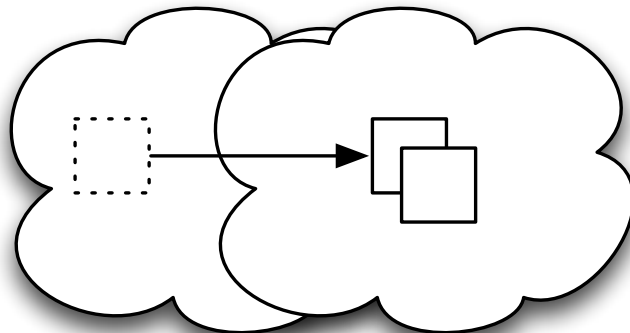
- DCP：分为多个集群
- 集群：为独立平台，对应业务线
 - 集群内：自由调度（跨池）
 - 集群外：配额调度
- 服务池：同一业务线的同构服务
- 设备：buffer池 = 共享池 + ECS

DCP-大规模集群扩容方式

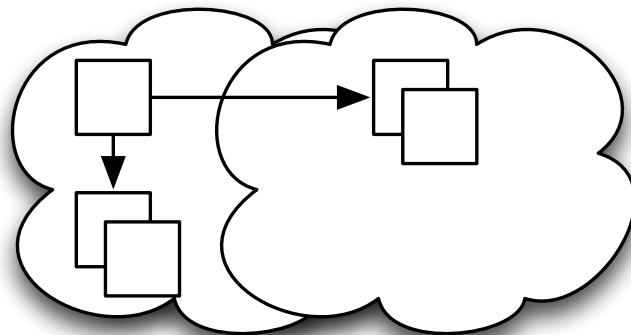
私有内弹性扩容

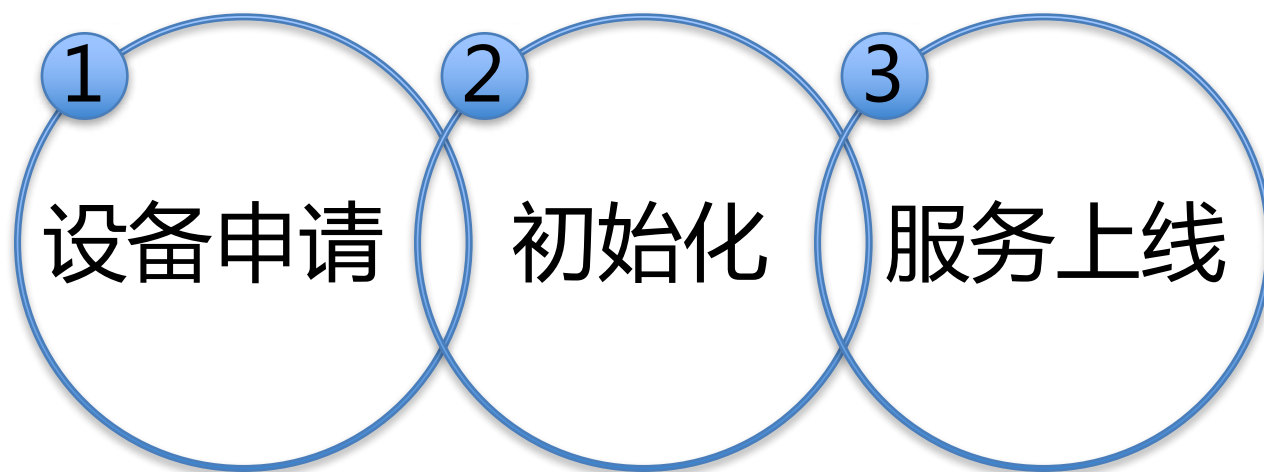


扩容到公有云弹性



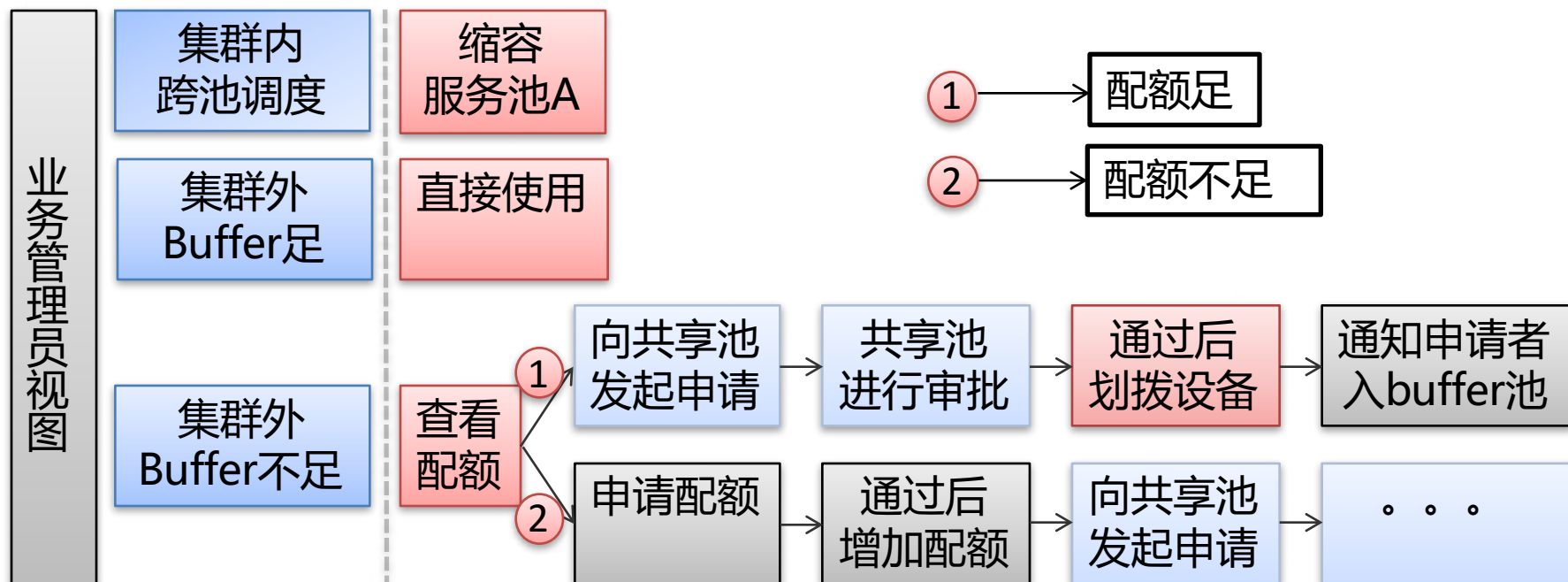
私有云、公有云同时弹性扩容





- 设备申请
 - 内: 共享池
 - 外: 阿里云
- 初始化
 - 包标准化
 - 配置管理
- 服务上线
 - 容器动态调度
 - 服务弹性扩缩容

- 私有云：共享池（离线集群，低负载集群，错峰）
- 公有云：阿里云（动态创建）



DCP-设备申请案例：阿里云主机

交易单

请输入交易单

Go!

资源申请 +

阿里云ECS创建 +

#	交易单	发起方	接收方	原因	交易数量	机型	CPU	MEM	开始时间	结束时间	状态	#
---	-----	-----	-----	----	------	----	-----	-----	------	------	----	---

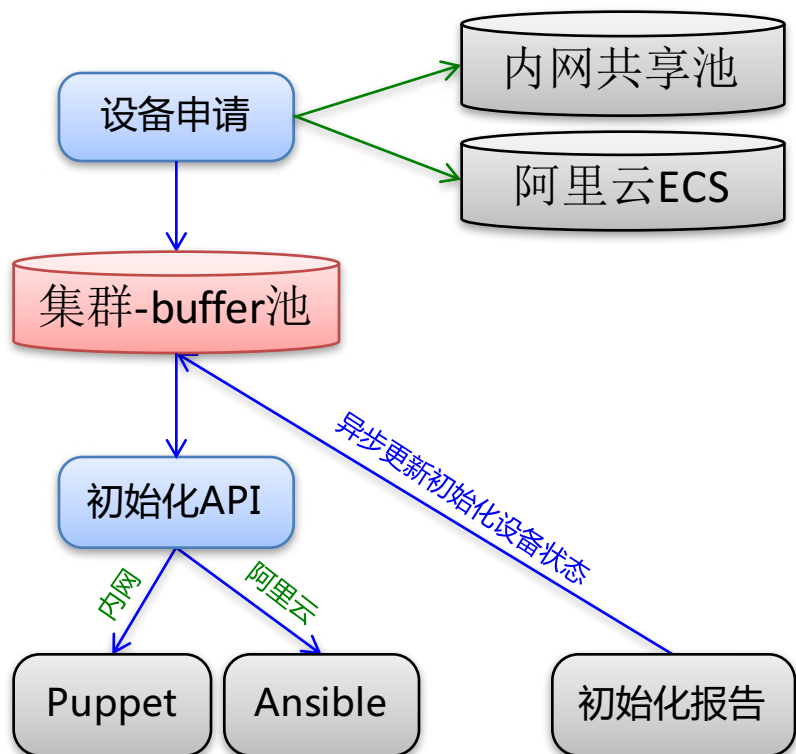
发起申请

×

发起方	WeiboPatform_Platform	可用地域	北京地区
可用区域	北京可用区C	VPC	weibo_vpc
交换机	Weibo_Switch_C_1	安全组	beijing-a-default
机器规格	16Cores-16GB	CPU	16
内存	16	cmdb服务类型	JAVA应用
申请机器数量	100	镜像	weibo_hongbao_v5_4
结束时间	2016-01-21T01:29:00Z		

DCP-大规模集群第二步:初始化

DCP初始化流程



线上案例

Buffer池管理

全部 Buffer池 使用中

default 全部机器类型 是否借用 机器状态 全部录入者

批量更新配置 批量初始化 批量反初始化 批量归还 批量停服务 批量进共享池

#	虚点号	内网IP	机器来源	所属集群	使用方	机器类型	是否借用	使用状态	机器状态
1	YY14070655	10.13.0.99	在线	微博平	default	内网	借用(共享池)	空闲	未初始化

初始化机器

初始化环境 * production

当前操作, 将影响如下机器

涉及机器数: 1

涉及IP列表如下:

10.13.0.99(空闲_未初始化)

DCP-初始化报告

63	547	微博平台/WeiboPatform_Platform	微博平台/共享池	40	阿里云	16	16	02.07 12:03	02.07 20:03
64	546	微博平台/WeiboPatform_Platform	微博平台/共享池	40	阿里云	16	16	02.07 12:00	02.07 20:00
65	545	微博平台/WeiboPatform_Platform	微博平台/共享池	40	阿里云	16	16	02.07 11:58	02.07 19:58
66	544	微博平台/WeiboPatform_Platform	微博平台/共享池	40	阿里云	16	16	02.07 11:55	02.07 19:55

获取申请单详情 / 535

行为	总耗时	平均耗时	最高耗时	最低耗时	成功率
创建	207	178	207	166	97.5%
初始化	172	108	131	79	97.5%
汇总(40台)	6分19秒	4分46秒	5分38秒	4分5秒	-

关闭

74	536	微博平台/WeiboPatform_Platform	微博平台/共享池	6	阿里云	16	16	02.07 03:16	02.07 11:13
75	535	微博平台/WeiboPatform_Platform	微博平台/共享池	40	阿里云	16	16	02.07 02:14	02.07 10:12

● 扩容

- 输入：服务池名称、服务类型，容器类型、数量、镜像地址等
- 输出：扩容报告

● 记账中心

- 资源拥有者、使用方、型号、使用时间、日期、信用等

● 容器服务类型：按照dPxx(dockerPxx)命名

dP01 8核12G	dP02 12核12G	dP03 12核16G	dP04 16核16G
---------------	----------------	----------------	----------------

任务详情

社区 文档 王关胜

任务信息 任务参数和属性

任务名称	auto_deploy_openapi_StartDocker_20160207_201117		
版本号			
保留版本号			
备注			
任务类型	auto_StartDocker	应用名称	openapi_webv2_docker
应用路径	/data1/weibo	超时时间	700
是否启动	是	执行比例	100
是否自动执行	否	自动执行步长	2000

任务概览 概要统计

应用池子名称	总台数	未分发	成功	执行中	跳过	超时	失败	进度
openapi_webv2-aliyun_tc-core-inner-docker	44	0	33	0	11	0	0	100%

DCP-扩容任务耗时

任务列表

2016-02-07



2016-02-07



扩容

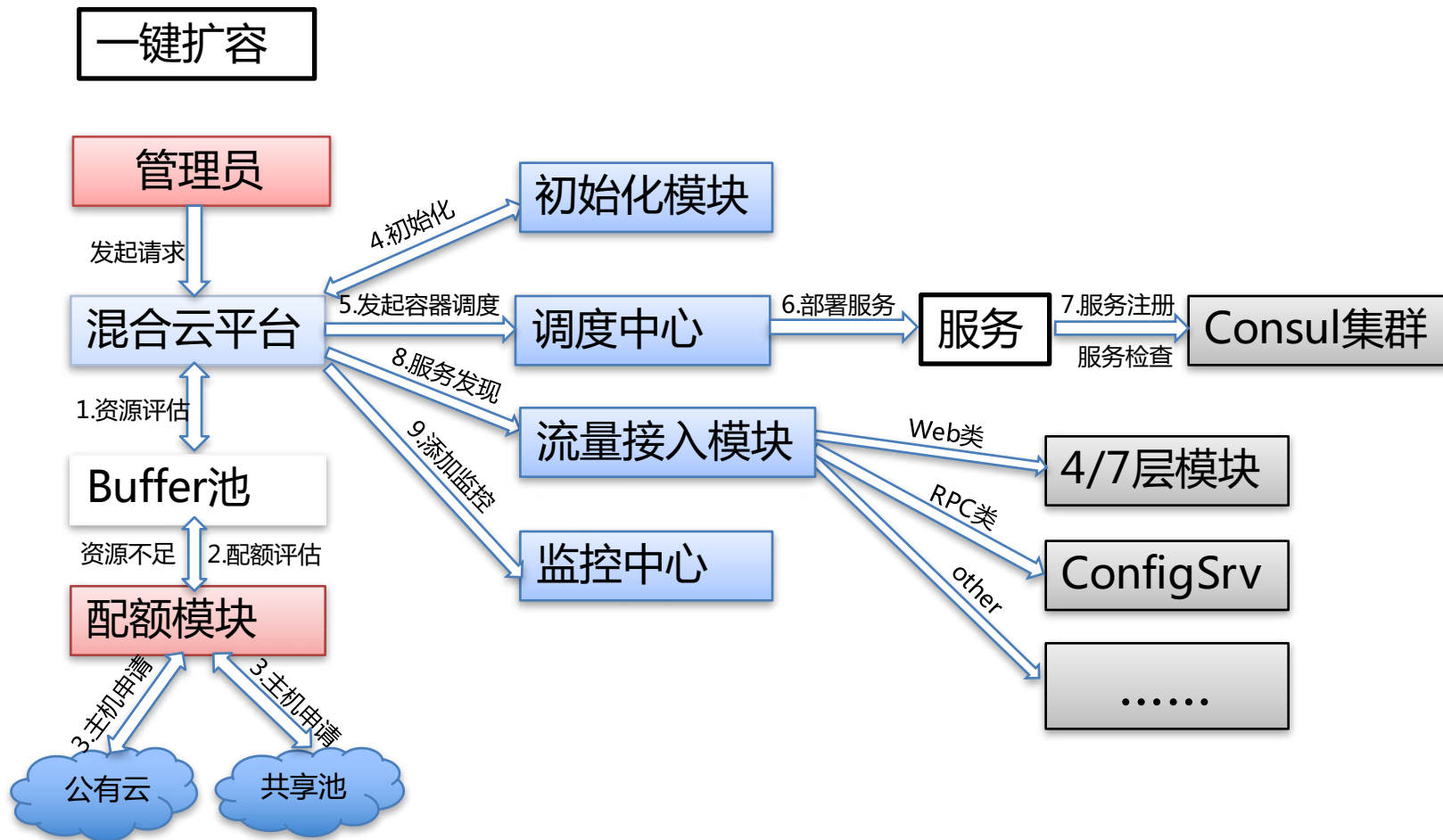


liupeng7

Go!

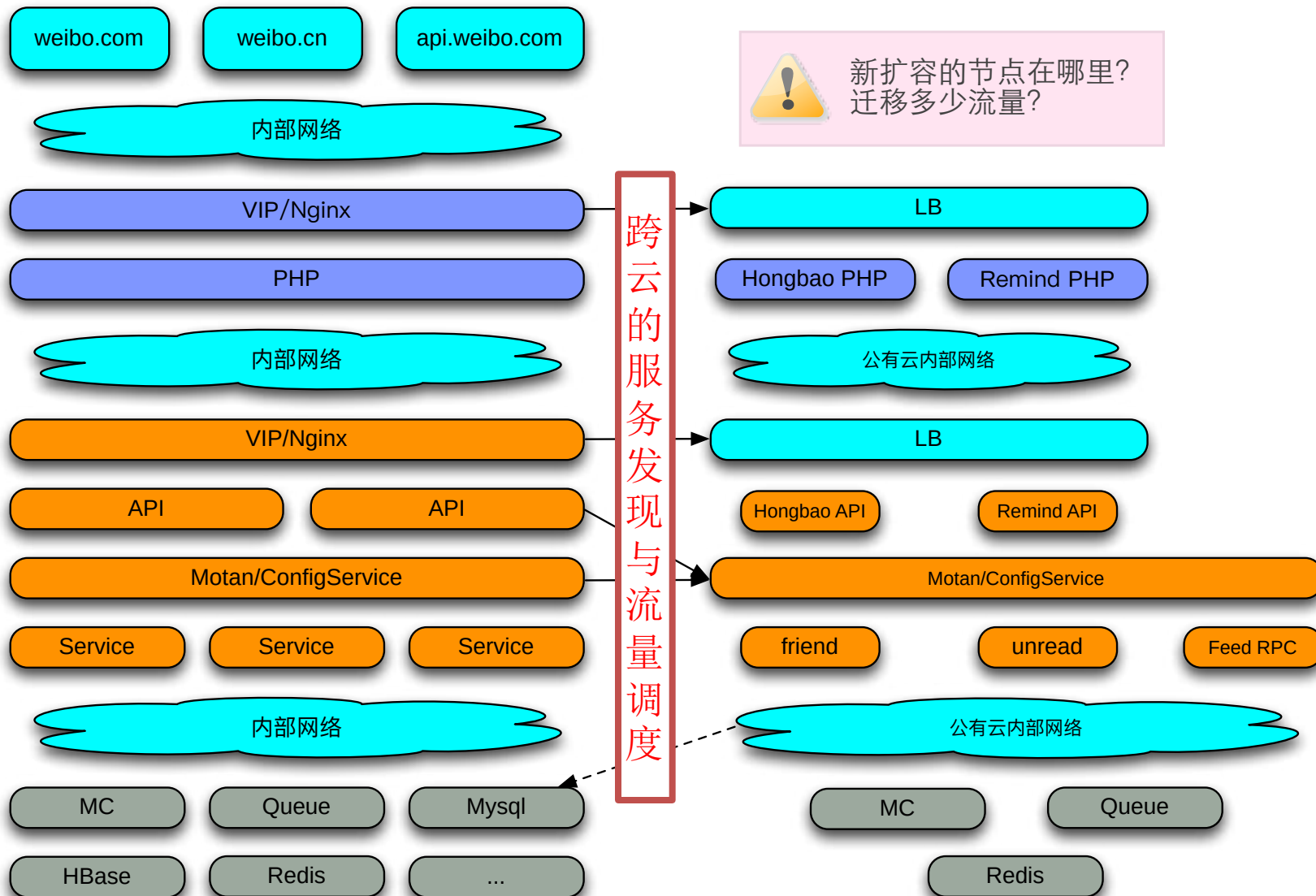
#	任务ID	操作	开始时间	结束时间	耗时
1	19839	扩容	2016-02-07 22:01:58	2016-02-07 22:08:19	6分31秒
2	19834	扩容	2016-02-07 21:43:22	2016-02-07 21:50:51	7分39秒
3	19825	扩容	2016-02-07 20:57:38	2016-02-07 21:09:05	11分37秒
4	19821	扩容	2016-02-07 20:44:40	2016-02-07 20:54:51	10分21秒
5	19816	扩容	2016-02-07 20:11:17	2016-02-07 20:21:35	10分28秒
6	19815	扩容	2016-02-07 20:08:30	2016-02-07 20:21:17	12分57秒
7	19801	扩容	2016-02-07 19:21:33	2016-02-07 19:26:31	5分8秒
8	19797	扩容	2016-02-07 18:47:05	2016-02-07 19:02:03	15分8秒
9	19796	扩容	2016-02-07 18:45:08	2016-02-07 19:07:31	22分33秒
10	19765	扩容	2016-02-07 12:37:34	2016-02-07 12:56:57	19分33秒
11	19762	扩容	2016-02-07 12:20:01	2016-02-07 12:32:25	12分34秒
12	19757	扩容	2016-02-07 11:41:08	2016-02-07 11:47:25	6分27秒
13	19755	扩容	2016-02-07 11:32:15	2016-02-07 11:36:47	4分42秒
14	19754	扩容	2016-02-07 11:25:52	2016-02-07 11:30:58	5分16秒

DCP-弹性扩容: 流程



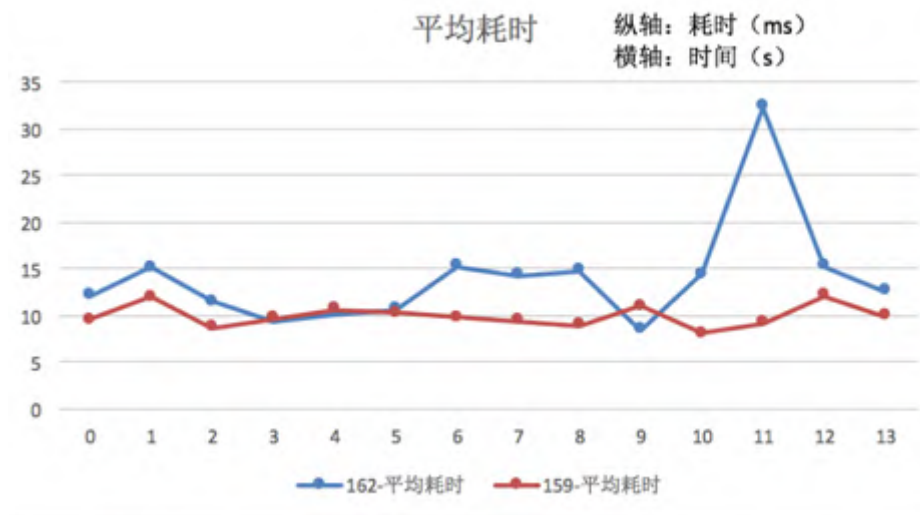
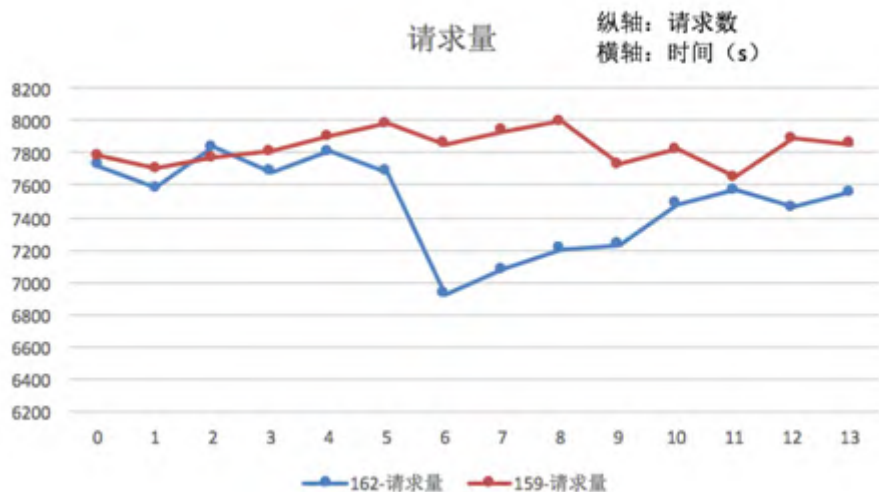
关键点：服务发现

流量要快速、安全的切到弹性节点



问题：Reload损耗

- 开源解决方案大多利用Nginx的Reload机制
- 性能损耗情况:
 - ◆ 请求量：普通reload会导致吞吐量下降10%
 - ◆ 平均耗时：差异不大



微博方案 - nginx-upsync-module

- Nginx Plus的开源版
- 支持基于Consul自动服务发现
- 开源: <https://github.com/weibocom/nginx-upsync-module>



➤ 弹性节点的处理能力不对等

➤ server 10.xx.xx.xx:xxxx max_fails=0 fail_timeout=30s

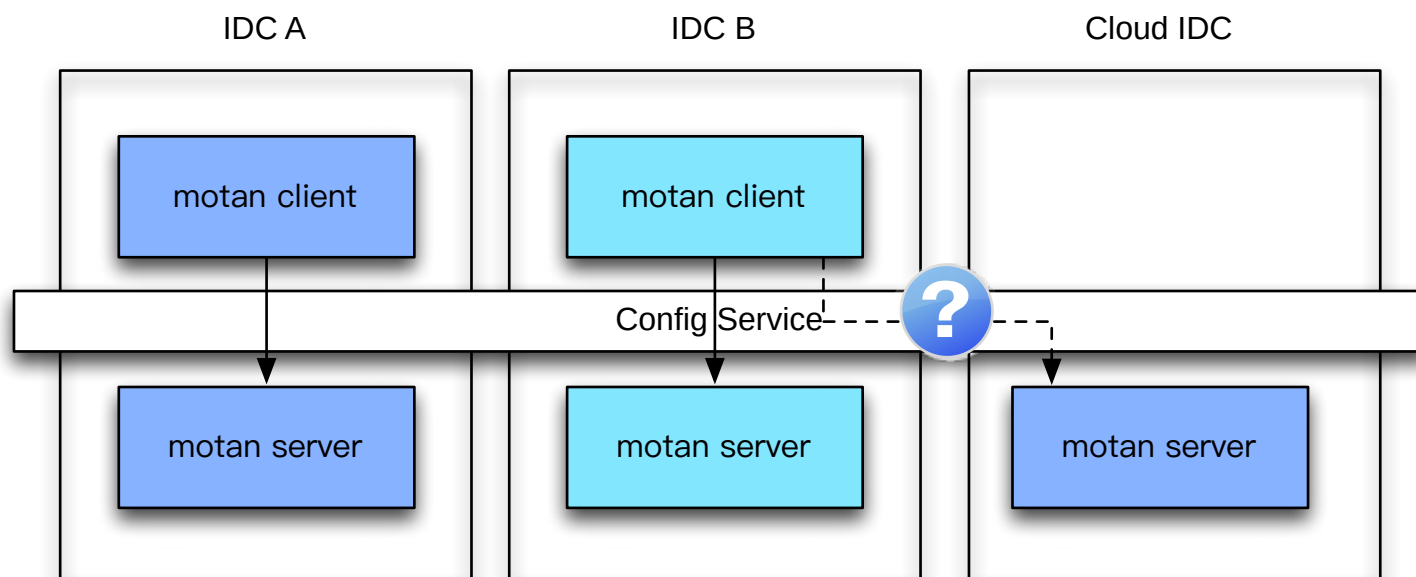
weight=20; #同样的权重导致单点性能恶化

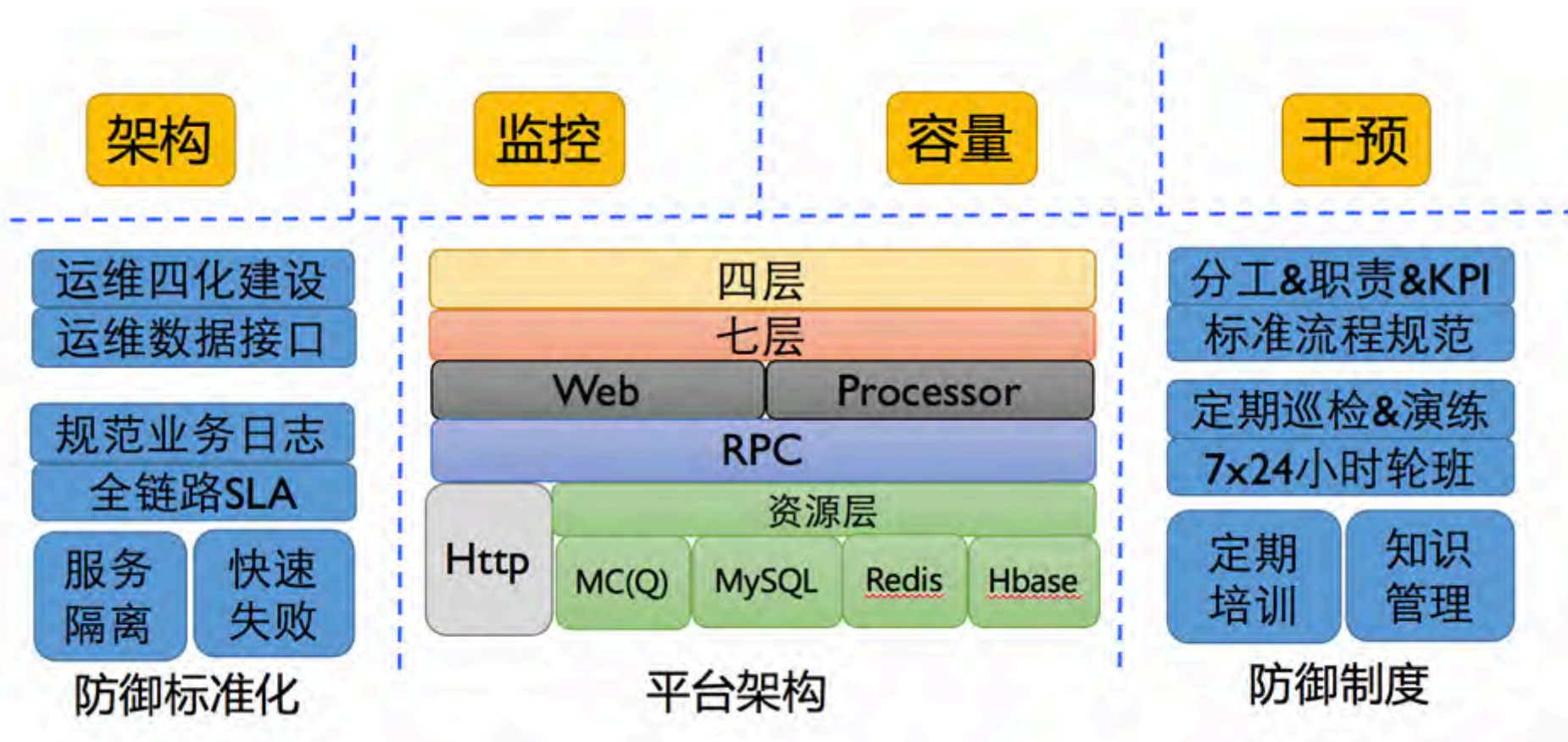
➤ 节点注册计算能力

➤ 所有节点默认权重是20;

➤ 公有云有20%性能损耗，权重=16;

- Motan流量路由模型优先匹配单IDC
 - IDC间切换一般仅在故障处理时采用
- 新版Motan已支持按流量权重配置定向路由





作战图类

云上服务器数

专线带宽

实时容量

实时报警类

接口监控
超过阈值
报警

监控类

QPS监控

平均耗时

4xx,5xx

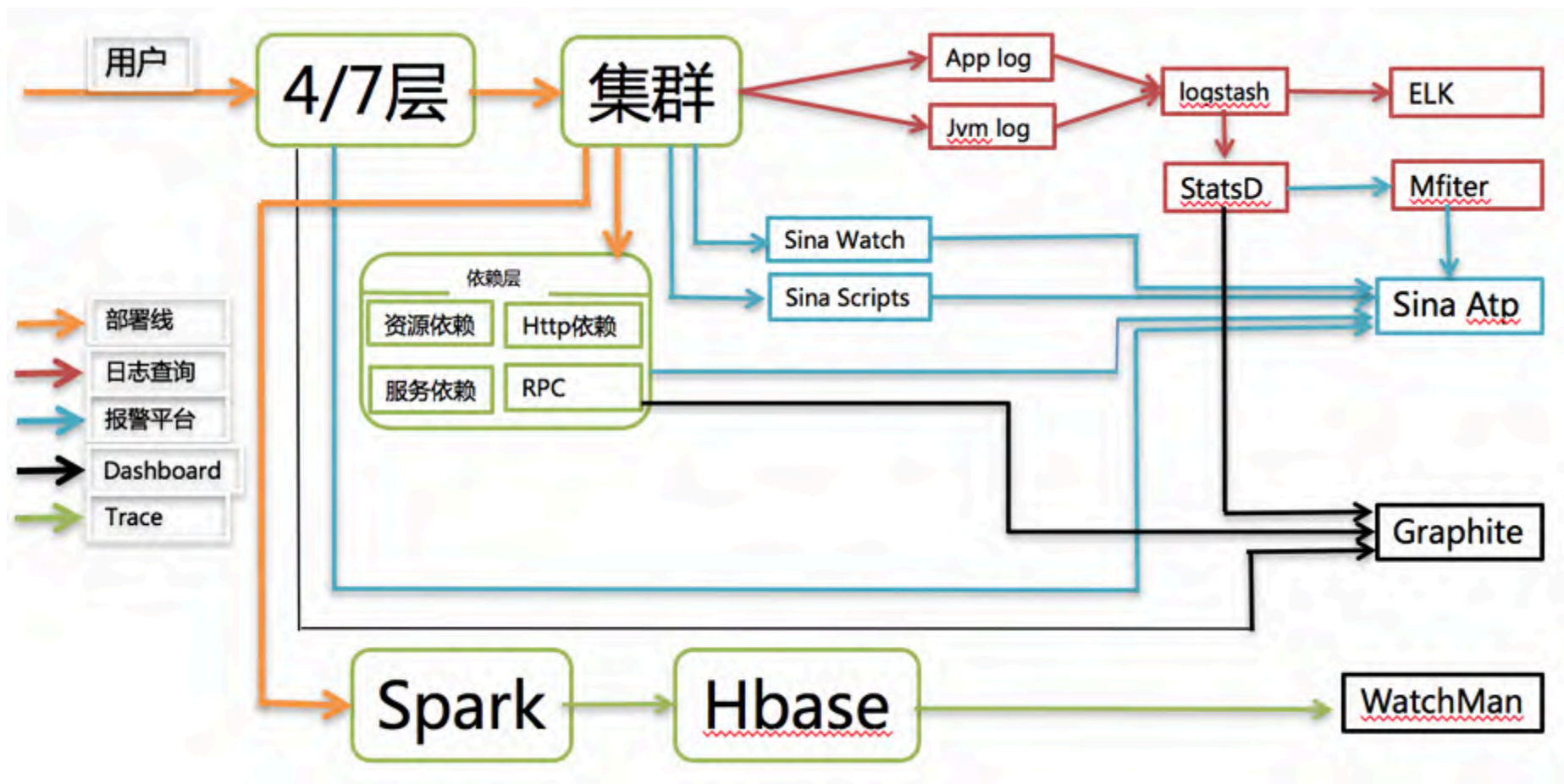
问题定位类

单机slow top

资源slow top

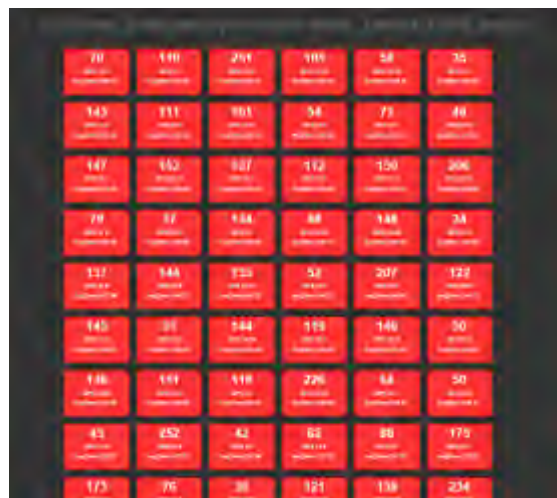
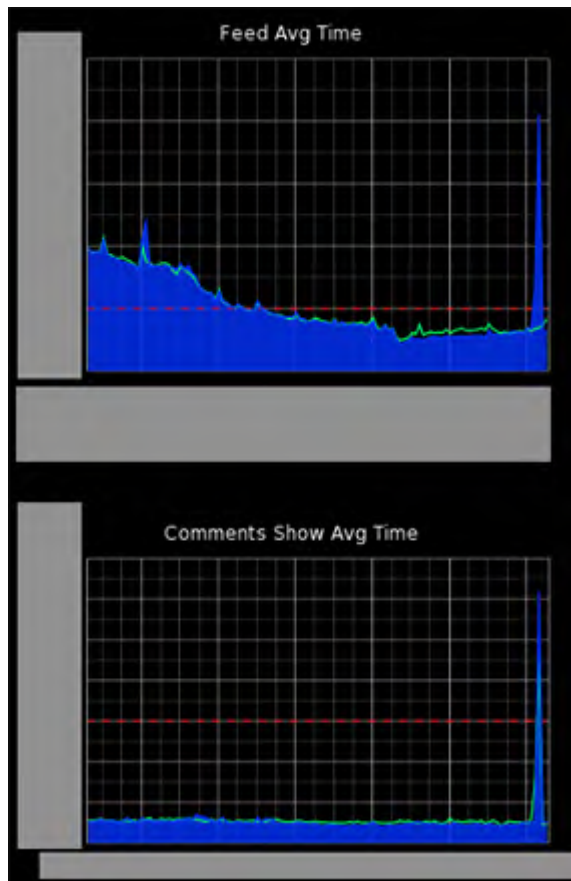
分步耗时

DCP-三节保障：监控体系



DCP-监控覆盖度：单机维度

单机性能恶化



服务器所在位置:osaka1/friendship

System Load:0.01

Tomcat Load:0.01

Swap使用比例:0.42%

本地使用比例:70.28%

10秒内HTTP请求次数:508

10s内错误响应次数:332

10s内超过500ms的响应次数:1

Tomcat繁忙线程比例:0.0%

系统当前停顿时间:0ms

系统10分钟内停顿次数:0

系统10分钟内YGC次数:1

GC后Old区使用比例:27.66%

GC后Old区减少比例:-0.56%

DirectBuffer数量:1263

DirectBuffer池总的容量:52.589467MB

DirectBuffer池使用的容量:52.589467MB

业务容器级报警

往往系统性能恶化是由于单机问题引起
如何快速定位有问题的机器？

DCP-三节保障：容量决策



- 根据服务池单机平均系统指标（CPU idle、mem load）、QPS、带宽、业务SLA综合指标容量评估

- 预案：100+
 - ◆ 日常&应急预案
 - ◆ 重大活动，三节等预案手册
- 服务降级：5000+
- 有效的干预手段



出发





Part 5

春晚实战&总结

微博混合云DCP成果



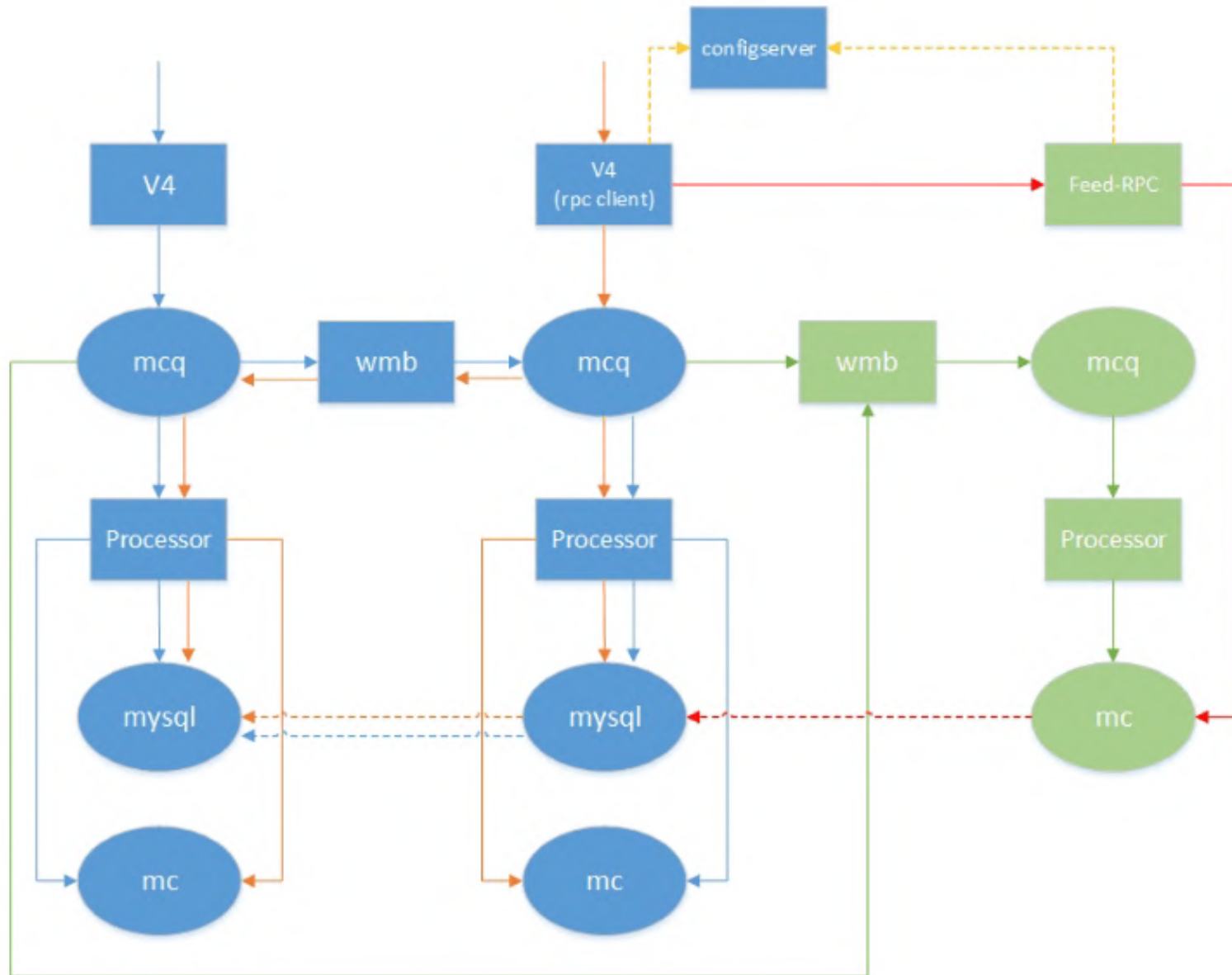
基础设施： Docker Registry、Ansible

初始化： Ansible、Registry依赖环境安装

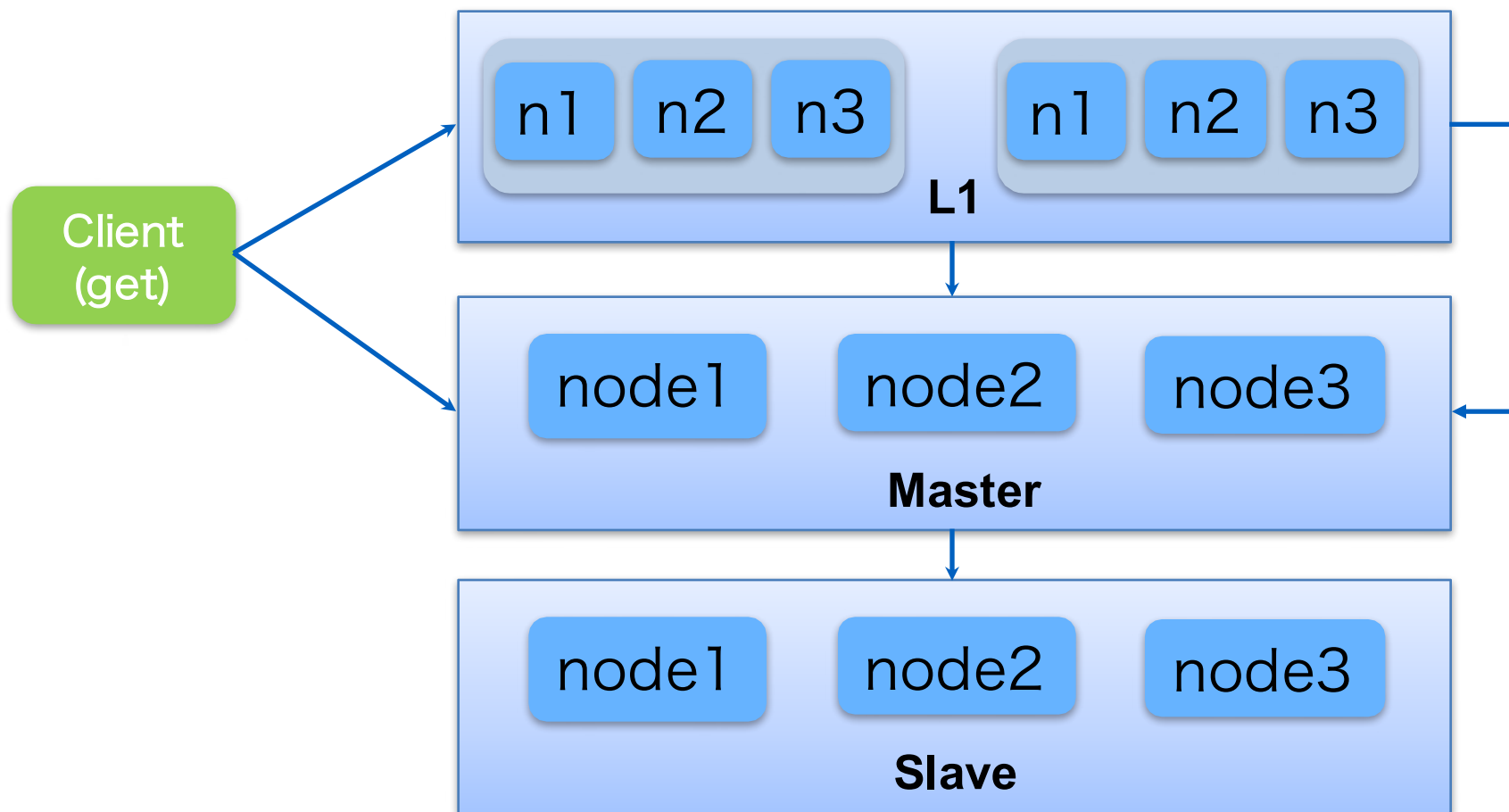
预热镜像： 镜像仓库预热

拉镜像： 单容器700M， 改造为公共容器镜像在VM镜像70M

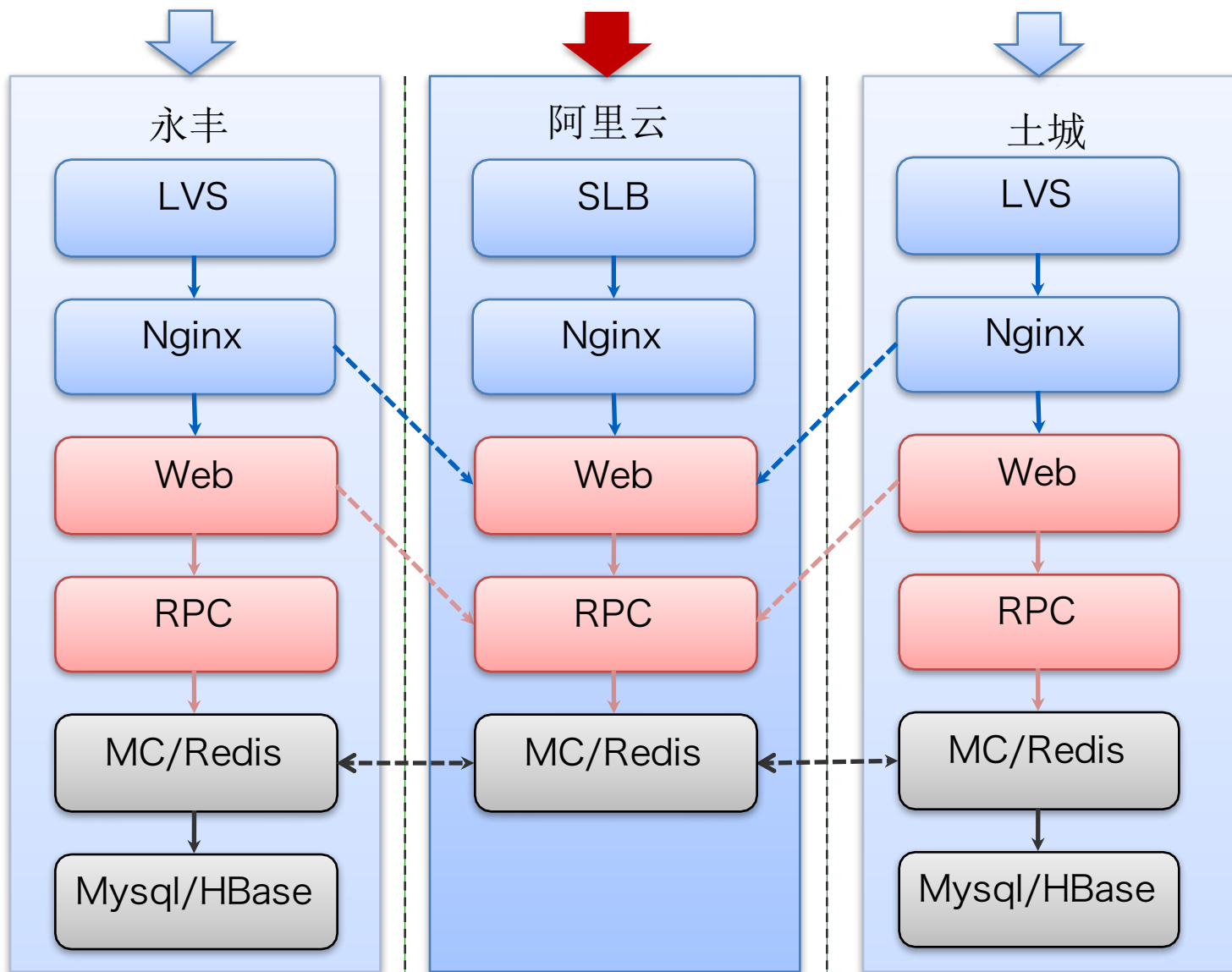
项目介绍：阿里云多机房部署



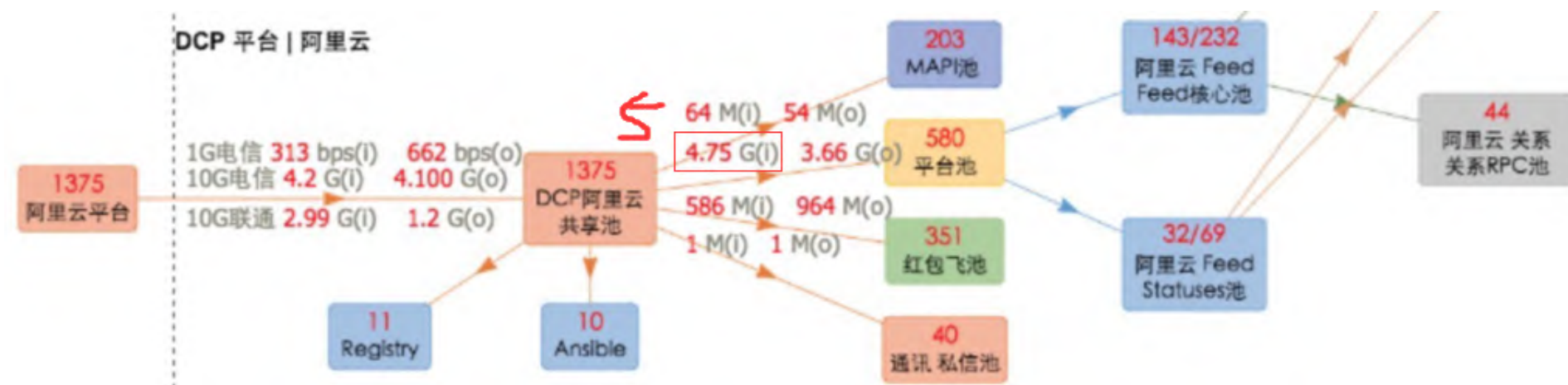
项目介绍：高可用、可扩展缓存集群



三节保障与阿里云部署



微博DCP-春晚作战图-各业务方



1-机房服务器个数

2-专线带宽

3-服务池容器水位

4-服务池容量承载倍数

5-各业务方占用带宽

总结 - 春晚问题与总结

问题	解决
阿里云部署缓存ECS PPS打满	扩容缓存MC L1一倍
Feed性能恶化，部分ECS（PPS、IOPS）性能差	单机监控发现性能恶化ECS，执行503
资源DNS解析失败	不使用SLB负载均衡，直接配置
带宽流量失衡，电信专线带宽达9.2G	1：拆分阿里云机房为yf_aliyun、tc_aliyun服务池，按照服务池分权重调整流量比例； 2：降级部分非主干消耗带宽回串内网功能
SLB负载均衡被打挂	IP直接配置resolv.conf

DCP(Docker Container Platform)开源项目



Thanks

以微博之力 让世界更美！

weibo.com