



华为企业云



云容器引擎

Kubernetes 在华为全球IT系统中的应用

华为PaaS开发部

王泽锋

华为内部企业IT系统

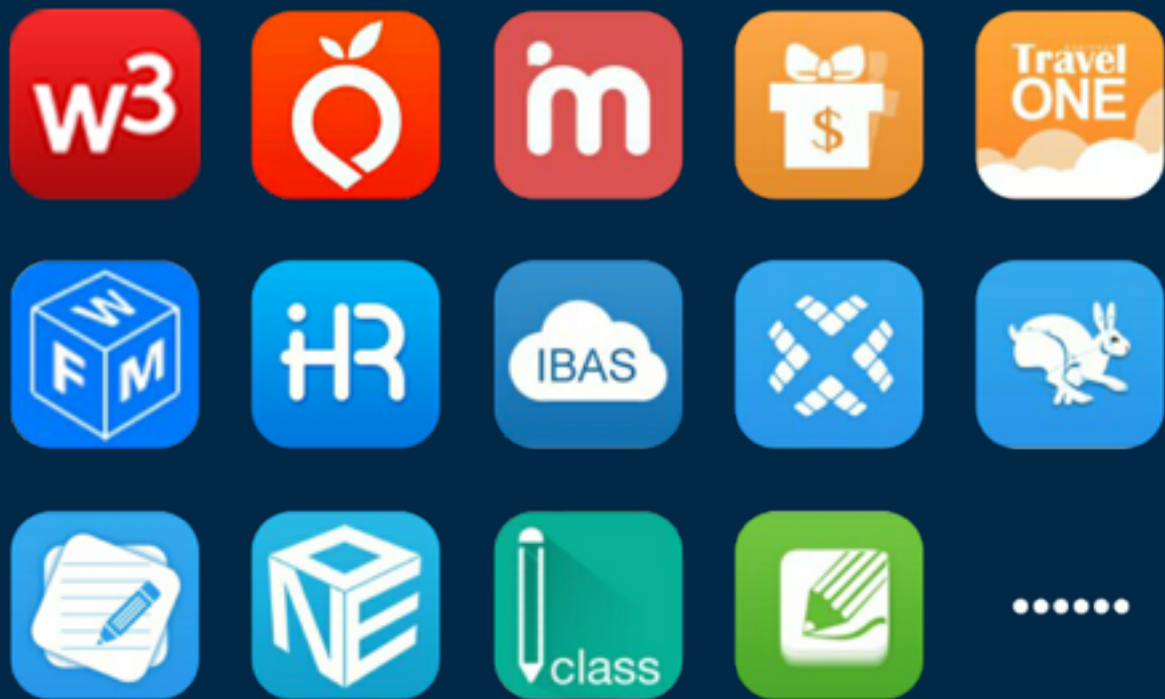
服务17万员工

业务遍及全球

8个数据中心

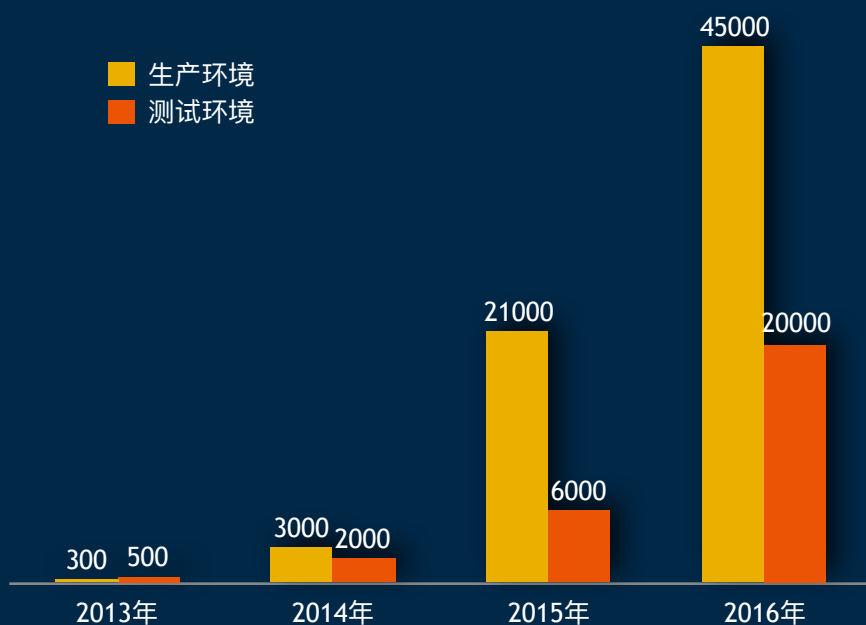


跑在华为IT系统上的业务



华为IT系统业务特点

MWC(中间件云)应用增长分析(以虚拟机计)



2015年统计数据

- 01 规模庞大 – 业务繁多，部署面广，覆盖全球多个地区
- 02 频繁发布 – 业务变更快，迭代周期短
- 03 冗余部署 – 同城双活、异地容灾、备份/滚动升级
- 04 容器化、微服务化，造成应用数量成倍增长

痛点与需求

01

资源利用率低，VM增长过快，成本激增

大量虚机应用，大规模部署，虚机OS消耗大量计算、存储资源；
虚机OS实际运维管理成本基本等同于物理服务器，虚机数量增加导致管理成本倍增

02

跨域多DC的部署维护无法拉通，运维投入巨大

缺少跨域多DC的应用统一部署和管理，依赖于大量的人工部署和维护

03

系统重载，应用伸缩周期长

对VM做伸缩，系统负载大，APP的VM环境无法快速制造，无法满足应用的大规模快速弹性部署需求

04

不支持快速迭代，业务上线效率低

不支持在线的应用环境申请和配置及切换，频繁发布、频繁升级背后的人力投入巨大；
缺少灰度发布、在线验证机制，难以快速定位修复线上问题。



华为企业云CCE – Cloud Container Engine



以 kubernetes、docker 为核心技术构建

- 面向企业市场提供：大规模集群管理、高性能并发部署、企业级用户安全等关键竞争力。
- 为企业应用提供：快速打包、秒级部署、自动化运维等全生命周期管理能力。



CCE在企业IT架构的目标与定位

- 跨域应用集中部署与管理
- 混合应用资源编排
- 开发测试生产一致性环境快速供给
- 开发性快速集成传统中间件
- 支撑传统应用中间件容器化



CCE在华为IT系统的进展

2015年底上线生产环境

- 实现自动化部署、提升资源利用率3-4倍；
端到端分钟级自助在线环境获取、自动弹性伸缩，大幅加速应用上线；
- 支持华为内部IT系统多项业务从传统SOA架构向容器为中心的微服务架构平滑演进。

TIME

SCALE

当前规模

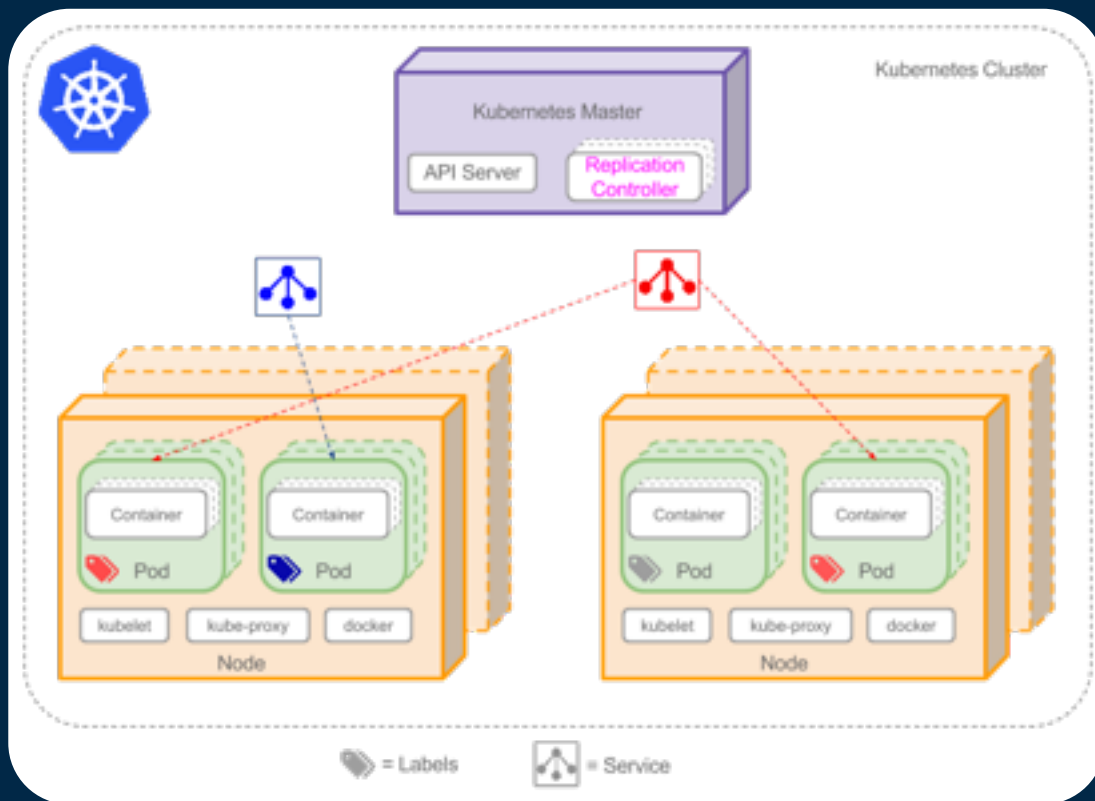
2k+ VM, 8k+ 容器

近期重点技术实践

- Kubernetes多集群联邦
- 应用间的亲和/反亲和调度

CASES

Kubernetes基本概念



鸟语	人话
Container	容器
Pod	容器组
Replication Controller (ReplicaSet)	副本控制器 (副本集)
Service	服务
Label	标签
Node	节点
Kubelet	节点Agent
Kubernetes Master	主节点

Kubernetes集群联邦

01

提供跨域应用集中部署与管理降低运维成本

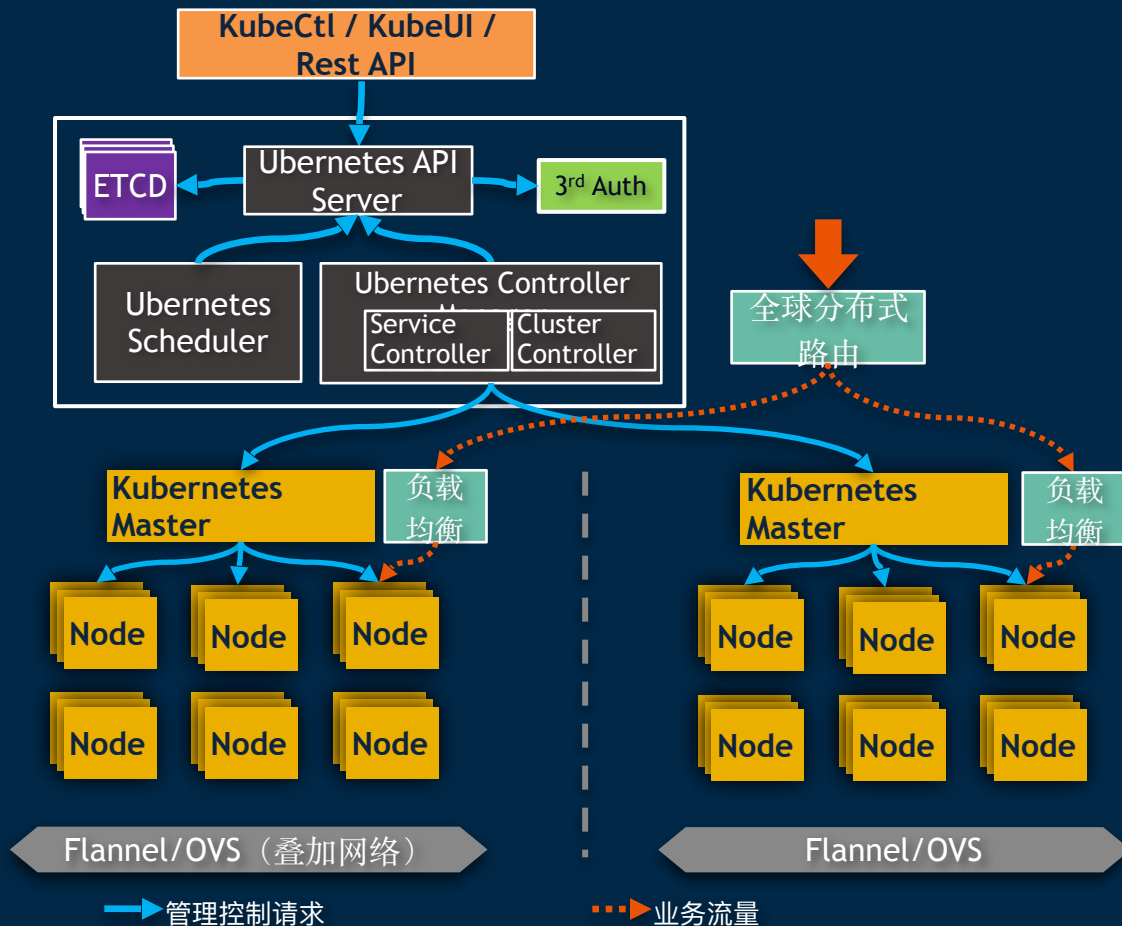
02

大规模部署的需求驱动

- 多数据中心统一管理部署应用，提供**跨域的应用服务发现**
- 跨域的网络限制与差异较大，难以通过k8s单集群支持
- 当前单集群的规模为**1k到2k**（k8s社区1.3版本规模为2k，年度目标为5k）
- 根据目前的应用增长速度，容器化之后整体规模需要支持**3w虚拟机，10w容器**
 - 1) 按虚拟机应用算年底VM数量将超过6万
 - 2) 应用/实例的部署比例平均在1:2到1:3之间
 - 3) 容器化1个应用平均拆分出4~5个微服务

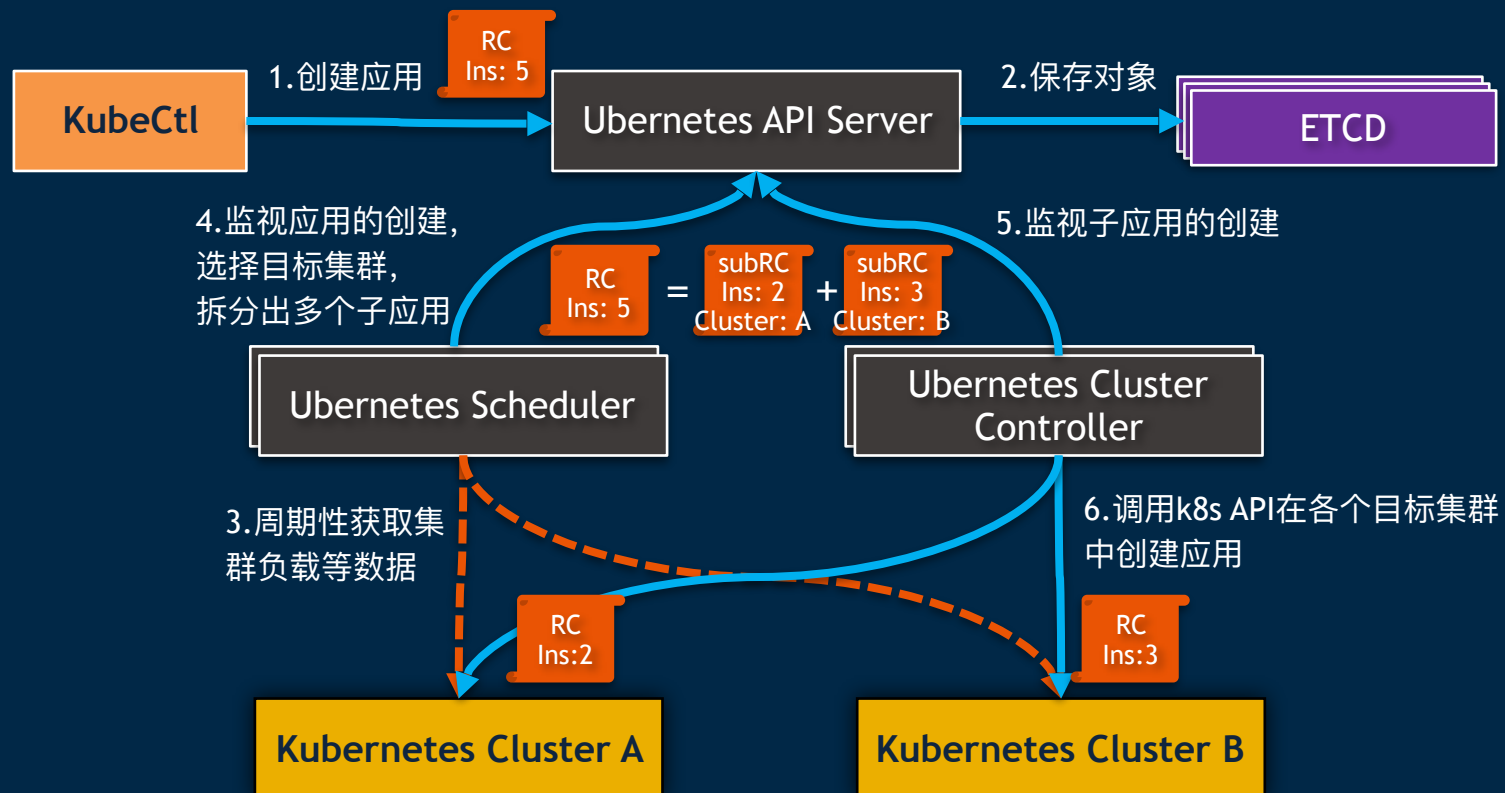


多集群联邦架构



- 01 **Ubernetes Api server** – 增加集群相关API，屏蔽集群差异，统一请求入口
- 02 **Ubernetes Scheduler** – 分拆联邦级别对象到集群
- 03 **Cluster controller** – 管理个集群状态，集群级别对象创建
- 04 **Service controller** – 跨集群服务发现

多集群联邦下的应用创建

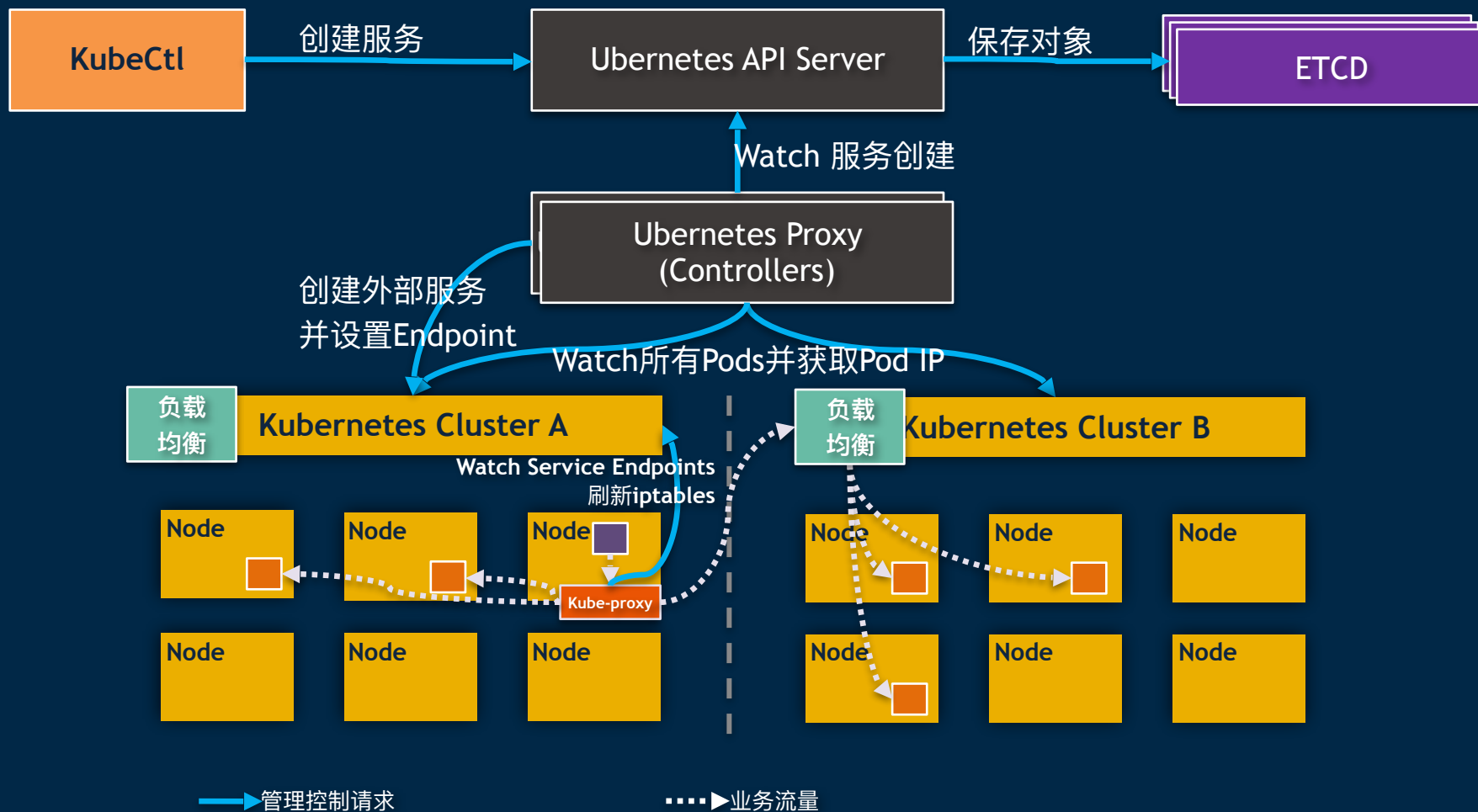


RC: Replication Controller 多实例无状态应用

subRC: 子应用

Ins: 实例数

多集群联邦下的服务发现



应用间的亲和/反亲和调度

A

亲和性

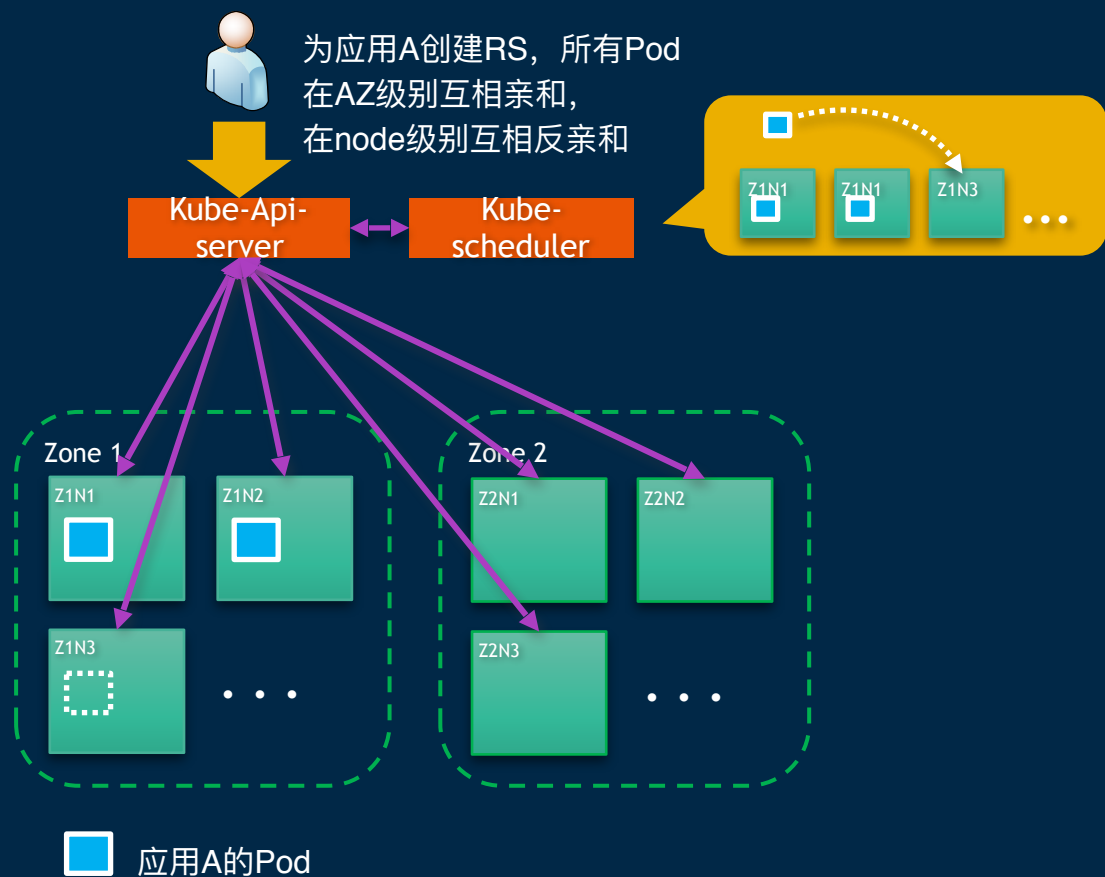
传统应用进行容器化，拆分微服务之后的部署约束，需要按实例逐一配对**就近部署**，容器间通信**就近路由**，减少网络消耗。

B

反亲和性

高可靠性考虑，同个应用的多个实例反亲和部署，**减少宕机影响**
互相干扰的应用反亲和部署，避免干扰。

应用间的亲和/反亲和调度

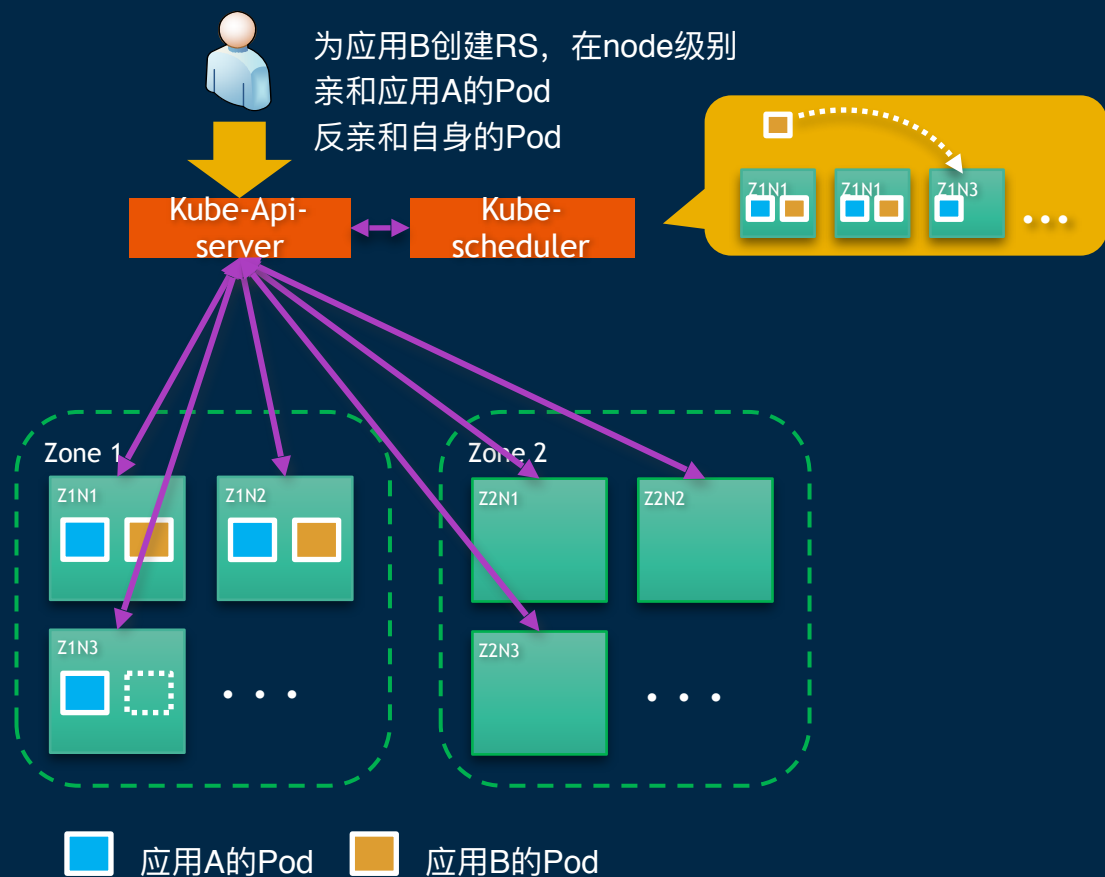


支持不同级别的亲和性

根据node按labels键值对
“动态分组”

在这个例子中：计算亲和性时，所有az=zone1的node为一组，az=zone2的node为另一组。计算反亲和性时，所有node各为独立的组。

应用间的亲和/反亲和调度



支持硬性亲和/反亲和

调度时**过滤**, 符合的node保留,
不符合的node直接排除

支持软性亲和/反亲和

调度时**评分**, 符合的node高分,
不符合的node低分

鸡生蛋蛋生鸡 – 亲和性调度的对称性考虑



应用B亲和A，应用A后于B创建怎么办？

对称性设计，调度B时会检查B是否被别人亲和/反亲和



被亲和的应用跑了（比如挂了异地重建）怎么办？

Pod挂掉通常kubelet会直接重新拉起。只有node挂掉时，node上所有的pod被终结，RC/RS重新创建Pod，才会有跑掉的情况。

鸡生蛋蛋生鸡 – 亲和性调度的对称性考虑

01

应用间亲和性 (pod affinity) 的对称性 – 不完全对称

检查待部署的pod亲和哪些已存在pod, 同时检查该pod被哪些已存在pod亲和
hard pod affinity: 不对称

正向的亲和为hard

反向的亲和为soft (不能因为没有被其他pod硬亲和就阻塞调度)

soft pod affinity: 对称

正向反向都为soft

02

应用间反亲和性 (pod anti-affinity) 的对称性 – 完全对称

检查待部署的pod反亲和哪些已存在pod, 同时检查该pod被哪些已存在pod反亲和。

对称性之后 – 未来可用的措施

01

限制异地重启 (forgiveness)

Node挂掉时, pod不被终结, 等待node恢复时由kubelet原地重启pod

02

运行时迁移 (rescheduling)

Rescheduler周期性检查集群中pod的亲/反亲和性规则, 并迁移调整

后续考虑投入方向



规模 & 性能

前期的规模性能分析主要在控制面，后面将投入数据面分析。

网络规模和性能优化，以支撑10万容器的需求。



调度与重调度

Dedicated Node
Forgiveness(限制异地重启)
Rescheduling(运行时重调度)



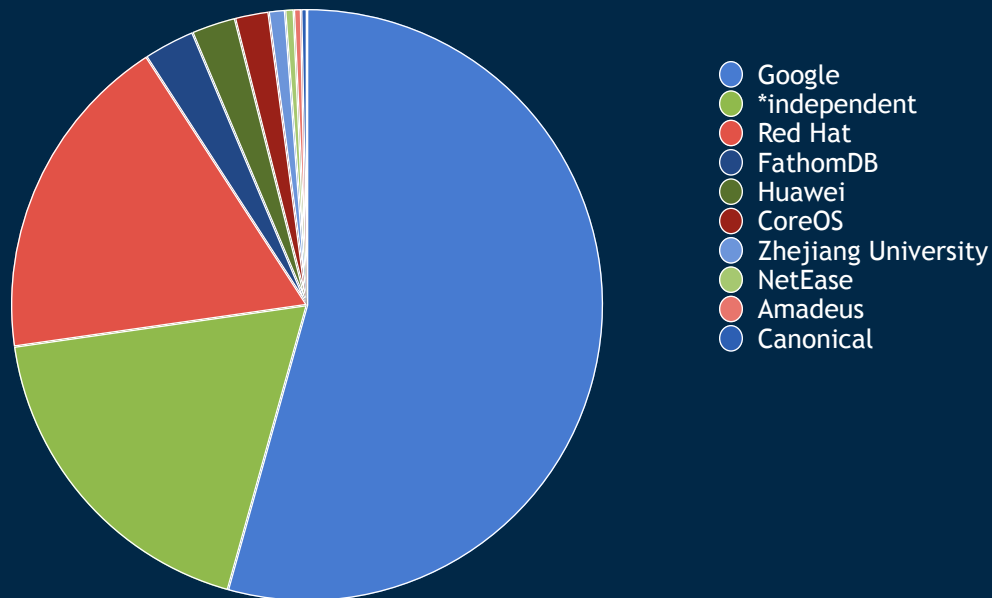
集群联邦

当前社区仅贡献了phase 1，尚不完整，需要持续投入。

Kubernetes开源社区贡献

commits 400+, 全球排名第四, 国内第一

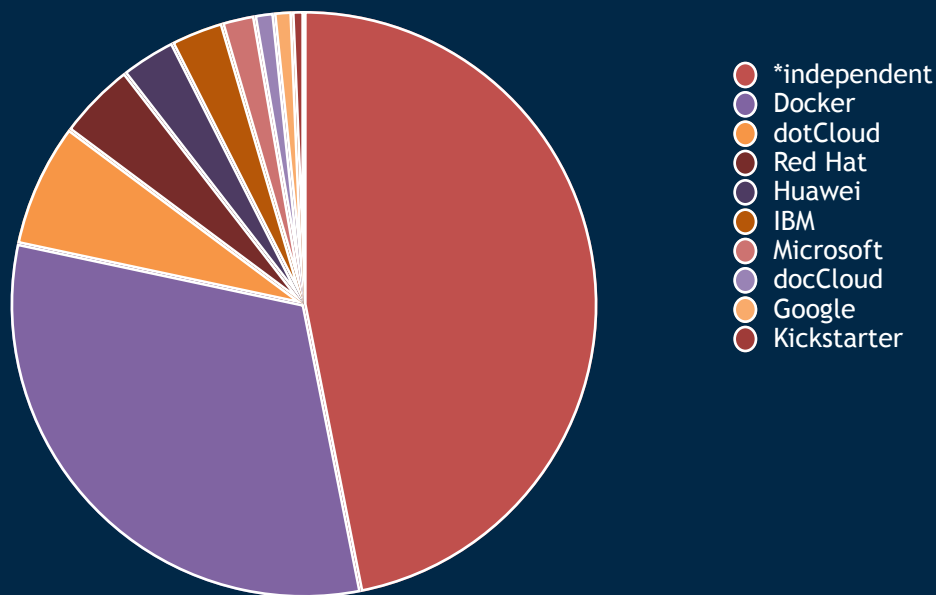
- CentOS k8s集群管理
- k8s运维系统对接kafka
- k8s运维系统对接elastic search
- Heapster重构及sink扩展
- 主导设计Ubernetes集群联邦
- Node Affinity
- Pod Affinity/Anti-Affinity
- Taints-tolerations
- 其他Bug和修复.....



Docker开源社区贡献

commits 800+, 全球排名第四, 国内第一

- 增加关闭容器oom功能
- 容器重启策略增强
- 日志格式优化, 更便于解析
- 环境依赖检测功能增强
- 增加容器cpu带宽限制
- 增加容器IO带宽限制
- 增加内存节点限制
- 增加内核内存限制
- 增加内存预留机制
- 增加swap内存限制
- Docker exec增强, 增加指定用户和特权用户执行exec
- Docker build资源限制增强
- 增加ARM64支持
- 安全加固, 增加seccomp支持
- 增加在容器内获取cgroup信息功能



后续技术交流 - 华为CCE交流4群



后续技术交流 - 华为CCE交流5群



云容器引擎即将公测 — 邀您体验



平台共建·市场合作·产品咨询

Thank you all for watching

谢谢大家!

