



# Ray融合计算在腾讯TEG推理方向的 落地实践

宋顾杨

腾讯TEG-Ray引擎负责人

2025/12/20



## 宋顾杨 grayson 久龙

腾讯TEG-Ray引擎负责人

个人简介：前蚂蚁Ray开源负责人，2017年开始接触Ray，国内最早的Ray开发者之一，Ray开源社区Committer，Ray中文社区布道师；经历过Ray在蚂蚁和腾讯两家大厂从0到百万核规模的落地

## CONTENT 目录

- 01 从Ray Summit看发展趋势
- 02 云原生架构演进
- 03 离线推理落地实践
- 04 在线推理落地实践
- 05 未来展望

# 从Ray Summit看发展趋势

Ray的历史趋势——大模型催生Ray站稳主流生态位



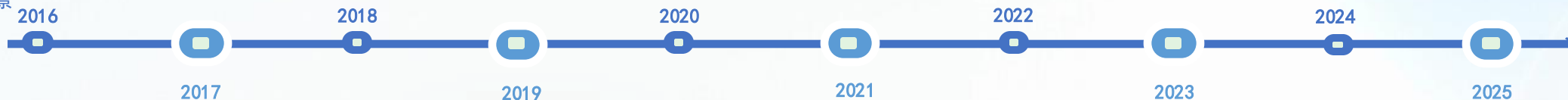
- Ray开源
- 面向传统**强化学习**场景

- Ray论文发表

- Ray 1.0发布, For Scalable AI

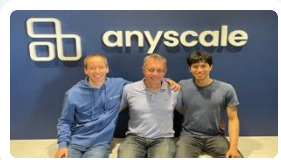
- Ray 2.0发布, Ray AI Runtime 框架
- 字节跳动、华为等公司开始大规模使用Ray

- Ray在国内加速发展中...
- **Ray成为Data+AI的主流解决方案**



- 国内蚂蚁公司与riselab合作

- Anyscale成立
- 后期融资\$260M



- Ray的开源stars超过flink, 成为发展最快的计算引擎

- **OpenAI基于Ray训练GPT4**
- Ray在蚂蚁生产规模超100W核

- 基于**强化学习**的post training成为主流,
- **Ray成为RL框架首选分布式底座**
- Ray加入Pytorch基金会

Ray  
周下  
载量  
趋势

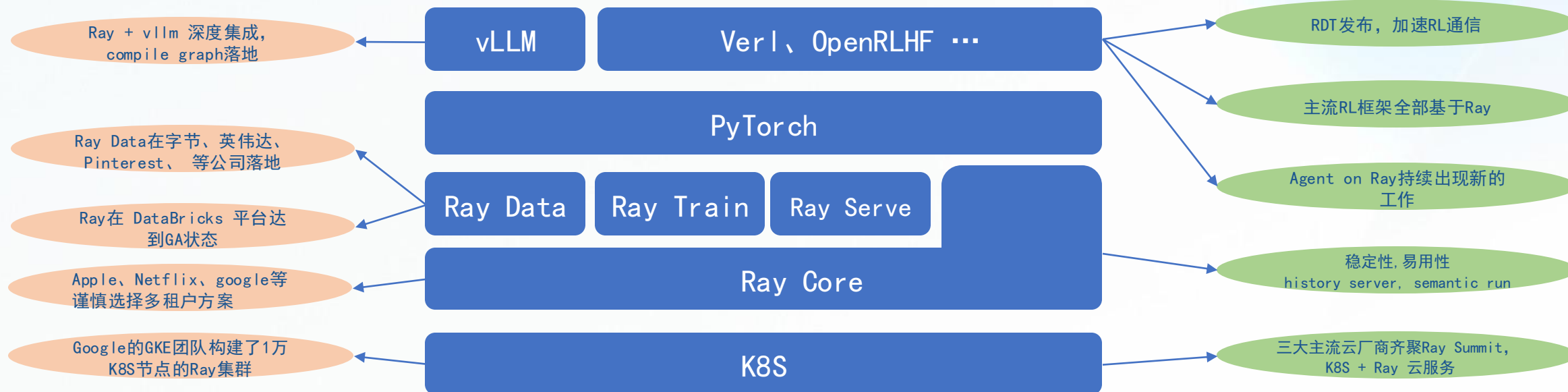


# 从Ray Summit看发展趋势



2024

2025



2024和2025共同趋势:

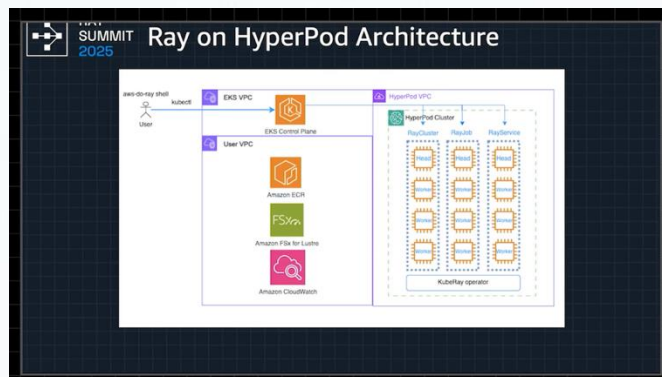
- 趋势1: Ray在Data + AI方向的应用逐渐成为主流
- 趋势2: Ray + K8S成为行业共识
- 趋势3: 多租户任重而道远, Workflow仍是主流
- 趋势4: Ray在AI生态的集成正逐渐深入

2025新趋势:

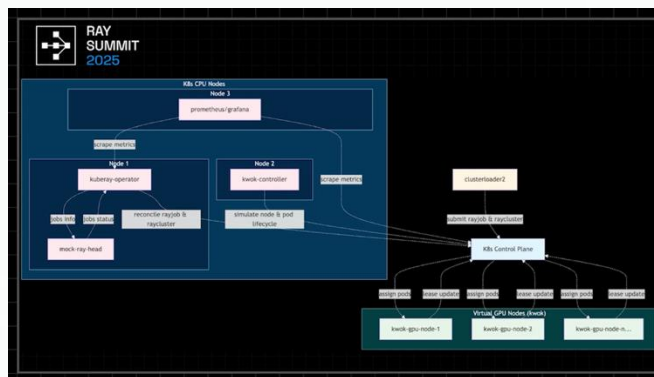
- 趋势1: Ray成为RL框架首选分布式底座
- 趋势2: Ray本身更注重稳定性和易用性

# 从Ray Summit看发展趋势

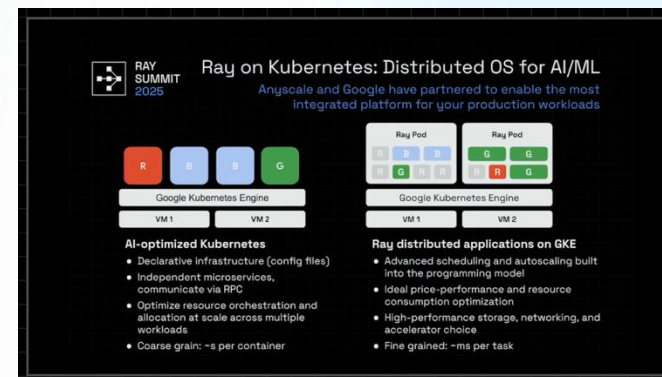
K8S + Ray 正成为 AI Infra 的基石



AWS



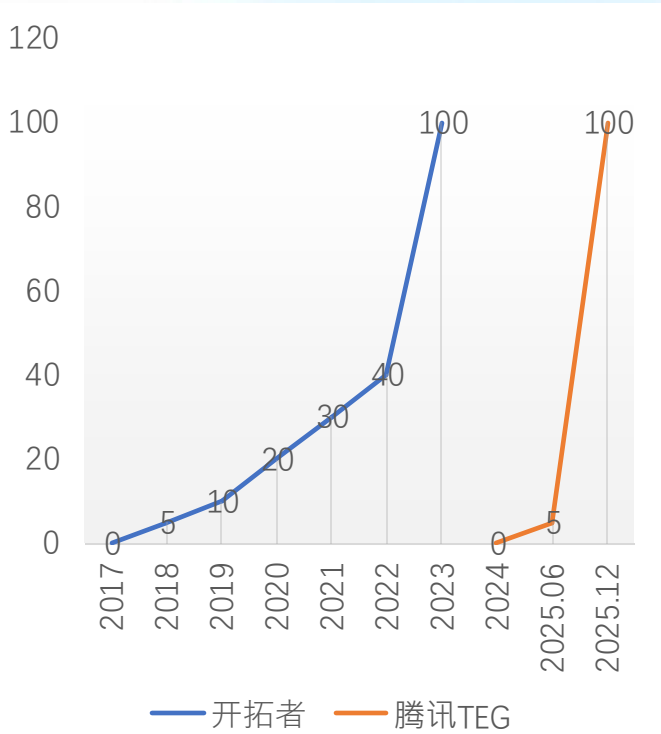
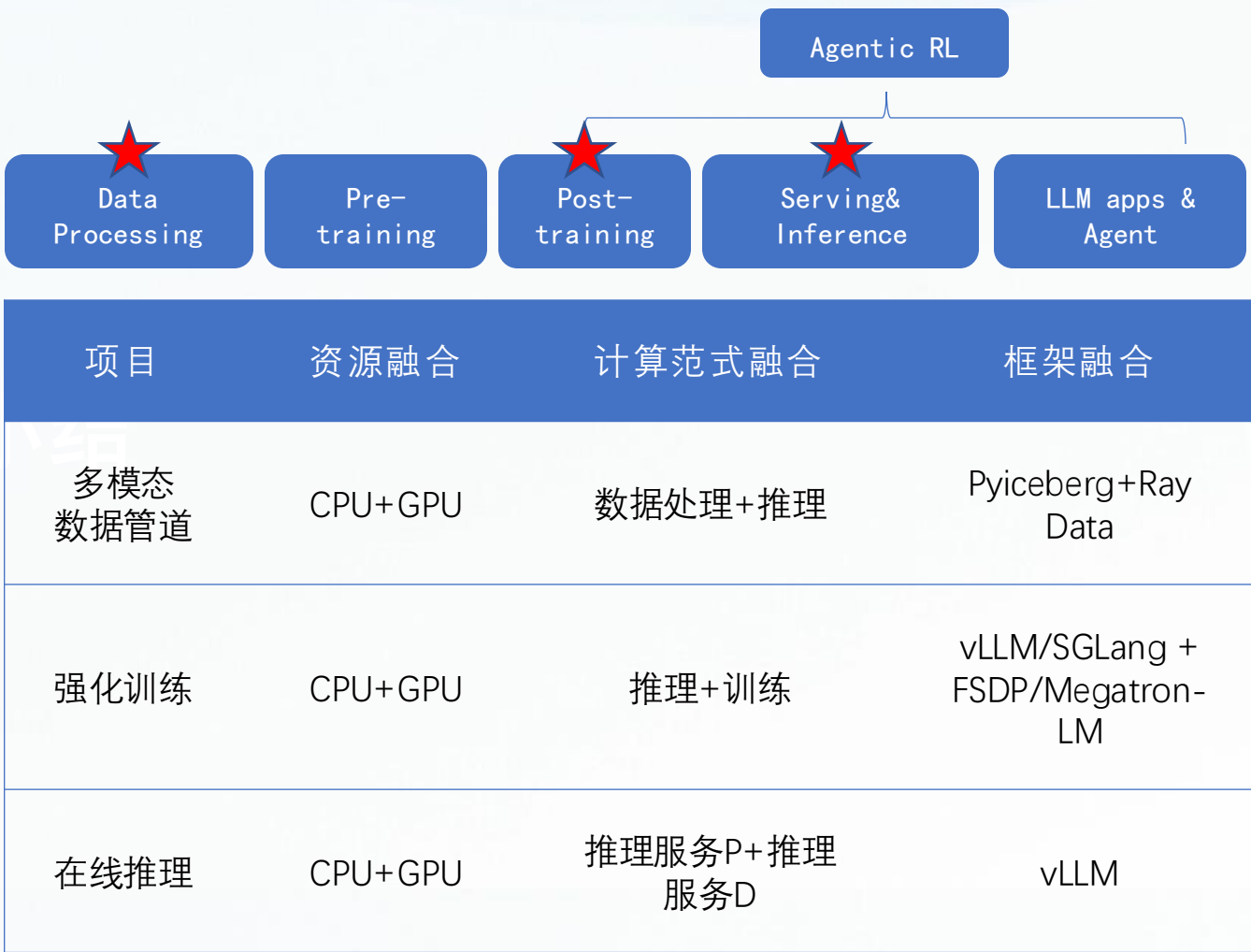
Microsoft Azure



Google GKE

国外三大主流云厂商齐聚Ray Summit，探讨 K8S + Ray 云服务

# Ray融合计算在腾讯TEG的落地场景



100w核 业务发展曲线

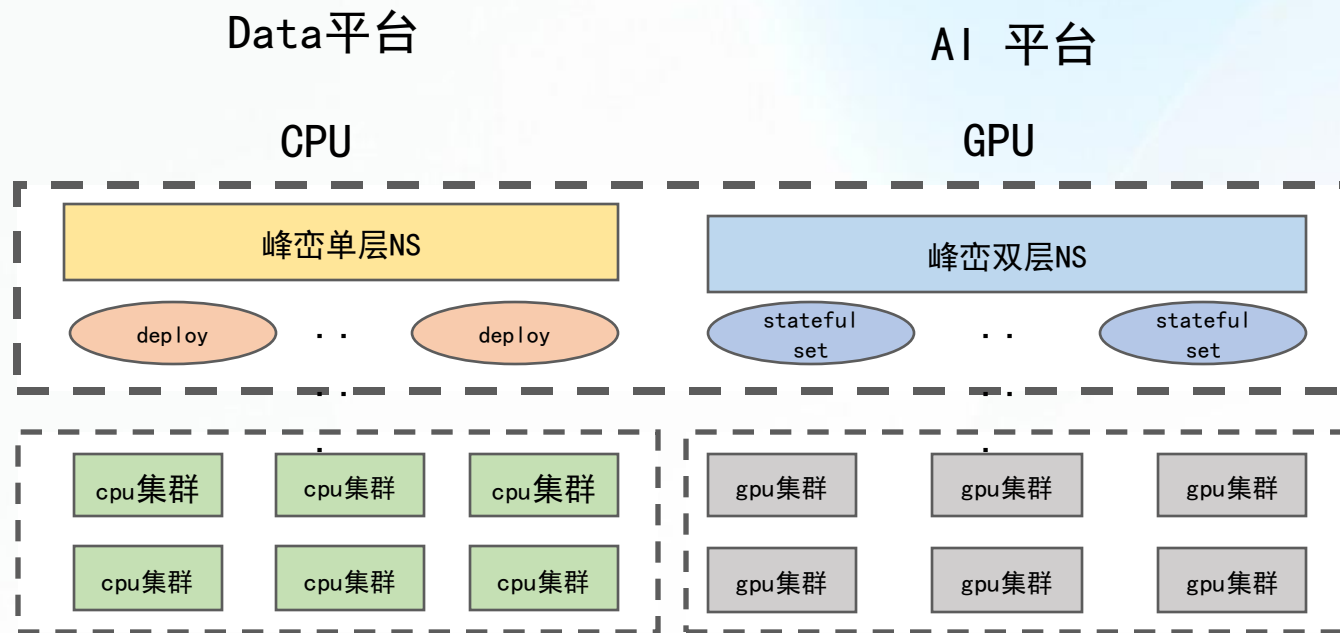
# 云原生架构演进

## Ray大规模部署面临的问题



### 腾讯内云原生环境特点

- 大规模云原生基础设施通常存在联邦架构
- 生产环境存在上百个 K8S 物理集群
- CPU算力 和 GPU算力的物理集群隔离
- Data平台和AI平台底层两套基础设施

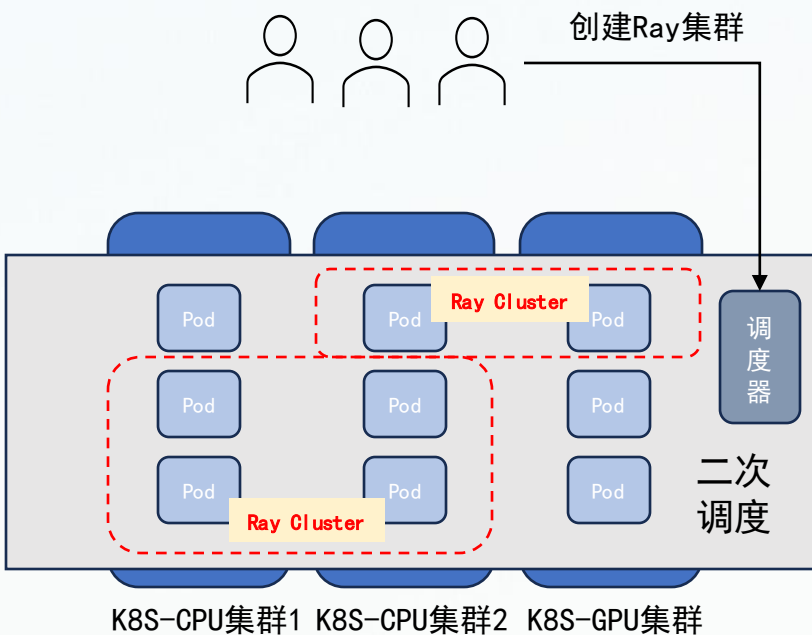


### Ray落地挑战：融合计算的范式与算力基础设施的矛盾

- “CPU + GPU” 统一调度难
- “数据处理 + 推理” 融合计算难
- “Data + AI” 统一平台难

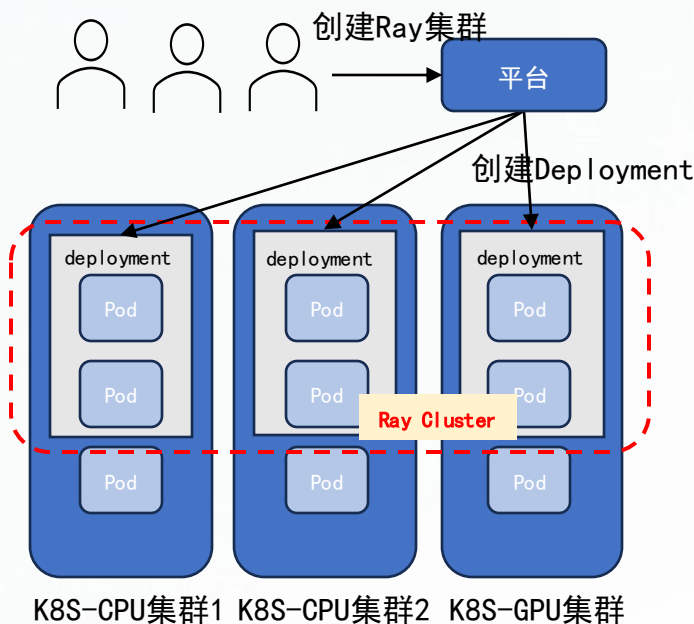
# 云原生架构演进

核心需求是要实现Ray跨K8S集群的联邦调度



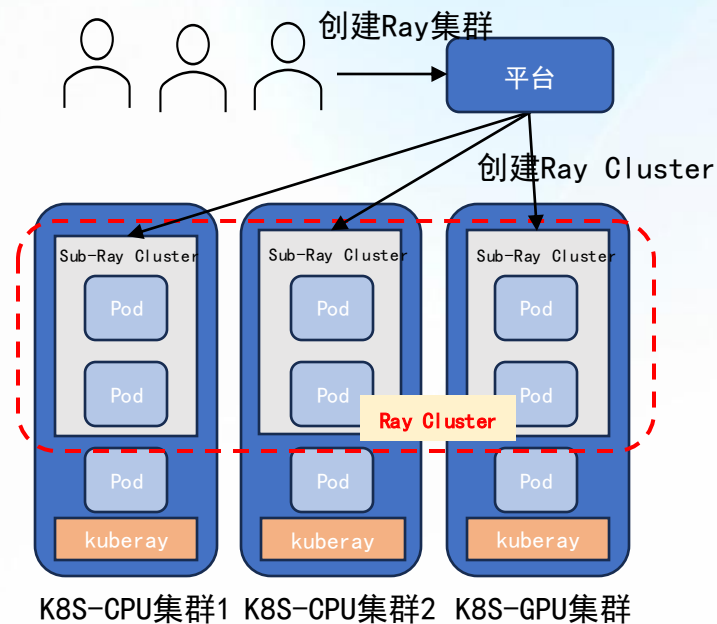
二层调度方案

放弃kubera y, 形成了“K8S + 中间层 + Ray”三层调度系统, 复杂度增加



standalone组网方案

放弃kubera y



kubera y联邦

继承原生kubera y的Cluster/job管理、autoscaling、history server管理等能力

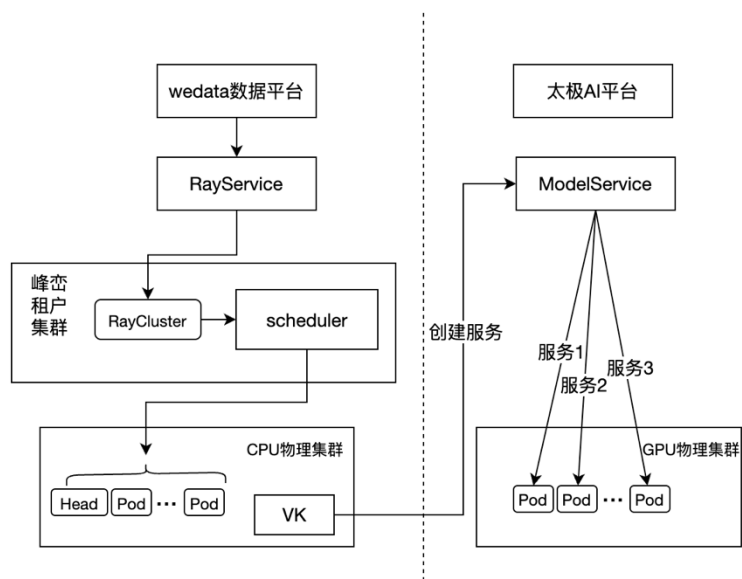


# 云原生架构演进

从vk到原生kuber ay联邦

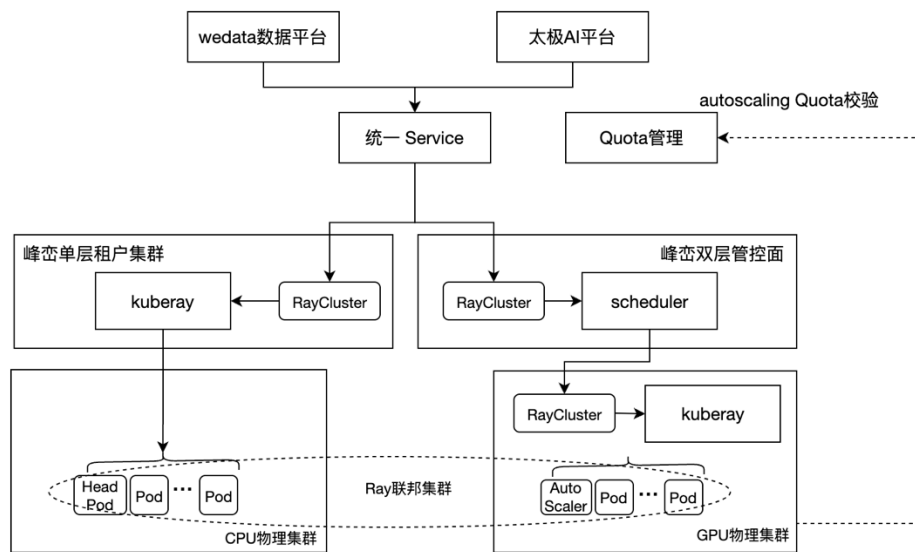


## 基于virtual kubelet的架构



- Data Infra、AI Infra分离架构
- 最大瓶颈：一个Pod对应一个AI服务
- 最大支持2k卡

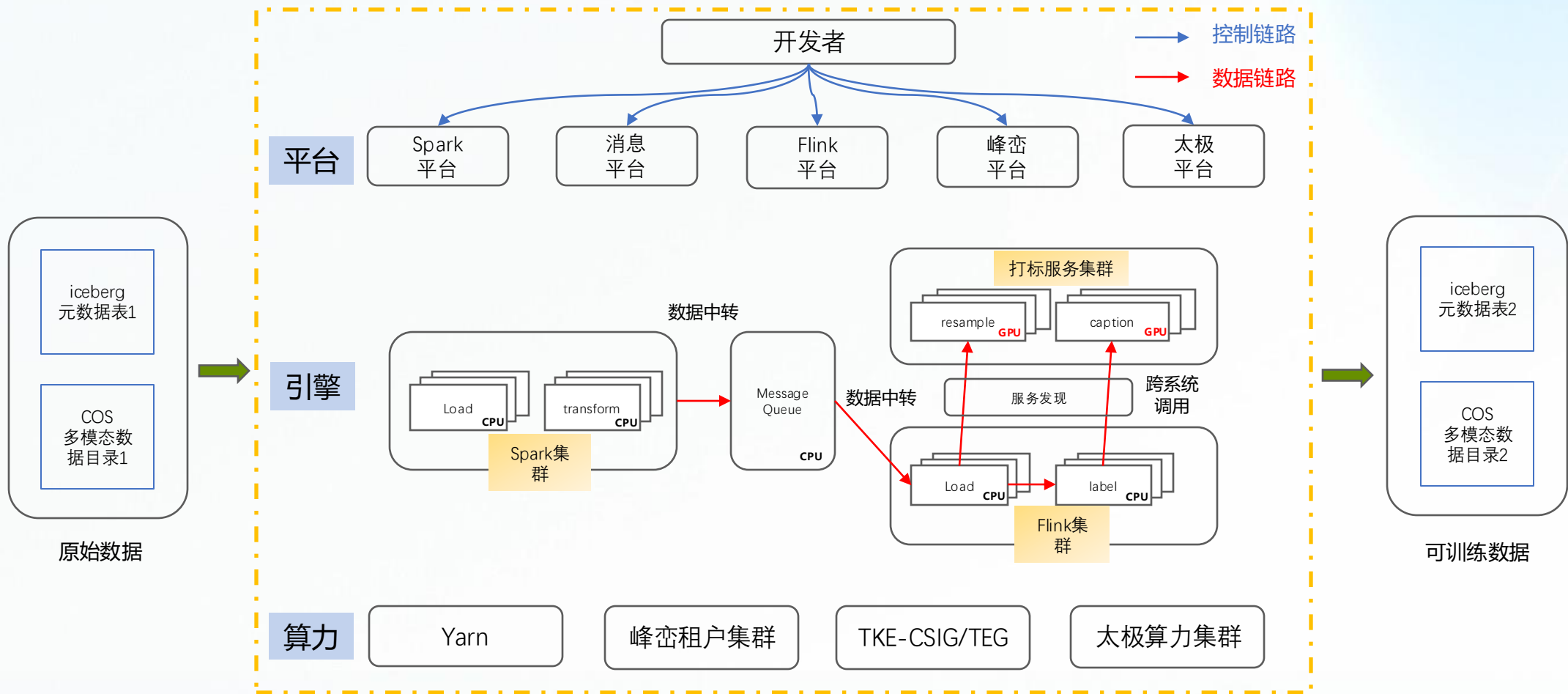
## Kuberay联邦架构



- Data + AI Infra统一架构
- 支持1w卡以上

# 离线推理落地实践

混元数据管道上一代架构

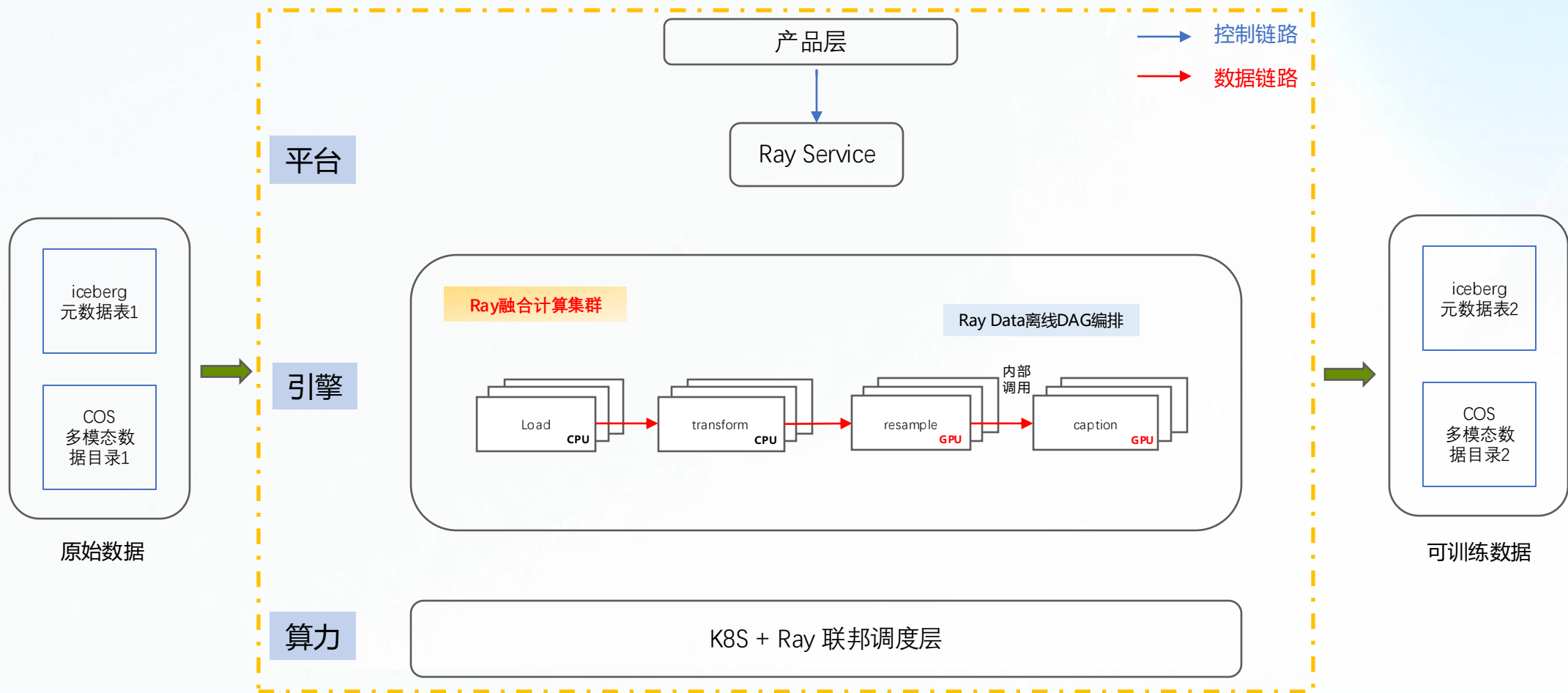


**问题** 1、平台多：控制链路复杂 2、引擎多：数据链路低效 3、算力集群多：资源利用率低



# 离线推理落地实践

基于Ray新一代混元数据管道



- 1、**单平台**: 架构简单, 好迁移    2、**单引擎**: 研发简单, 运行高效    3、**统一调度**: 灵活分配, 资源利用率高

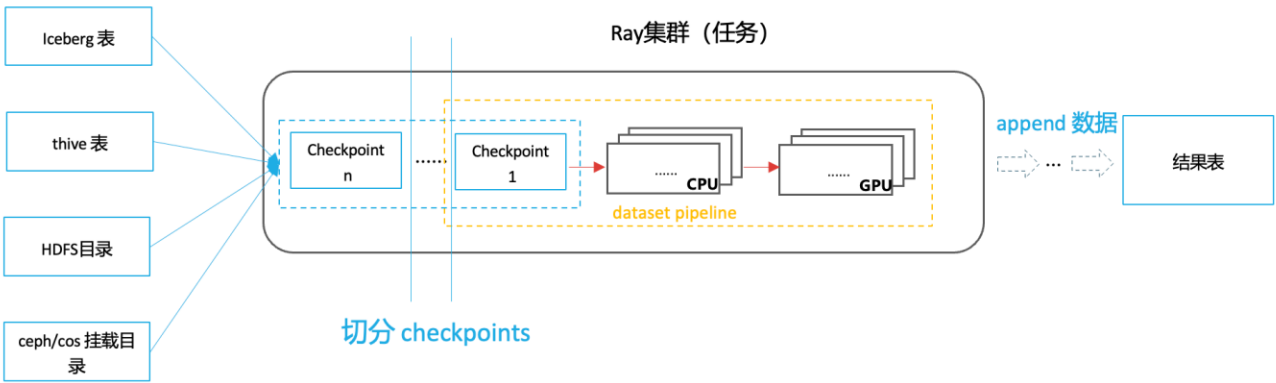
# 离线推理落地实践

容错“三部曲”—— 前两部



## 第一部：基础容错

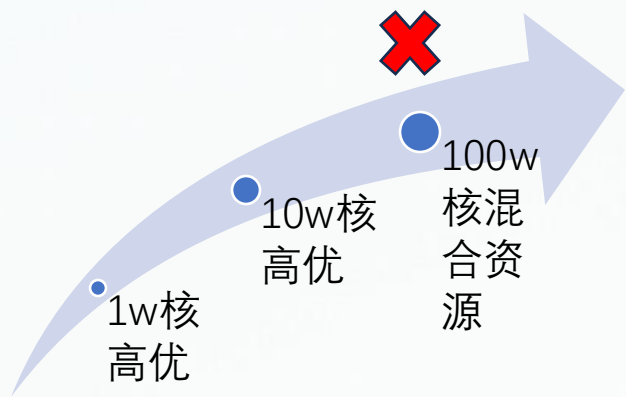
分类	社区版能力	内部版增强能力
Ray Core系统组件	Head节点高可用，worker节点故障替换 Task重试，Actor重调度 Object血缘回溯	Task/Actor调度超时机制 Object血缘回溯超时机制 节点故障动态感知
Ray Data框架层	依靠Ray Core层容错	Actor黑名单机制 算子执行超时机制



## 第二部：checkpointing

# 离线推理落地实践

容错“三部曲”—— 规模化必须面临多种不稳定资源



百万核规模落地中引入的不稳定资源

- 弹性GPU卡

随机分配和回收  
作业长时间拿不到卡

- 混部GPU卡

算力压制  
显存压制

- 混部CPU

GPU机器上挖掘的CPU  
Disk限制，进程线程限制

- 低优CPU

CPU压制严重  
磁盘io不稳定

# 离线推理落地实践

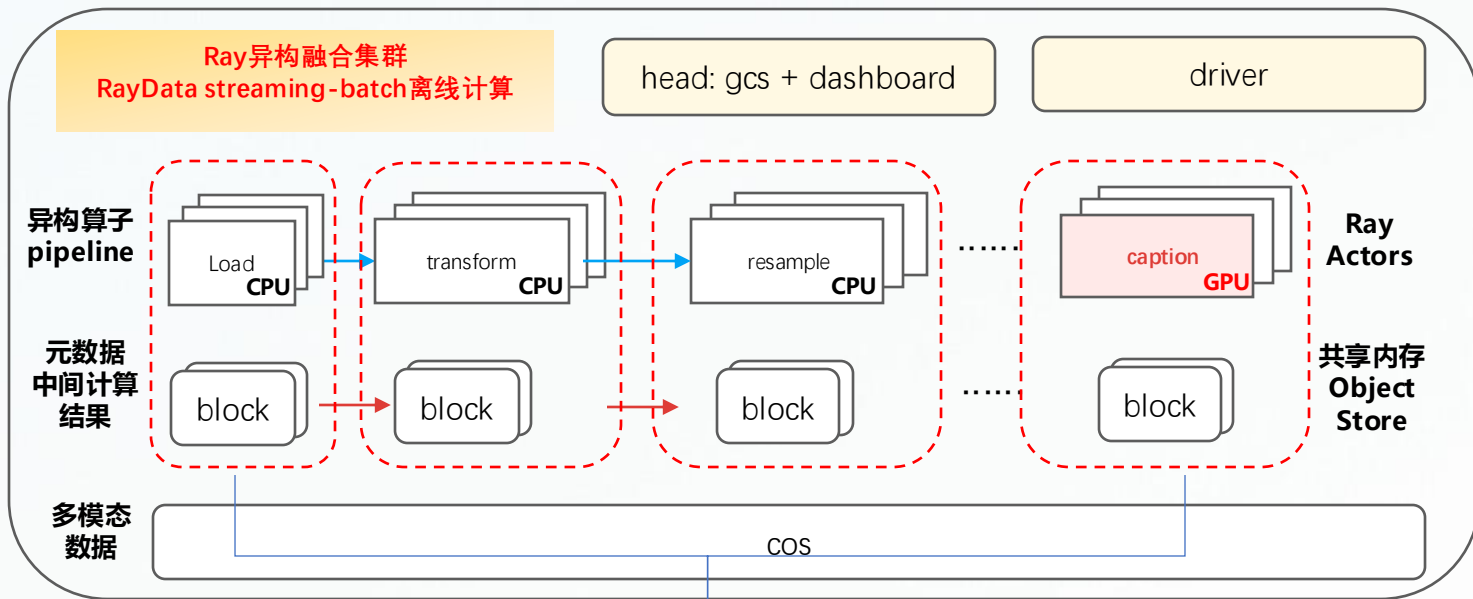
容错“三部曲”——为什么需要架构重构



蚂蚁开源  
ANT OPEN SOURCE



RAY



Dataset / Operator Name	Progress	State	Rows Outputted	Memory Usage (current / max)	Bytes Spilled	Logical CPU Cores (current / max)	Logical GPU Cores (current / max)
dataset_25	3016 / 10000	RUNNING	639042	13.56GB/15.22GB	0.0000B	5566.05/5676.4	721/721
Input0	1 / 1	RUNNING	0	0.0000B/0.0000B	0.0000B	0/0	0/0
Repartition1	10000 / 10000	RUNNING	0	0.0000B/0.0000B	0.0000B	0/0	0/0
MapBatches1	10000 / 10000	RUNNING	60000	760.18MB/3.74GB	0.0000B	0/0	0/0
MapBatches2	8907 / 10000	RUNNING	53184	1.44GB/2.28GB	0.0000B	107.80000000000001/271.40000000000003	0/0
MapBatches3	7600 / 10000	RUNNING	45342	1.08GB/1.36GB	0.0000B	230.75/230.75	0/0
MapBatches4	7230 / 10000	RUNNING	43230	1.99GB/1.99GB	0.0000B	0/0	360/360
MapBatches5	5558 / 10000	RUNNING	33156	1.65GB/1.66GB	0.0000B	480/480	0/0
MapBatches6	4565 / 10000	RUNNING	27270	677.43MB/706.47MB	0.0000B	0/0	41/41
MapBatches7	4550 / 10000	RUNNING	27138	568.42MB/664.04MB	0.0000B	0/0	64/64
MapBatches8	4481 / 10000	RUNNING	26748	308.34MB/317.56MB	0.0000B	0/0	32/32
MapBatches9	4467 / 10000	RUNNING	26634	282.32MB/330.96MB	0.0000B	0/0	32/32

复杂任务超过20个算子



在存算一体架构下，血缘回溯带来的**重算成本不可接受**，这是AI计算和大数据计算的本质区别

低劣资源GPU量波动大 **Tencent**

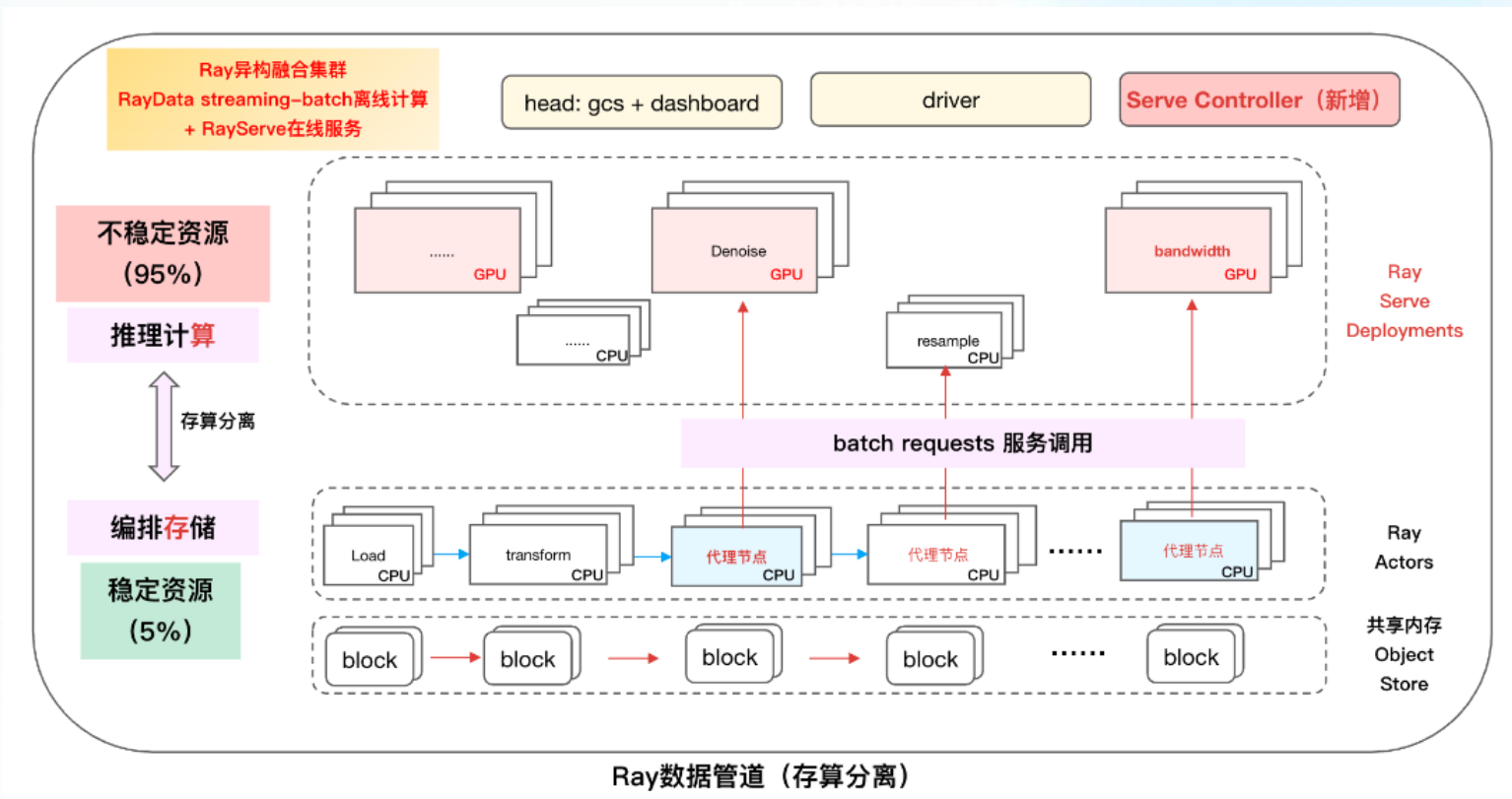
# 离线推理落地实践

## 容错“三部曲”—— 第三部 存算分离



实现存算分离的三种方法：

- object冗余备份（已实现）
- 引入Ray Serve（已实现）
- object多级缓存（未实现）



基于Ray Serve的存算分离方案

# 离线推理落地实践

Ray Data的尽头是Ray Serve?

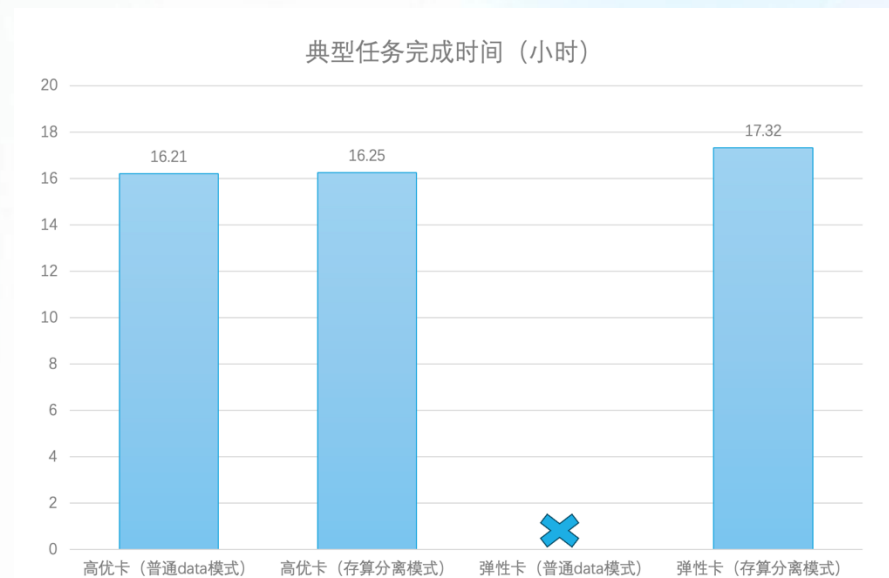


## 引入Ray Serve的收益

- 存算分离
- 容忍更大的block size, 减小driver压力
- checkpoint之间可以复用deployments

## 引入Ray Serve带来的问题

- 从两个调度器变成三个调度器 (GCS、Driver、Serve Controller)
- Data和Serve用户接口不统一 (可以通过封装解决)



经过优化后的Data + Serve架构性能 No Regression

# 离线推理落地实践

## 利用率优化



### ● 冷启动优化

eager化setup环境

合并序列化

指数扩缩容

### ● 计算Bubble消除

Autoscaling优化

Job之间集群常驻复用

Checkpoint之间进程池复用

### ● 长尾问题治理

算子实例timeout机制

不健康算子实例自动检测

缩容效率优化

### ● 算子计算效率提升

兼容 triton server

复合算子拆分

多算子资源碎片优化

### ● 任务参数自动优化

算子基线自动压测

任务基线自动测算

资源智能调度

# 离线推理落地实践

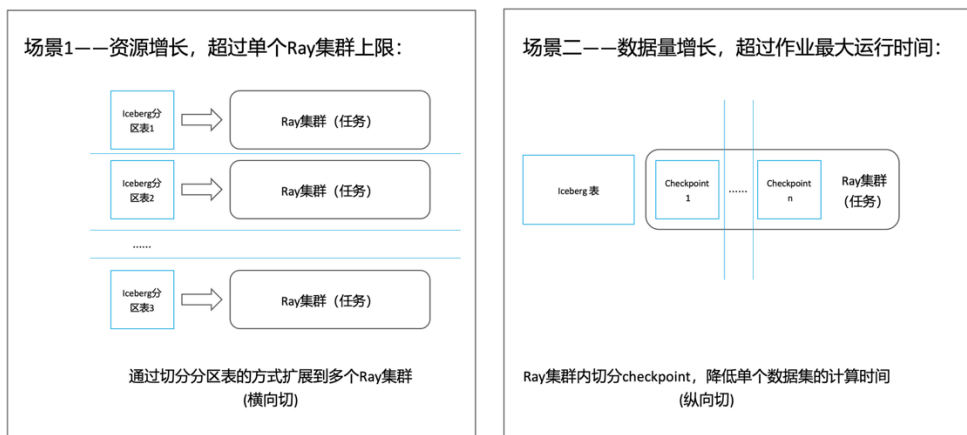
规模化和收益



## 安全基线:

安全基线	6月情况	8月底	当前
Ray Cluster	500节点	1k节点 2k卡	2k节点 1w卡
Actor规模	1w	2w	4w
blocks数量	2w	4w	12w
算子复杂度	5个算子	20个算子	20个算子

## 扩展方式:



## 业务收益:

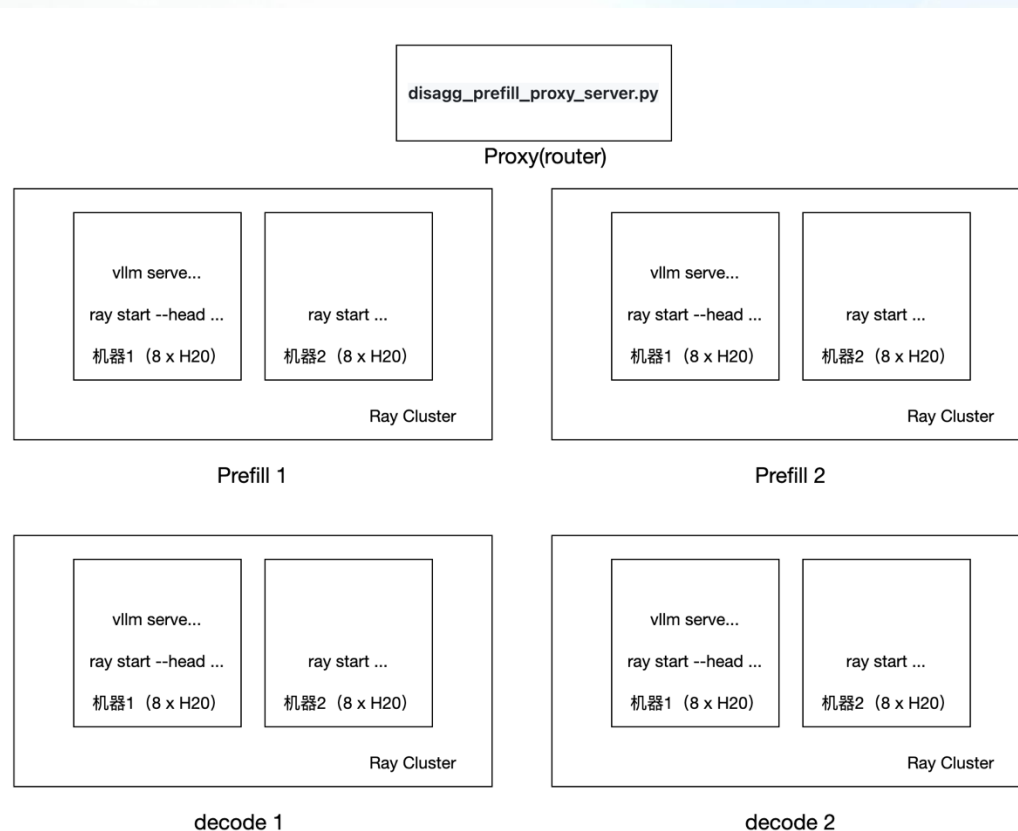
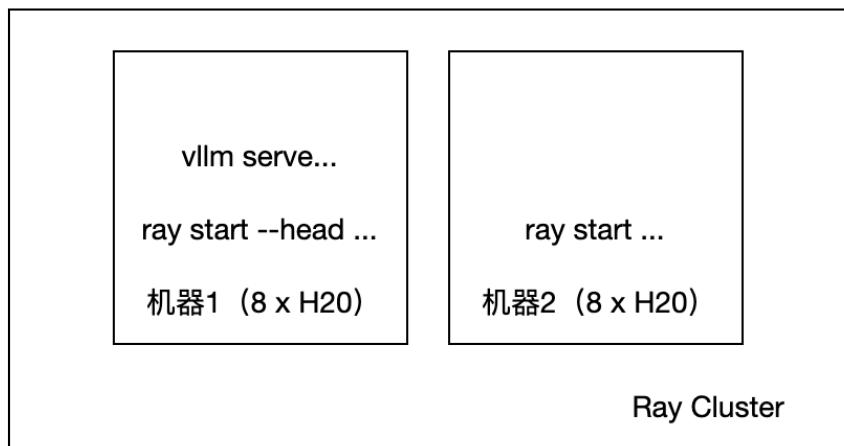
- 部分离线推理任务，通过CPU+GPU混合调度，**节省90% GPU卡**
- 部分离线推理任务，通过融合计算优化，**GPU利用率提升3倍以上**
- 部分离线推理任务，**推理成功率40%**
- 新任务研发到上线，**人效提升1-3倍**

# 在线推理落地实践

大模型推理PD分离架构带来的分布式新场景



- 如何 deploy?
- 如何 service discovery?
- 如何 failover?
- 如何 autoscaling?



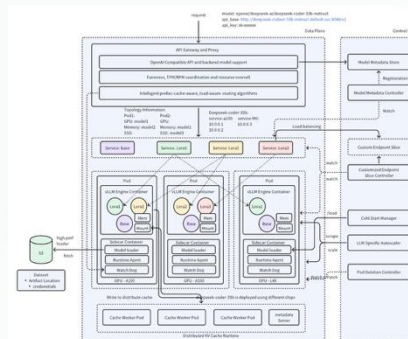
# 在线推理落地实践

业界方案为什么不好在公司内落地

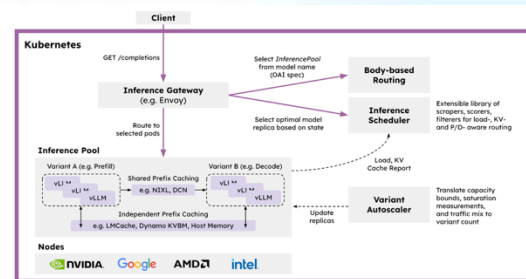


现有PD分离部署框架特点：

- “全家桶”，一般cover全链路
- 通常会引入较重的云原生组件



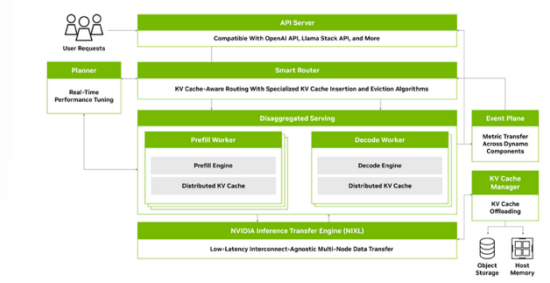
aibrix



llm-d

而公司内的落地需求：

- 尽可能复用已有组件
- 降低平台改造成本



dynamo

# 在线推理落地实践

## 基于 Ray Core的轻量级 PD分离部署方案

### 特点与优势

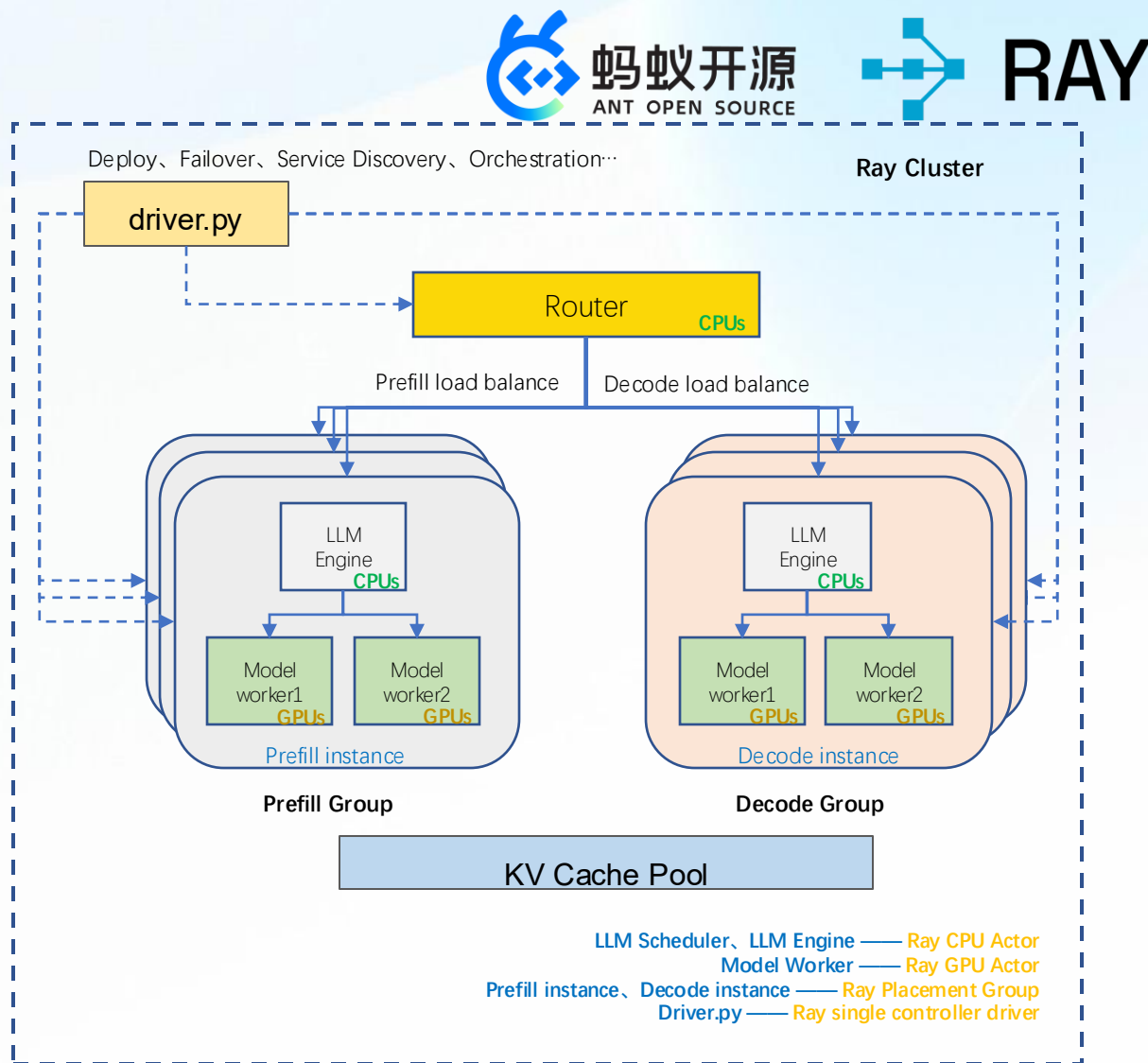
- 多Ray Cluster 合并成统一的大 Ray Cluster
- 轻量化，无需引入任何的云原生组件和新框架
- vllm项目内部工具化集成，命令行从 vllm serve 到 vllm pdjob

### 与Ray Serve LLM区别

方案	Router	支持框架	适用场景
Ray Serve LLM	Ray Serve 内部实现	vllm/sglang	在线推理
vllm pdjob	复用vllm官方任何router实现	仅vllm	在线、离线批量推理、强化学习

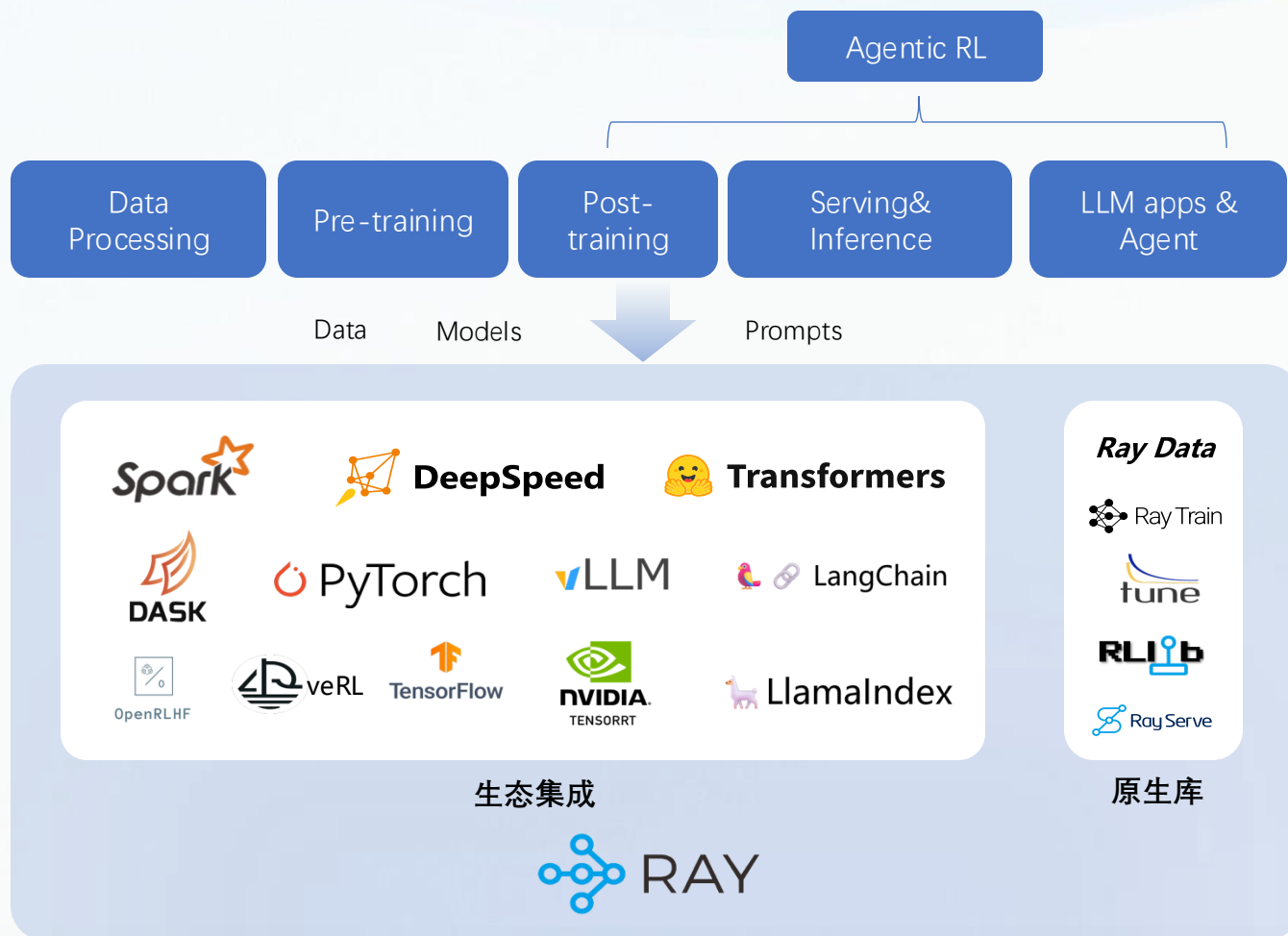
### 开源地址

<https://github.com/vllm-project/vllm/pull/29649>



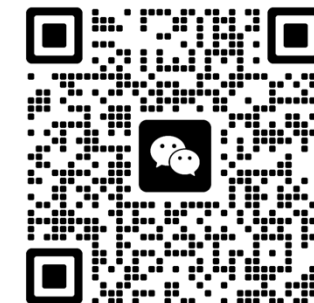
**vllm pdjob --config=/tmp/vllm\_pd\_config.yaml**

# 未来展望



## 未来工作方向

- 离线推理规模 scale up, 支持5k节点规模
- PD分离在在线推理、离线推理、RL中的架构统一
- Agentic RL 融合架构:  
Agent + Sandbox  
+ Inference + training
- 招聘人才



扫一扫上面的二维码图案，加我为朋友。