QCon
全球软件开发大会

InfoQ 极客传媒

基于Ray的AI工程化实践
大模型端到端的训推链路优化

郑志升

B站-基础架构部技术专家

目录

01

场景与痛点概况

Case1 – 应用工具类

数据处理

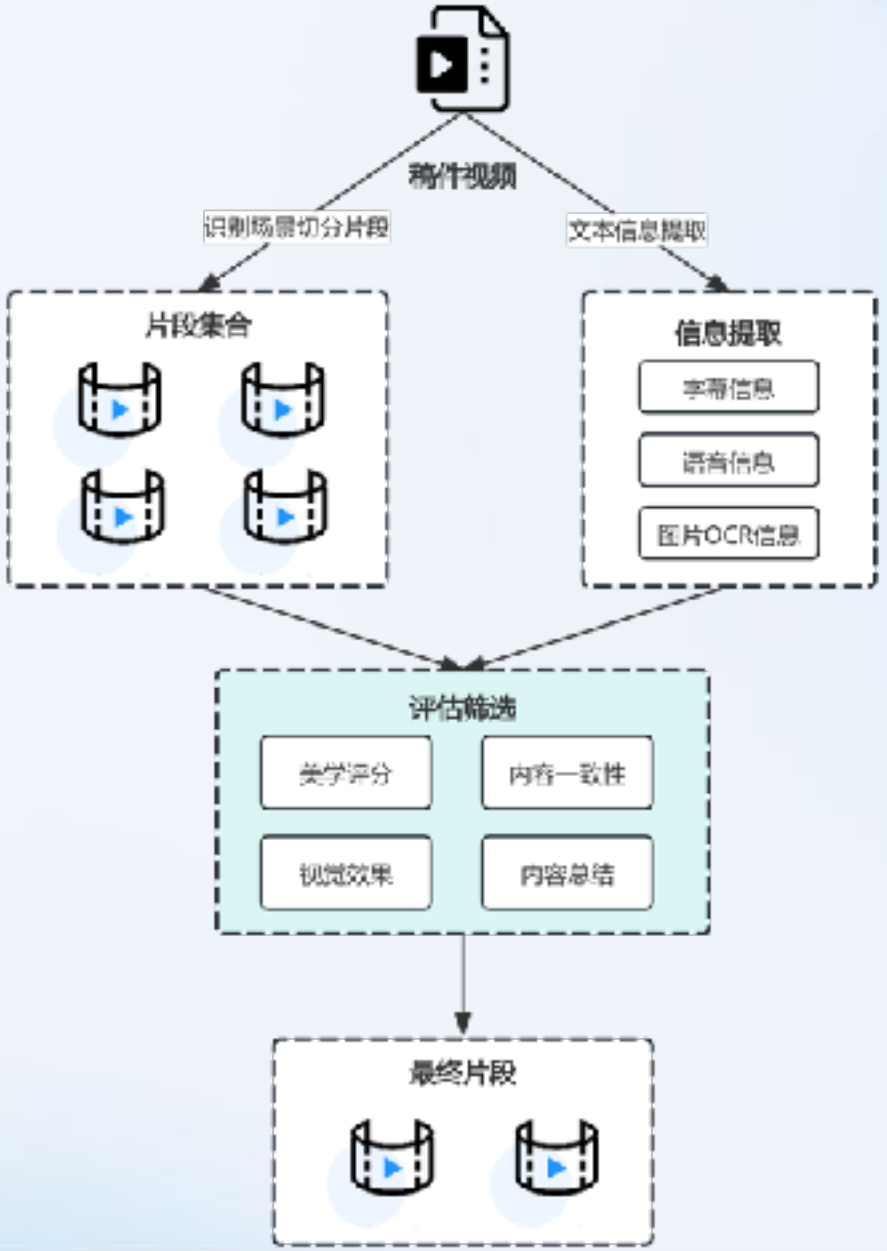
- 字幕提取，CPU密集型
- LLM和多模态推理，涉及GPU计算

多业务诉求

- 部分流程一致，字幕提取、抽帧、抽片段
- 差异在于业务的抽取逻辑，分析逻辑不一致

业务驱动

- 右侧为高光片段链路，典型结合AI能力
- 其他如视频的翻译、文字驱动视频创作工具等
- 对于业务研发，需要搭建复杂计算工程来完成全链路



Case2 – 内容理解类

时效性

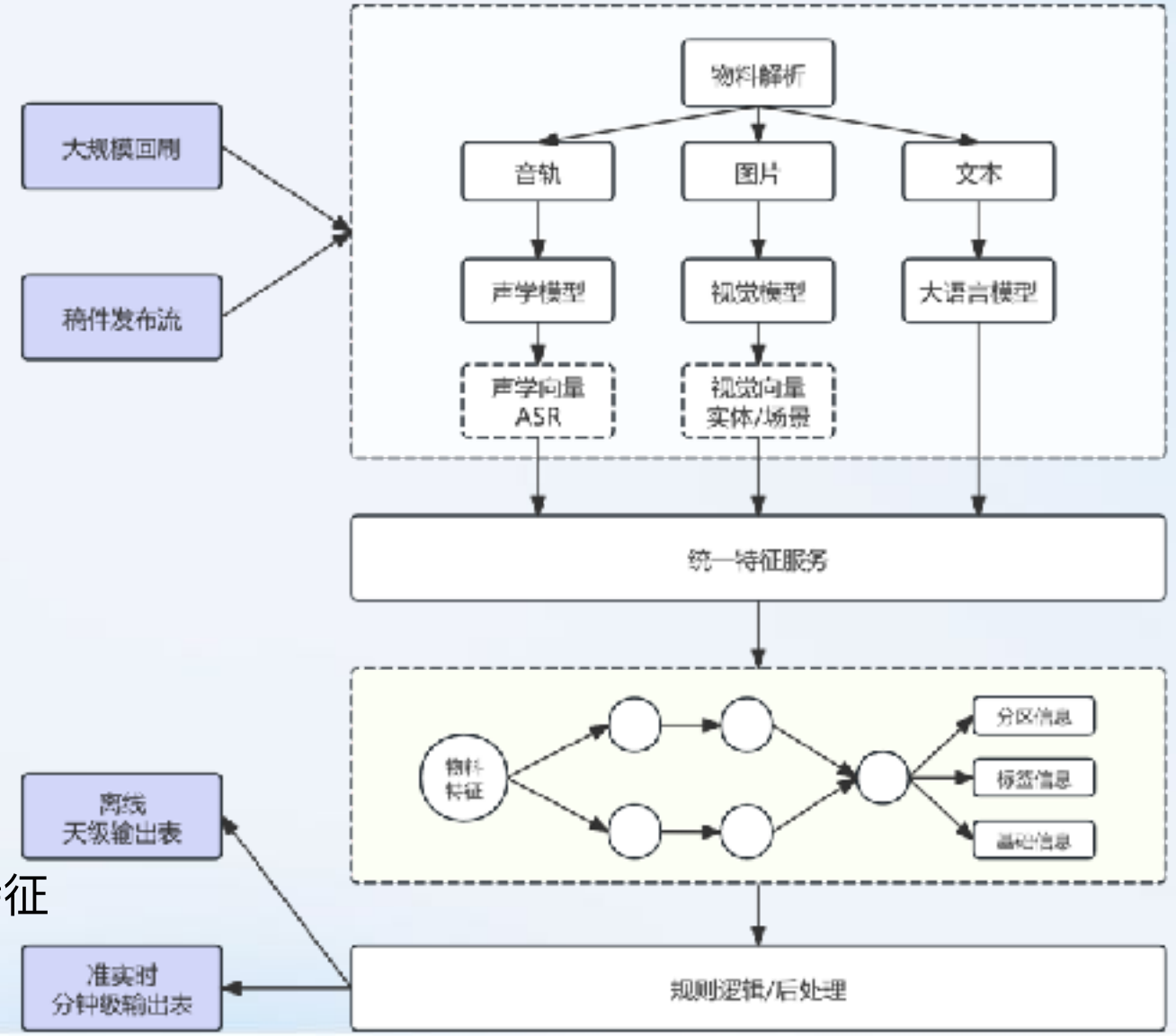
- 线上投稿产生视频稿件流
- 分钟级计算，产出特征和模型推理结果

大规模回刷

- 新增不同特征或模型迭代，历史需回溯

复杂Pipeline

- 计算量大，多步骤特征提取和标签生成
- 偏多模态数据中台类服务，提供视频多维特征



Case3 – 特征融合类

传统搜推广

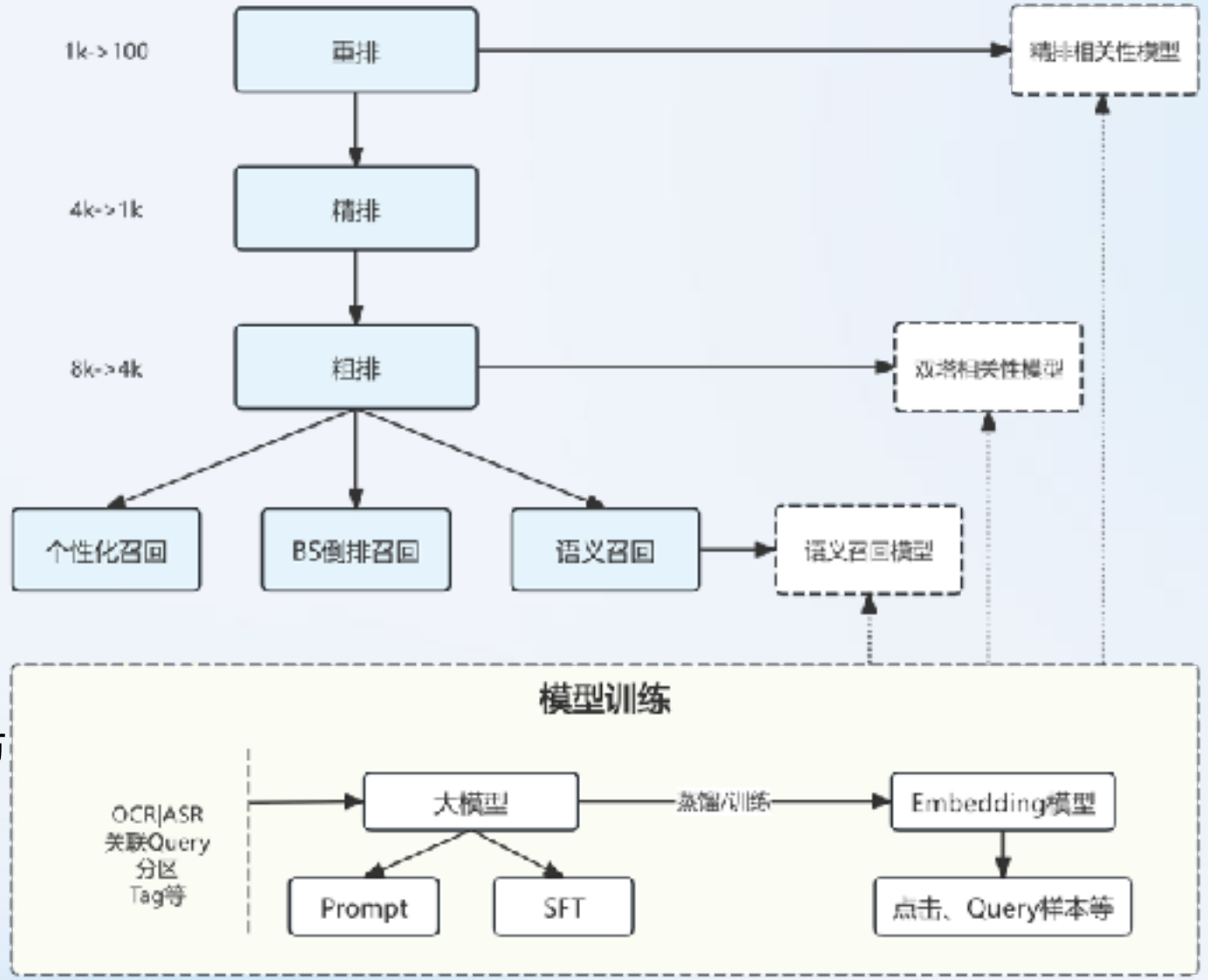
- 线上多阶段，融合多模态特征

多模数据处理

- 相关性特征提取，ASR/OCR提取

LLM蒸馏模型

- 微调LLM模型，进一步蒸馏Bert模型
- 解决语义召回，双塔及精排相关性等环节



Case4 – 强化训练

多角色训练

- 四大模型，既有推理，也有训练
- Policy/Rollout/Reward/Reference

灵活多样训练方法

- PPO、GRPO、DAPO、Agentic Tool

工程效率与成本

- RL训练成本高，如何优化全链路MFU

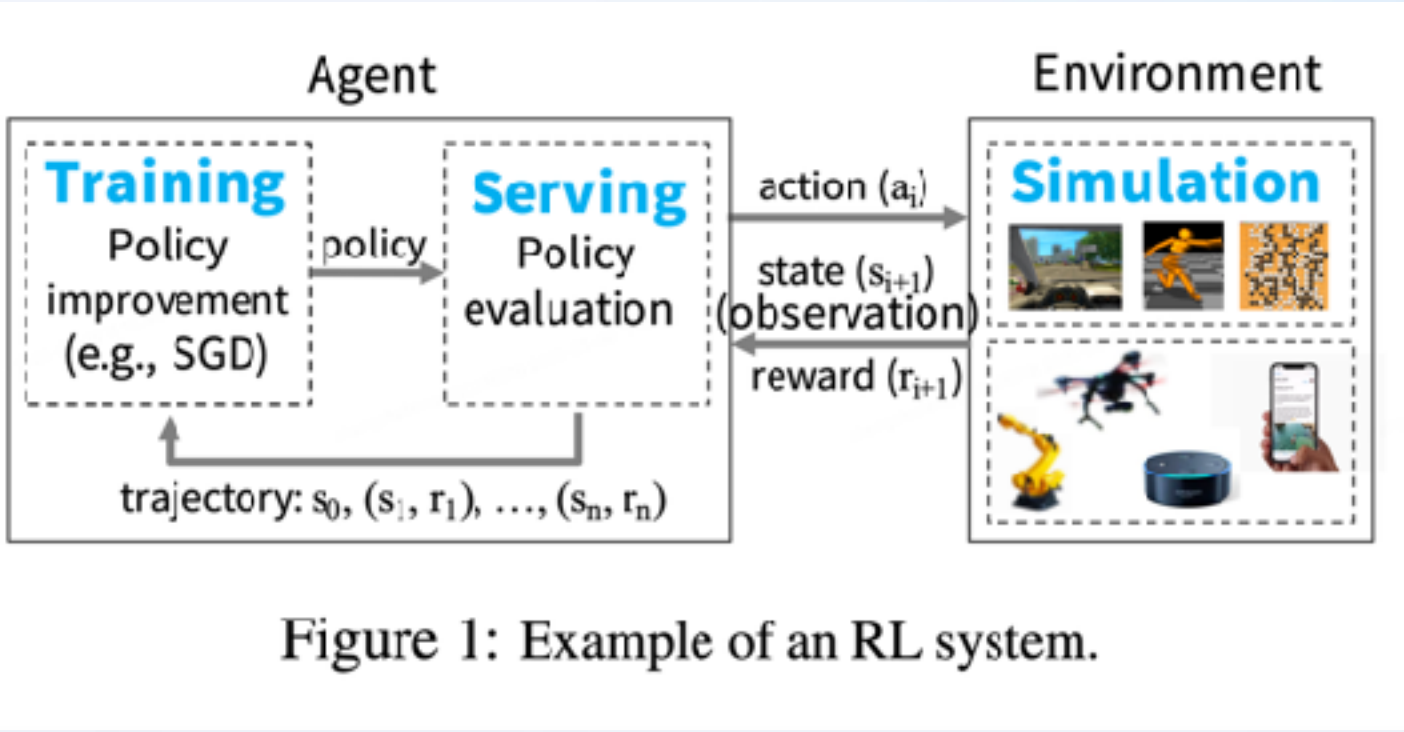


Figure 1: Example of an RL system.

多模态 – 工程痛点

烟冲式架构

大模型的应用结合

- 大模型时代的到来，M/LLM标配
- 缺乏模型微调到应用推理的工程标准

多模态数据计算共性

- 重复建设，如切片、抽帧、OCR、ASR等

资源成本效率

大量的业务，微服务

- 资源碎片化，难以规则和复用
- 缺乏灵活弹性，无法更好的按需供给

GPU计算效率低

- 消费卡/高性能卡/显存卡，无使用标准

问题剖析视图

复杂编排能力

多Pipeline

- 视频–>分片/抽帧–>MLLM推理

异构计算

- 混合CPU+GPU
- 各类混合GPU资源，如H20/A10/N910B

时效性诉求

天/小时级，离线计算

- 大规模数据回溯
- 模型调研与训练

分钟/秒/级，近实时计算

- 视频稿件的内容理解实时订阅

引入Ray

计算原语

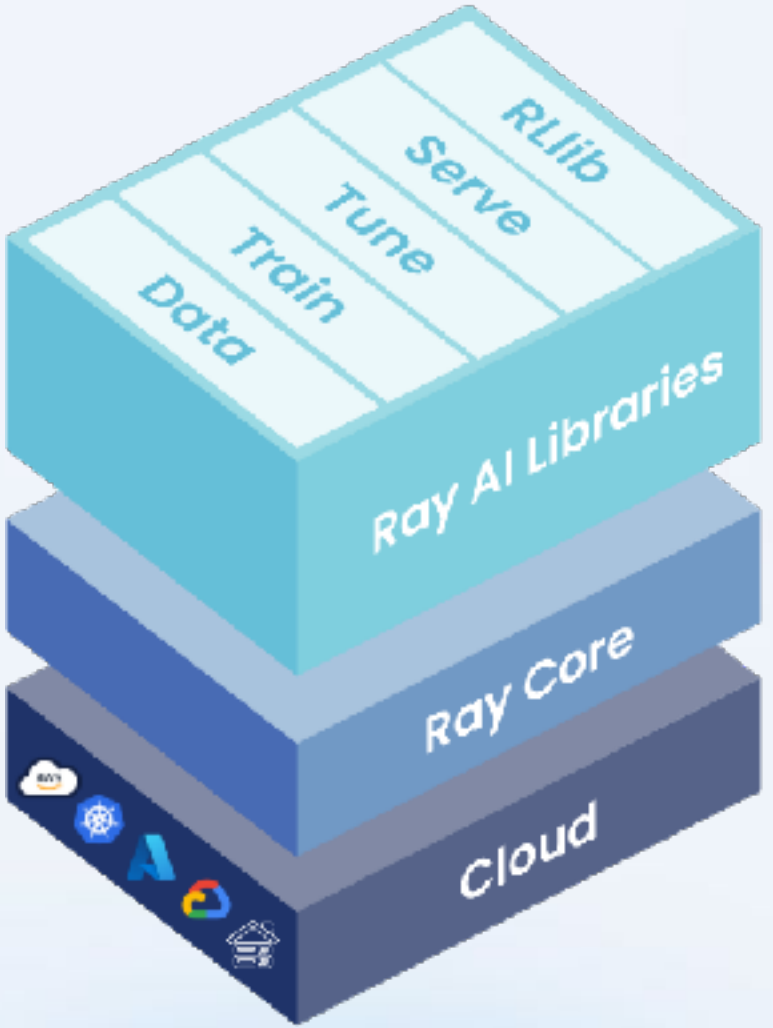
- Actor、Task
- 动态任务图计算模型
- 细粒度并行+异构计算（函数级）

灵活弹性

- 云原生，提供多层次的资源弹性

AI生态丰富

- Core底座C++，上层丰富Python库



high-level libraries that enable simple scaling of AI workloads

a low-level distributed computing framework with a concise core and Python-first API

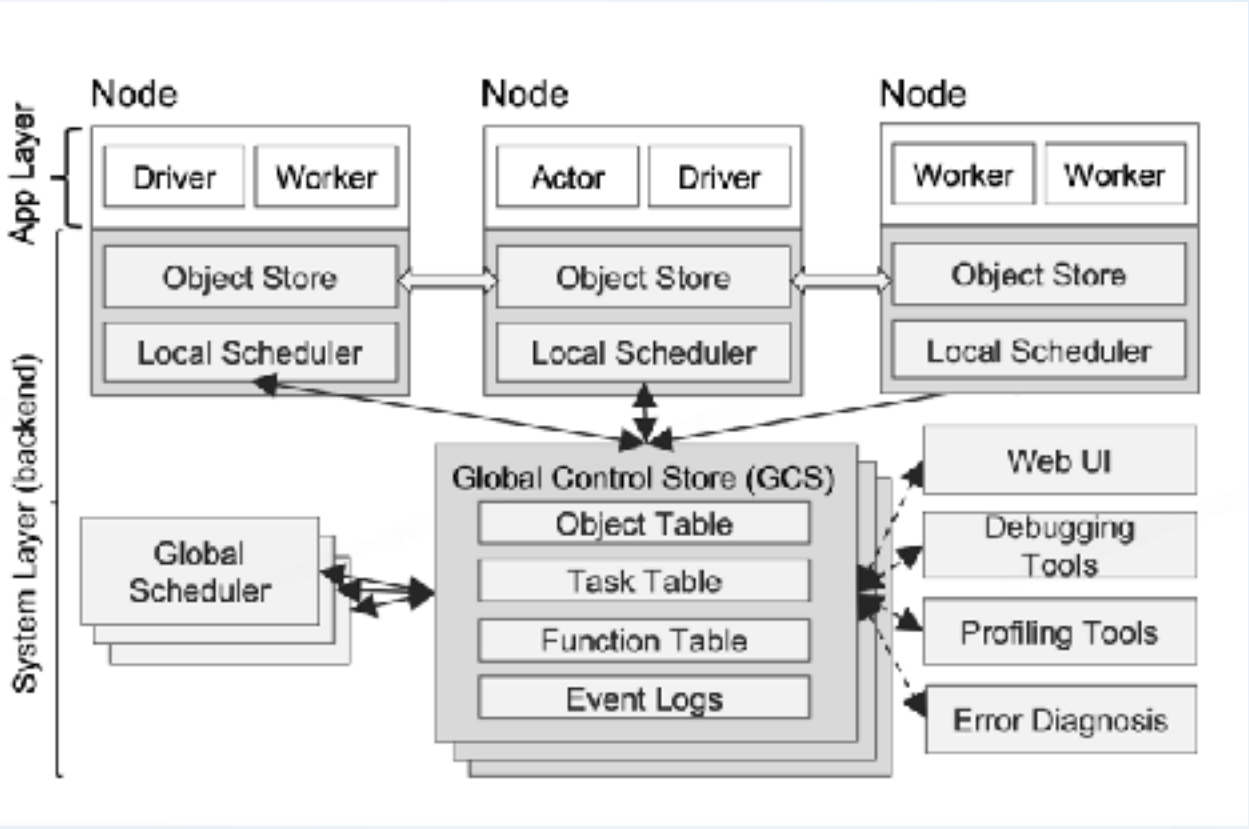
Ray的架构

系统层

- Distribute Scheduler
- Distribute Object Store
- Global Controller Service

应用层

- Driver，用户程序入口，Job执行进程
- Actor，对应class类，有状态行动器进程
- Worker：对应method，无状态task执行载体



Ray的生态优势

DAG计算

- Core之上，官方提供Data组件
- Daft基于Ray的分布式多模态计算引擎

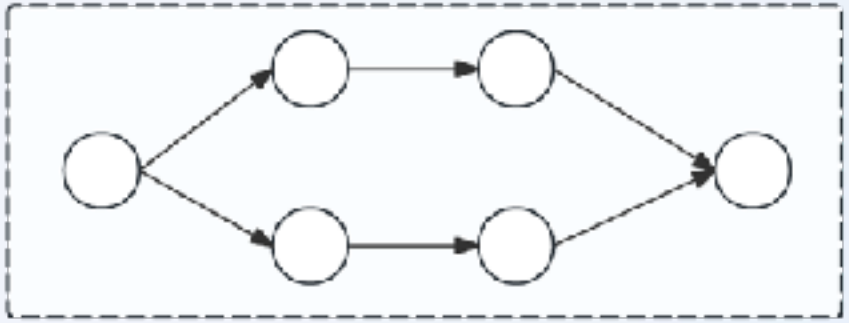
LLM结合

- 官方Serve，内置LLM多种分布式部署

大模型训练

- 当下火热的RL，开源OpenRLHF和VeRL

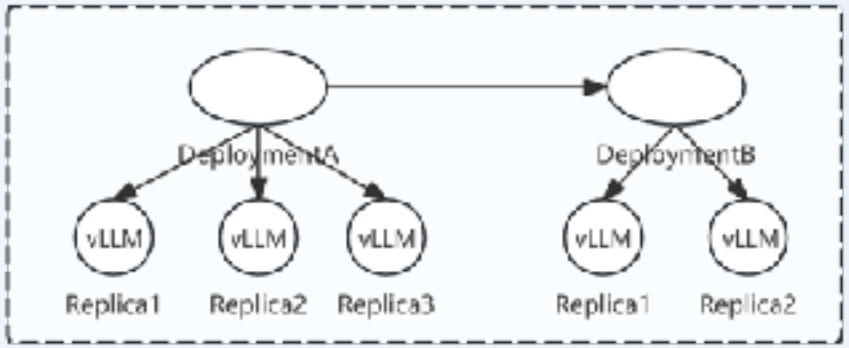
DAG Pipeline



Ray Data

Daft

Serve LLM

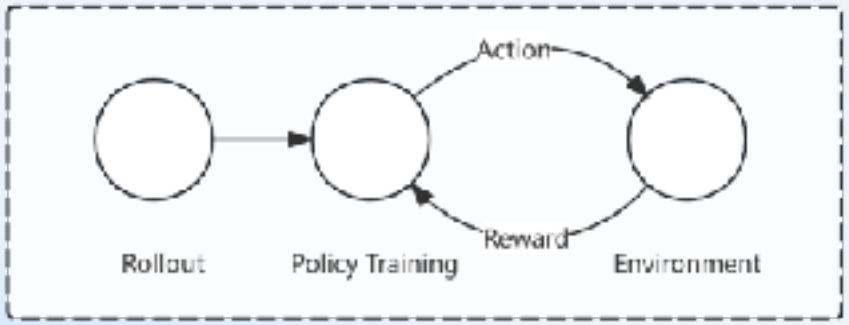


Ray Serve

Vllm PD On Ray

AiBrix

RL Train



OpenRLHF

VeRL

Ray Train

02

Ray+LLM
分布式流批推理管道

Ray Data – 架构

Design

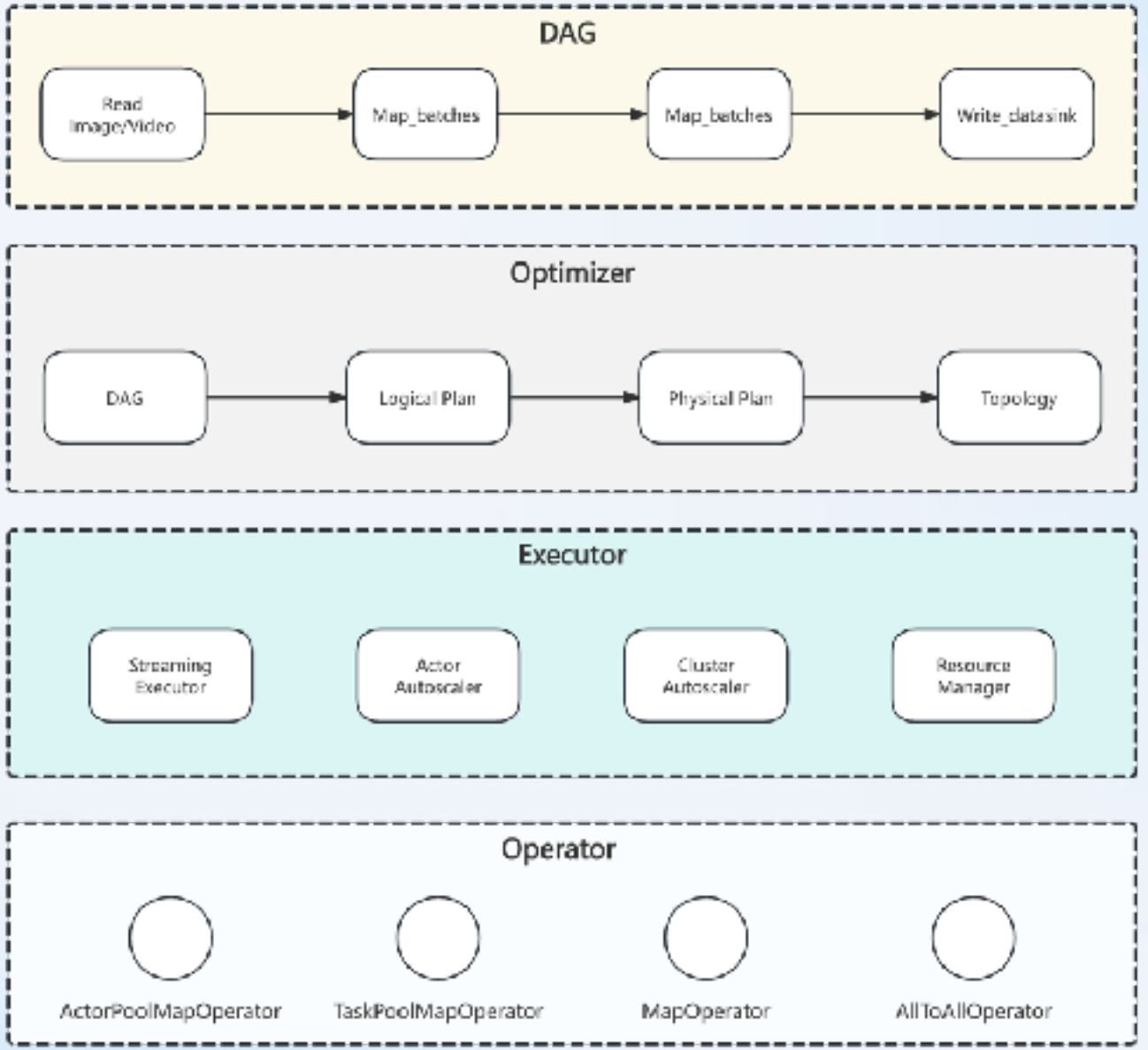
- DAG调度
- 自带反压机制
- 丰富Source/Sink生态
- 弹性机制+细腻度资源调配

融入LLM

- LLM Operator，支持主流大模型
- 对接推理生态，支持vLLM、SGLang

存在优劣

- 解决离线计算链路，但无法满足高时效场景



Data – 流计算

数据流调度

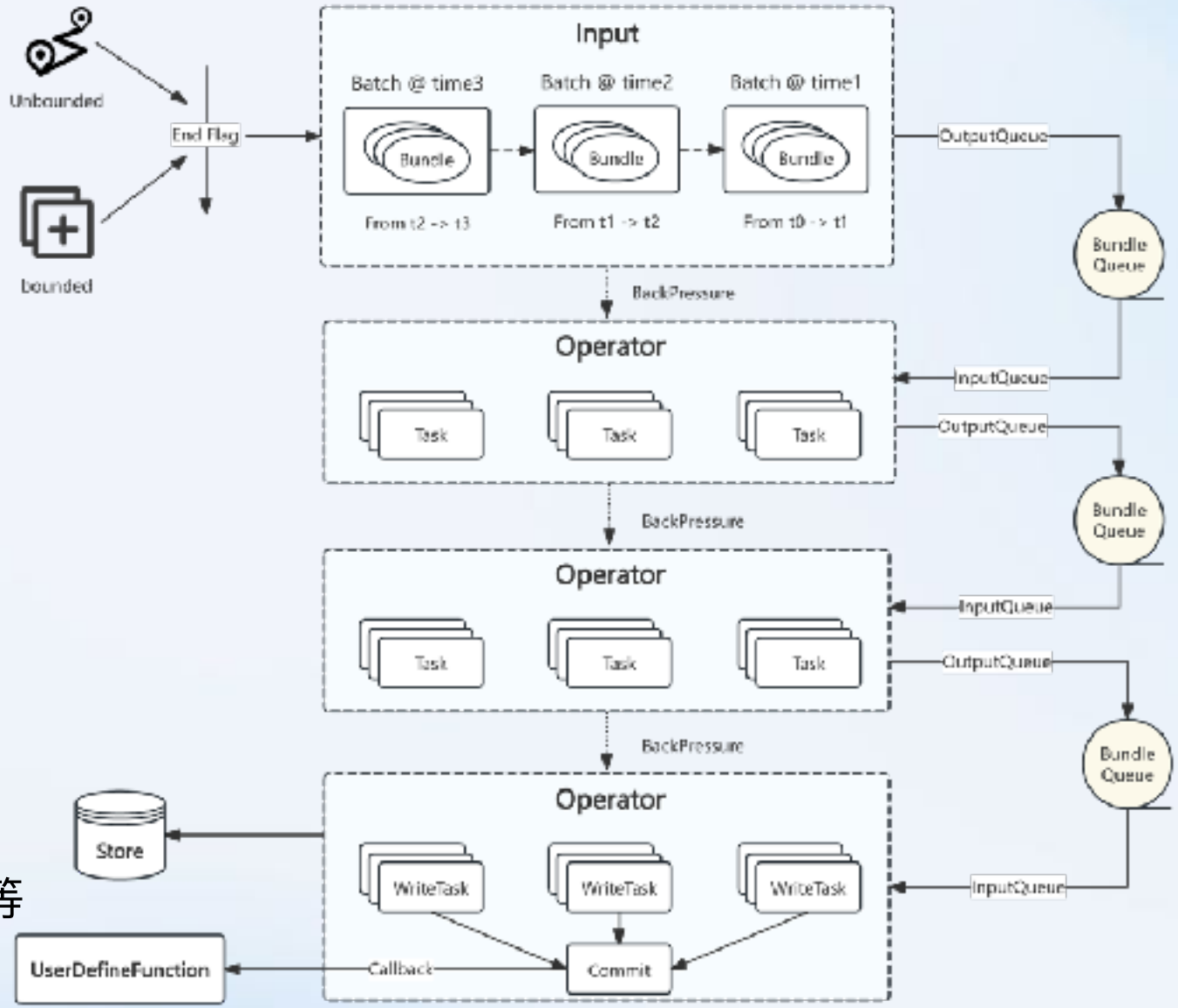
- 离线方式统一抽象为Stream
- Stream分为UnBounded和Bounded
- 引入End Flag用于控制数据流的结束标志

Backlog机制

- 基于下游Queue的数据积压做反压调控
- 低峰数据的IDLE超时下发机制，避免空等

回调通知

- 完成数据小批量回调通知用户，业务做幂等



Data – Source/Sink

Kafka

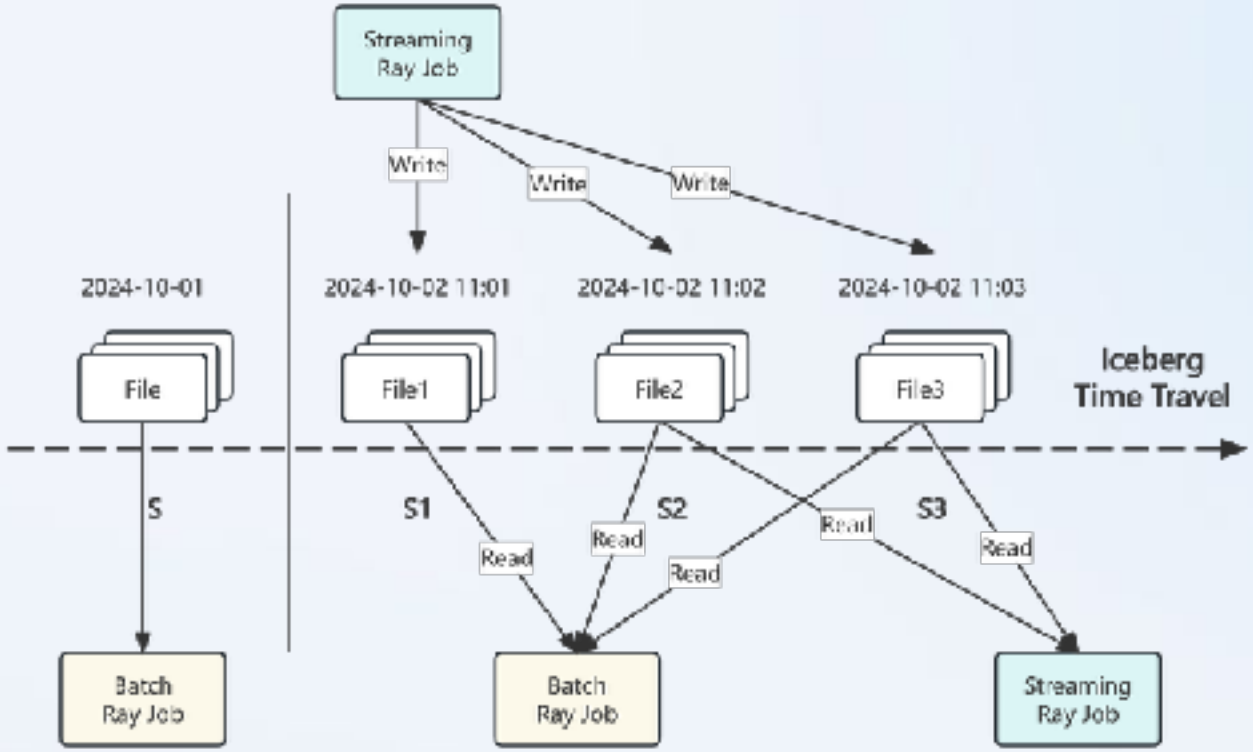
- 实时秒级场景，流式读写消息队列

Iceberg

- 近实时场景，流和批打通数据孤岛
- 上游的Ray流作业按Snapshot增量写入
- 下游批和流作业消费不同Snapshot Between

多场景复用

- 回刷、离线推理、离线训练、近实时（分钟级）



案例/特征工程 – 流批管道

现状痛点

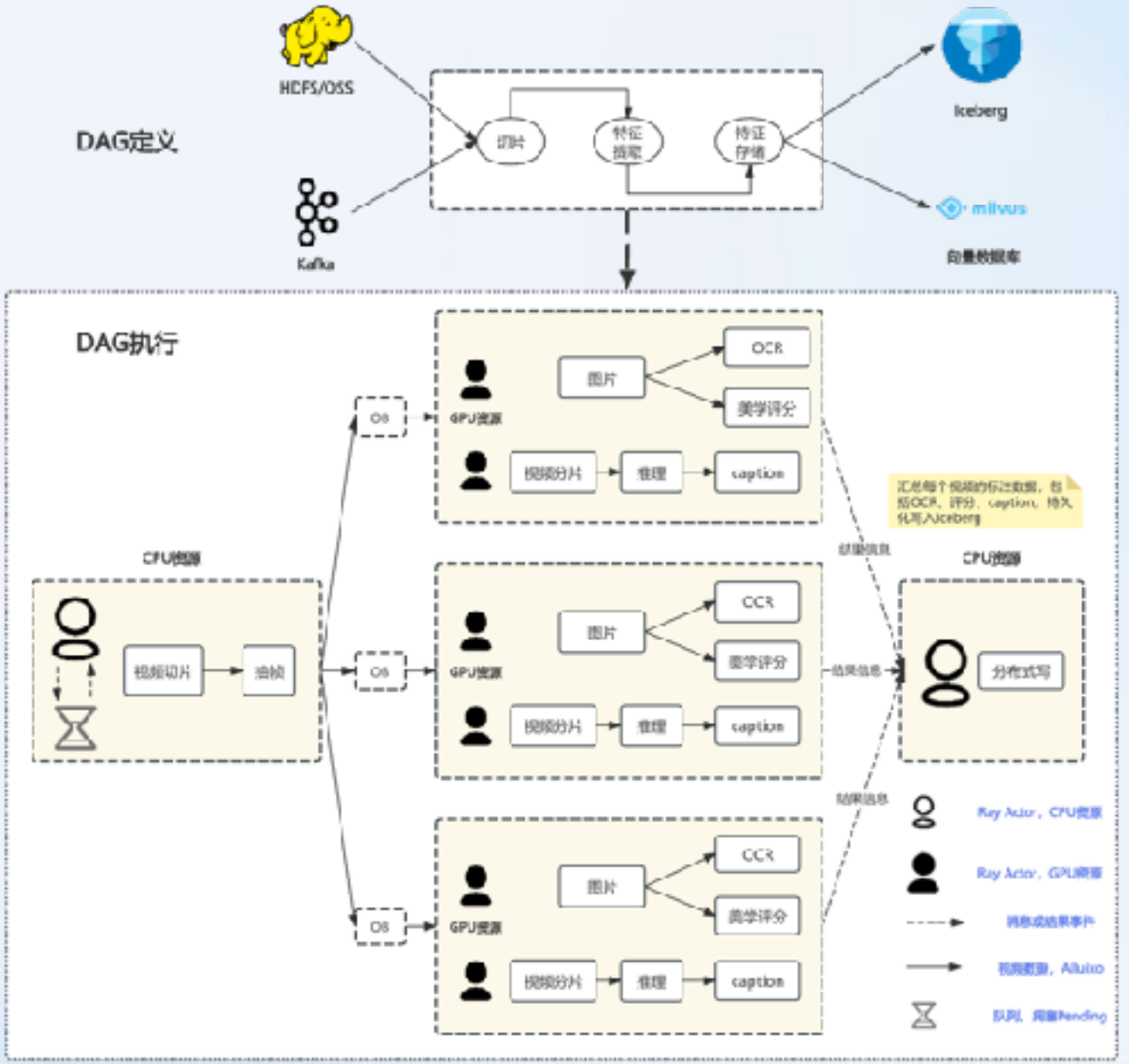
- CPU切片/Spark，GPU推理/k8s
- GPU利用率低，链路断层时效不足

DAG定义

- 业务定义计算流程
- 抽象每个Stage的算子逻辑
- 对于特征场景，实时离线代码共用

效果优劣

- 吞吐提升4倍，GPU日均利用率75%+
- 需幂等保障恢复，否则回刷失败从头计算



Data – 引入Checkpoint

DAG增强

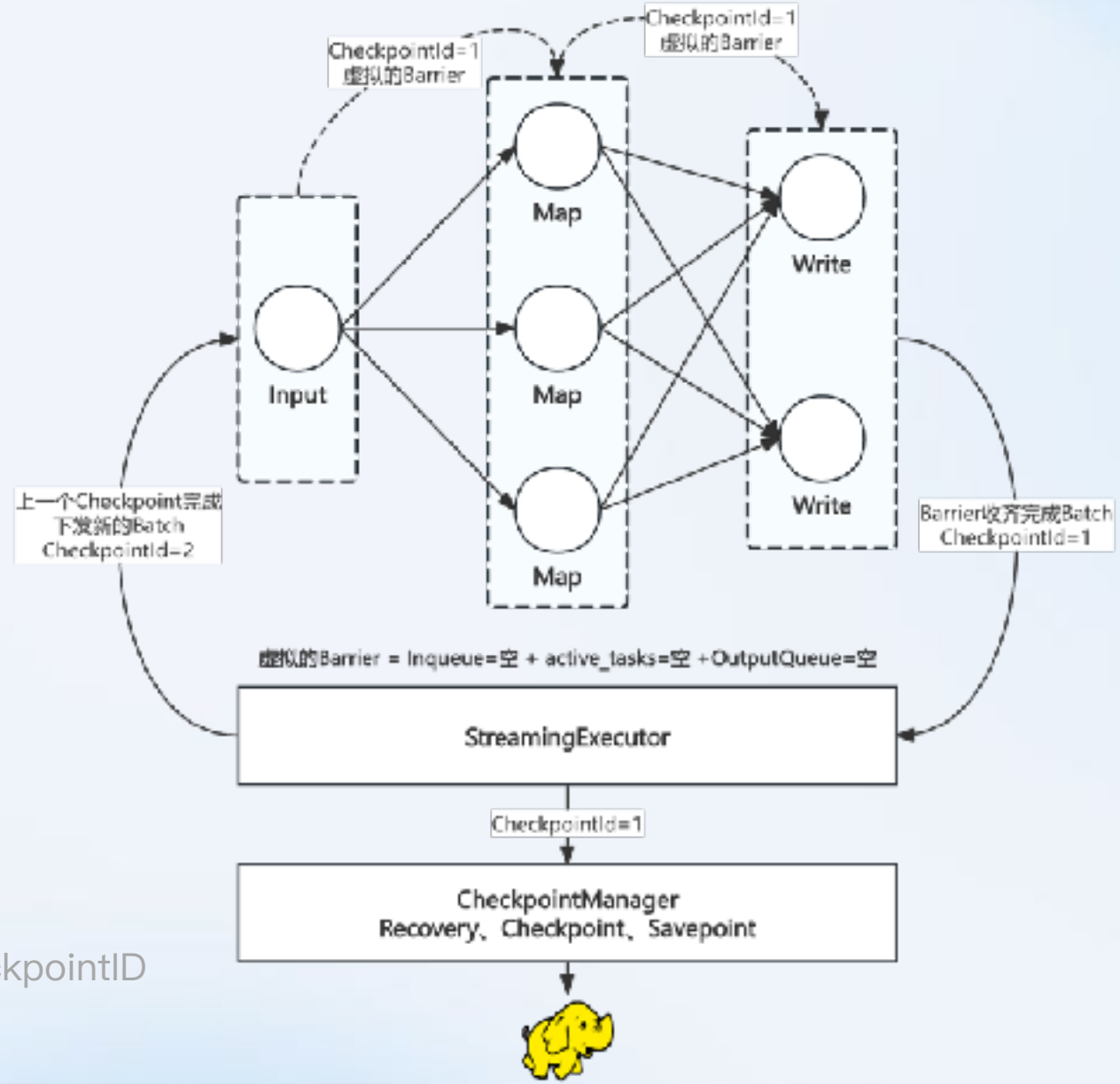
- 标记CheckpointID
- 虚拟Barrier记录每个环节进度

CheckpointManager

- 运行时的状态持久化，存储到远程HDFS

标准化API

- Source支持Checkpoint状态的恢复机制
- 扩展HDFSSource，大规模离线回刷场景
  - 切分HDFS目录划分BatchGroup，一一对应CheckpointID



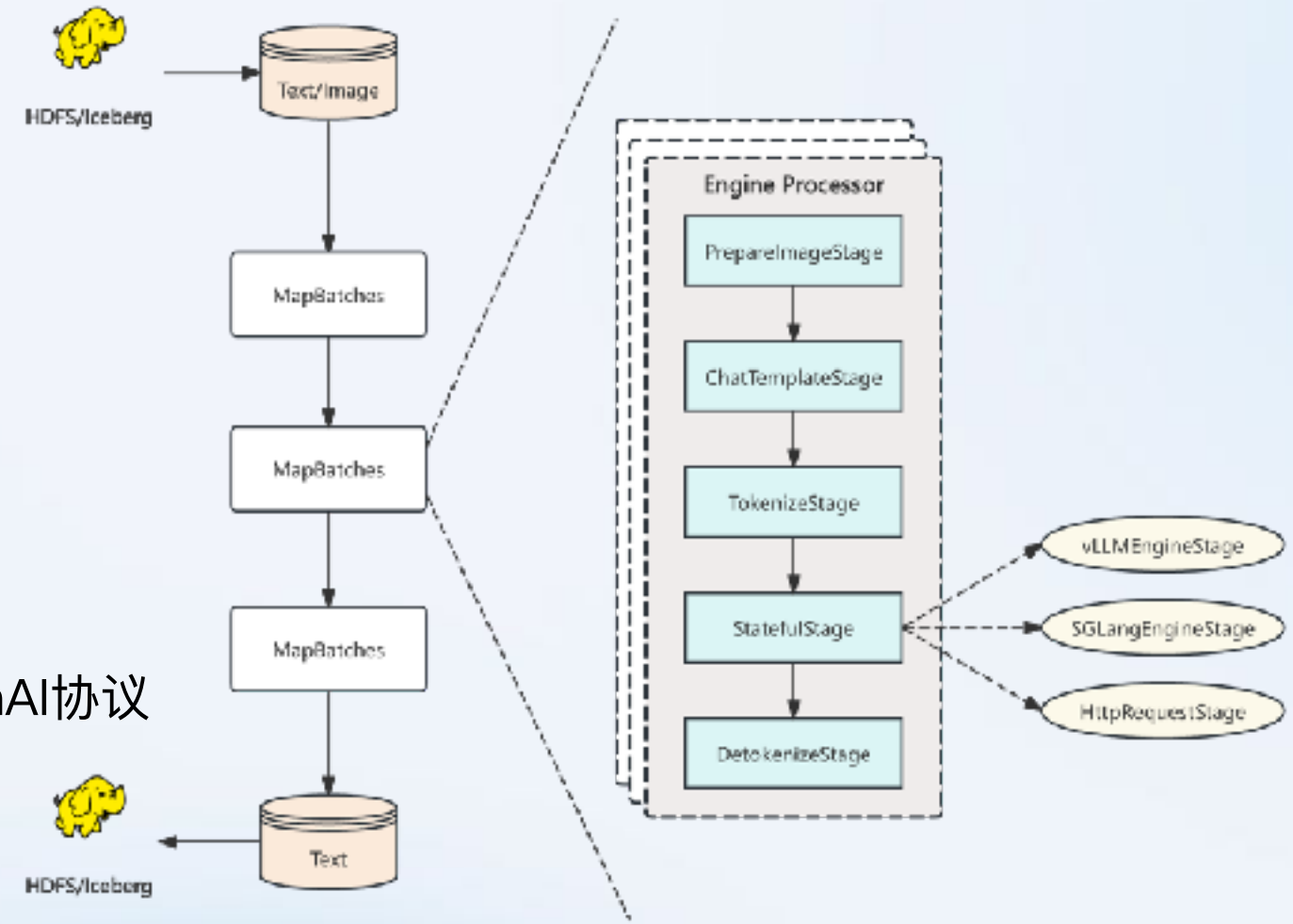
Data – LLM推理

融入LLM

- LLM Operator，支持主流大模型

整合社区新版

- Source/Sink复用生态
- 标准化LLM流程，Processor+Stage
- LLM支持vLLM/SGLang，也支持OpenAI协议



案例/特征融合 – 广告长序列

两阶段建模

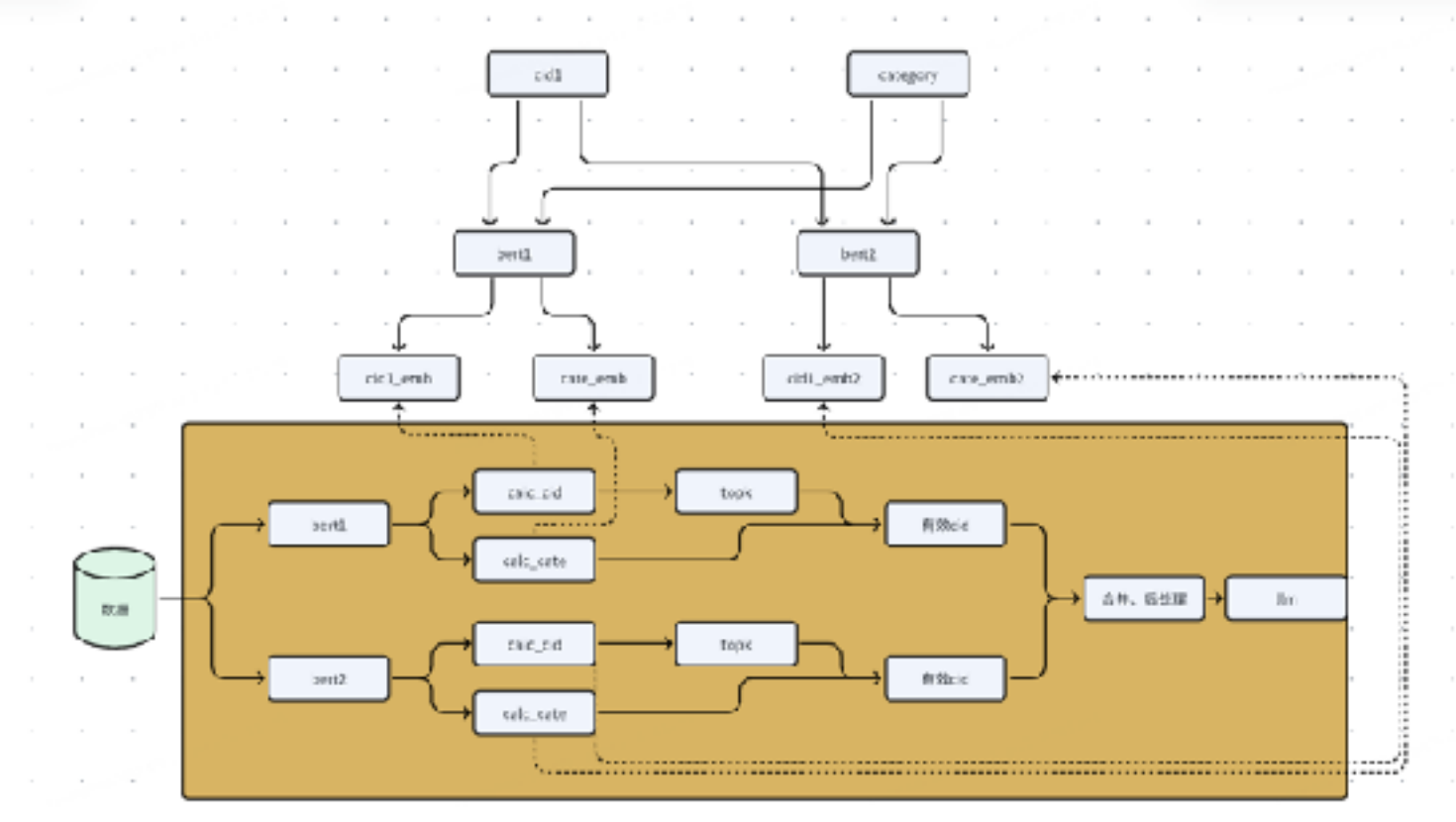
- GSU需要大规模回刷离线

引入Ray

- 基于Bert，初始化emb特征
- 用户历史基于Bert+emb计算topk
- 进一步输入Qwen7B产出用户兴趣

落地效果

- GPU利用率高，Ray弹性凌晨潮汐的GPU资源，可更好满足业务吞吐
- 极大降低离线场景接入Ray的工程门槛



Data – 实时工程门槛

更多实时场景

- 门槛低，Checkpoint?
- 链路更长更复杂，引入了Shuffle

流模式下的问题

- 同步Checkpoint，Stage间无法并行
- 业务对数据的失败希望有更灵活的重试策略
- 低延时下链路缺乏细粒度可观测，如视频处理进度

思考剖析

- 满足效率的同时提供灵活性，并行Checkpoint？记录级ACK?

Data – 并行Checkpoint

Bundle扩展

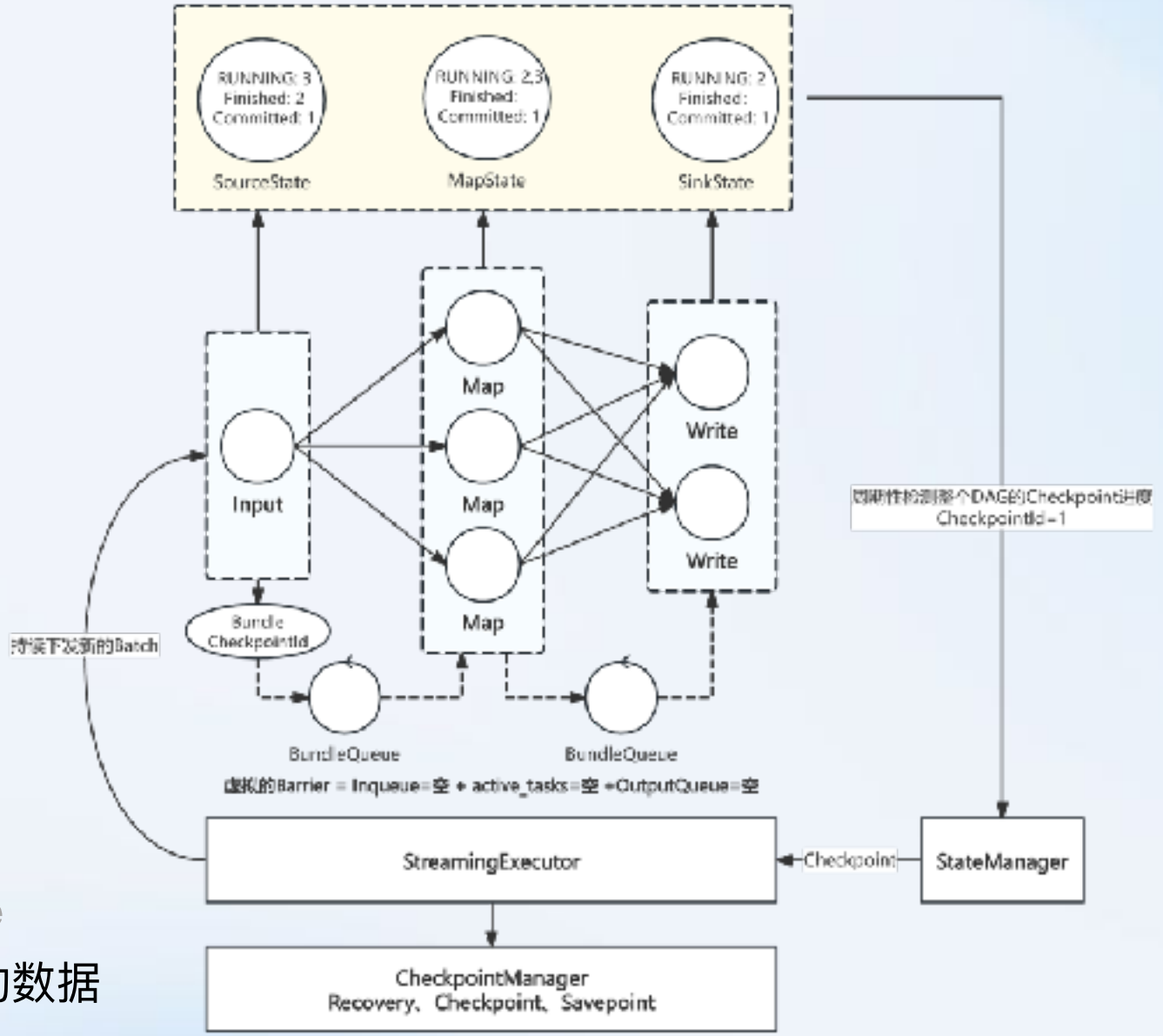
- 数据附带CheckpointID
- Stage流转标记CheckpointID血缘

状态管理器

- 记录DAG各个阶段的State
- Barrier收齐检测，触发Checkpoint

Shuffle扩展

- 原生Shuffle是前置流程处理完所有数据
  - 支持Push-based和Pull-based Shuffle
- 下发upstream完成的CheckpointID对应的数据



案例/应用类 – 视频检索生成

流式处理

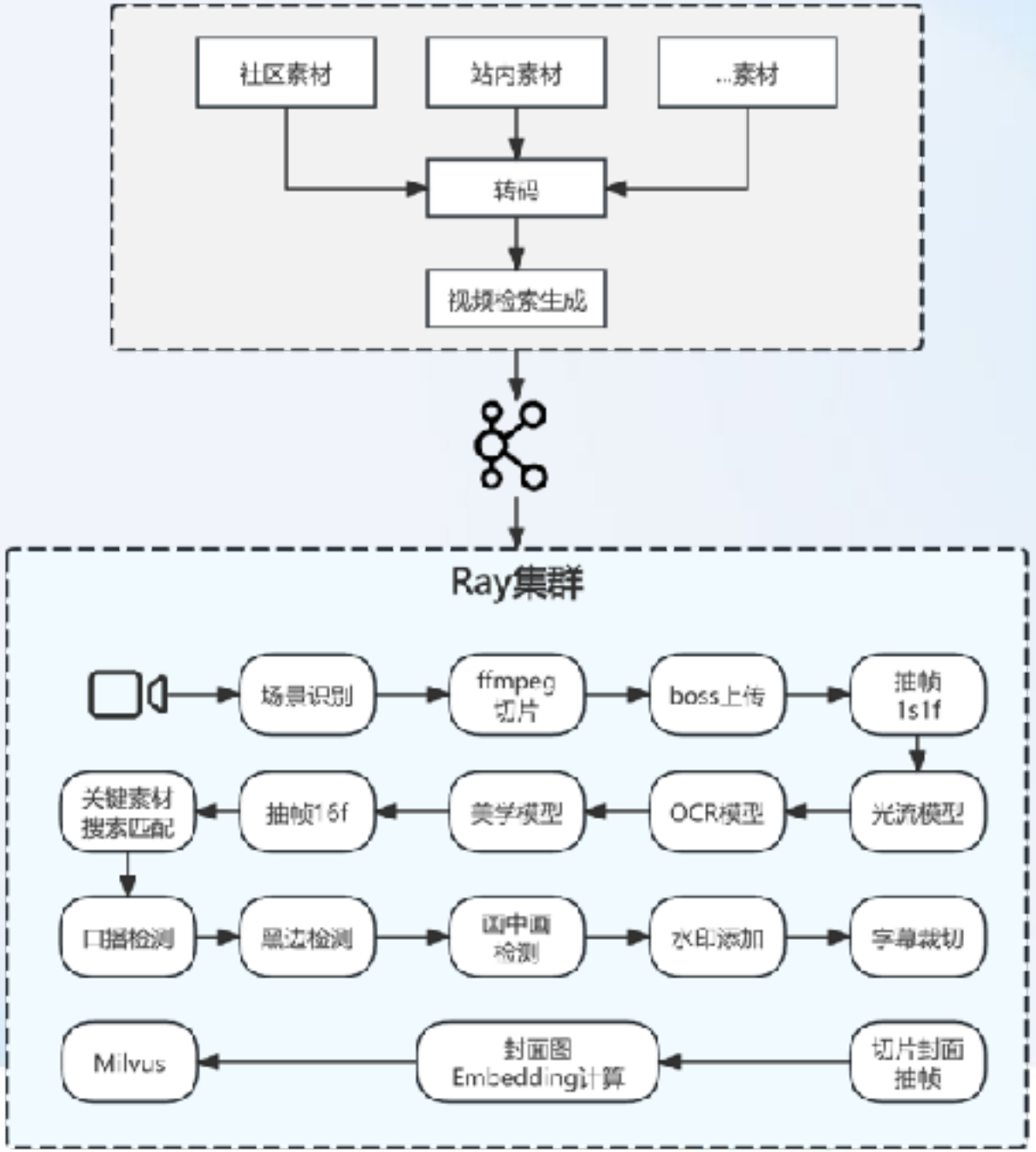
- 上游下发视频素材
- 多Stage阶段分布式处理

计算效率

- CPU+GPU异步并行计算
- 资源弹性，解决多Stage气泡问题

Checkpoint机制

- 3s一次Checkpoint，视频断点恢复



Data – Identifier ACK

ID机制

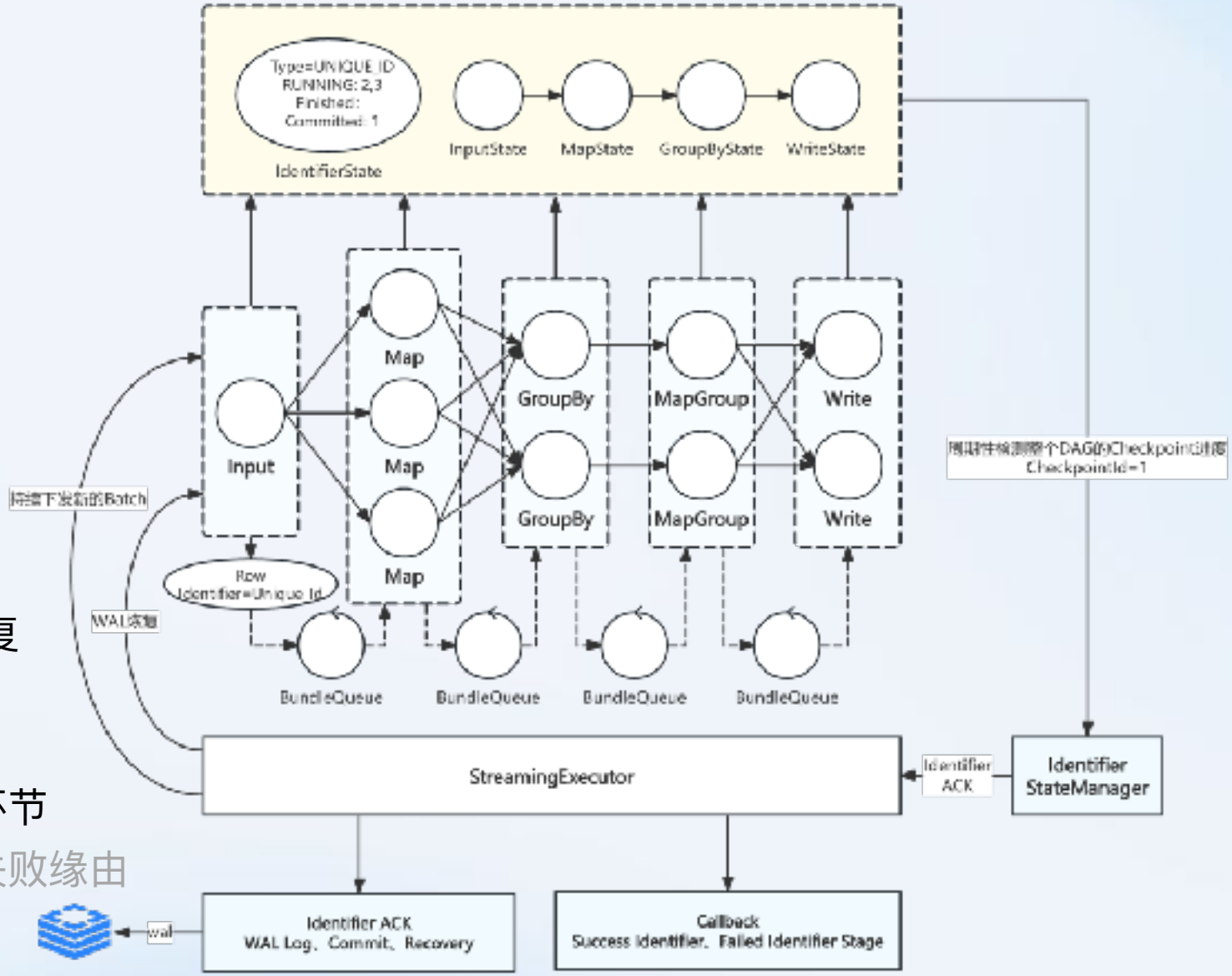
- CheckpointID
- 用户自定义UniqueID
- 通用–Checkpoint，定制–Unique

记录级容错

- 针对失败的ID，支持Callback通知
- 基于Redis引入WAL和Commit断点恢复

可观测

- 基于MessageID，便于排障问题链路环节
  - 如监控视频粒度的处理进展，以及失败缘由



案例/内容理解 – OCR服务

计算模式

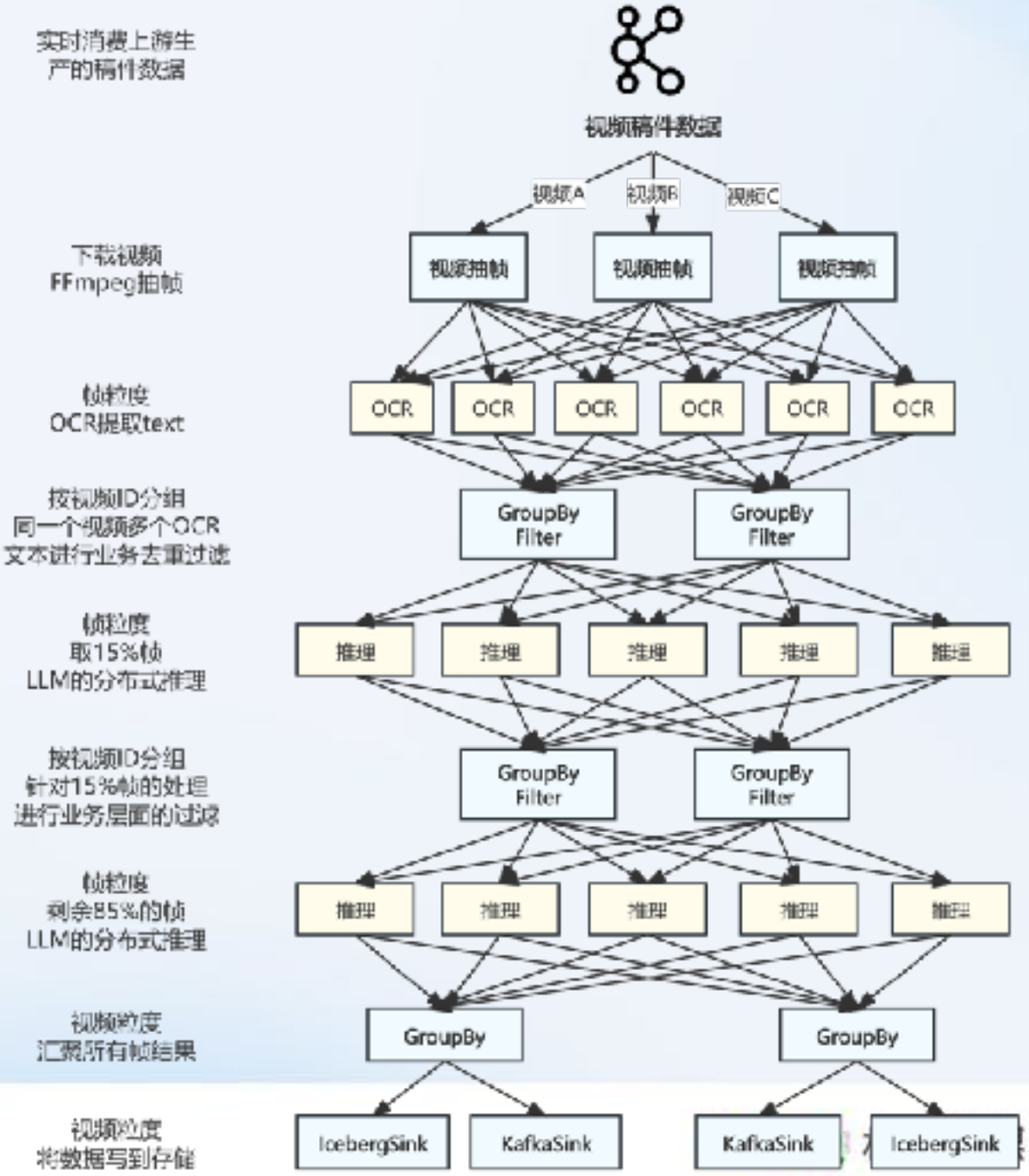
- 多阶段Shuffle
- 分片粒度分布式并行计算
- 视频粒度汇聚，层层过滤（2层）

视频容错

- 分片处理基于task原生的Retry策略
- 异常失败则Callback用户自定义幂等策略
- Shuffle过程，先完成视频快速下发下游计算

进一步Case

- 资源复用？弹性效率？调度策略？计算吞吐？



Data – Others

极致按需弹性

- 凌晨半夜无流量，支持min为0的弹性，GPU回收

有序调度

- FIFO模式，基本有序调度
- 先到的数据先完成处理，如视频处理

优先级调度

- 长视频处理耗时，容易堵塞下游GPU消费
- 针对短视频分片优先处理，长视频分片填充利用率

共享Actor Pool

- Map推理池分太细，利用率低
- 多个Map阶段可共享Pool，减少弹缩和碎片

Data – Autoscaler

反复弹缩

- 模型推理类计算，冷启动慢
- 过频繁弹性，导致利用率低和吞吐差

防抖机制

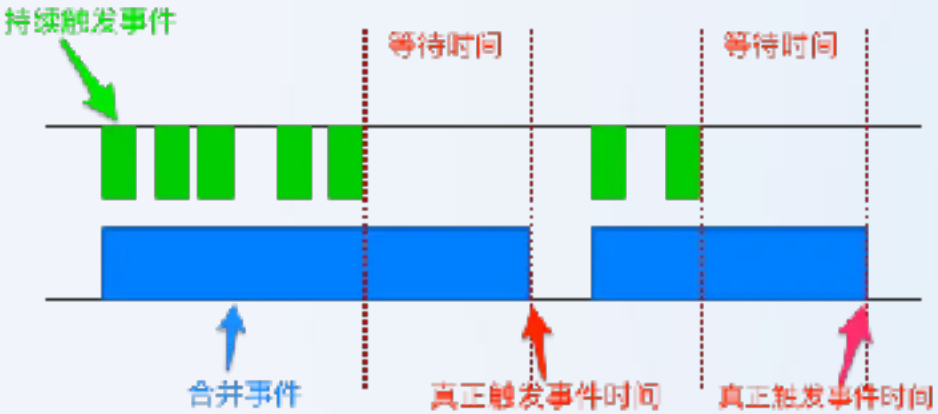
- 防止1s内Pool触发多次弹性（含扩和缩）

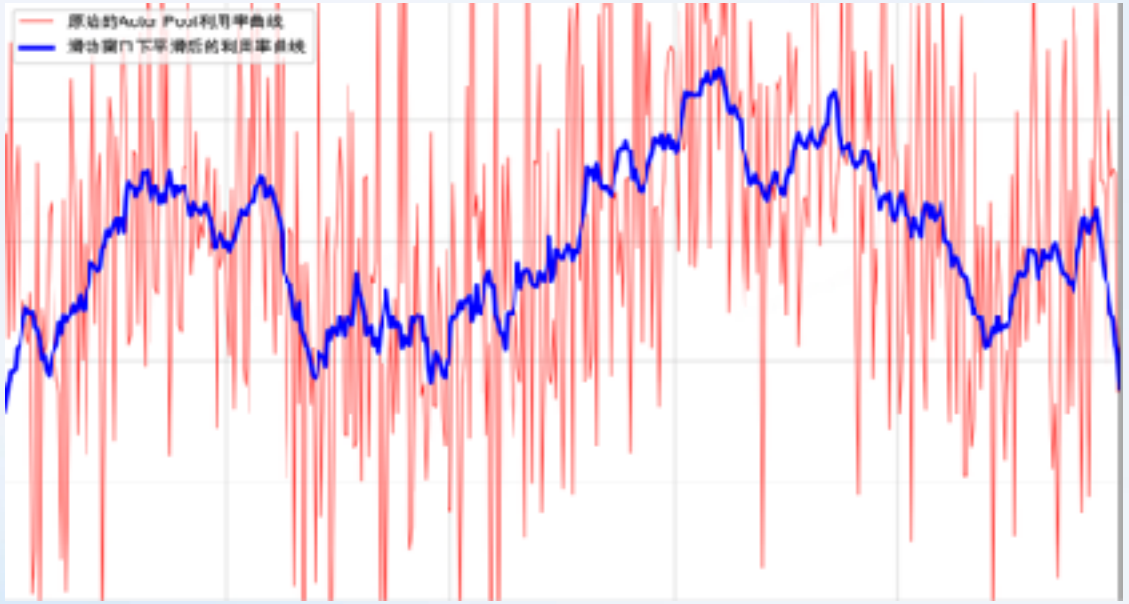
滑动窗口策略

- 利用率 = Active Num/Total Num
- 原生瞬时值计算Pool利用率，不稳定性
- 滑动计算5s内各个Pool的利用率max/min

弹性可观测

- 反压原因可视化、各个阶段弹性次数和耗时





Ray Autoscaler – 多k8s池融合

资源割裂

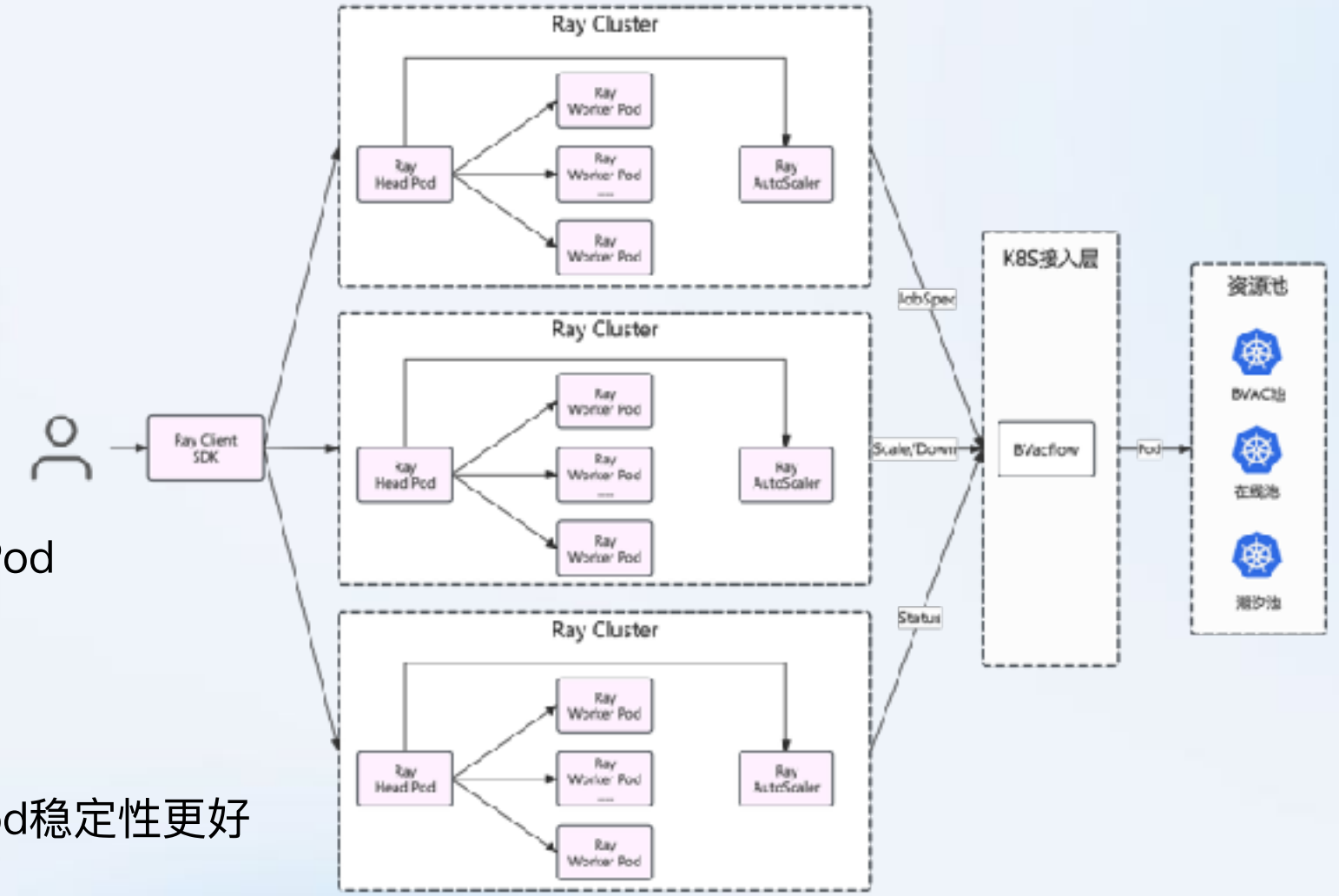
- 在/离线多套k8s池子
- GPU难以灵活弹性使用，如潮汐

调度接入层

- 对接多套k8s资源池
- Pod粒度申请管控，区分优先级
- Ray集群对应多个池子的Worker Pod

Autoscaler

- 按Pod优先级申请和释放资源
- 如潮汐Pod优先释放，独占资源Pod稳定性更好



Ray Head – 高可用

Head HA

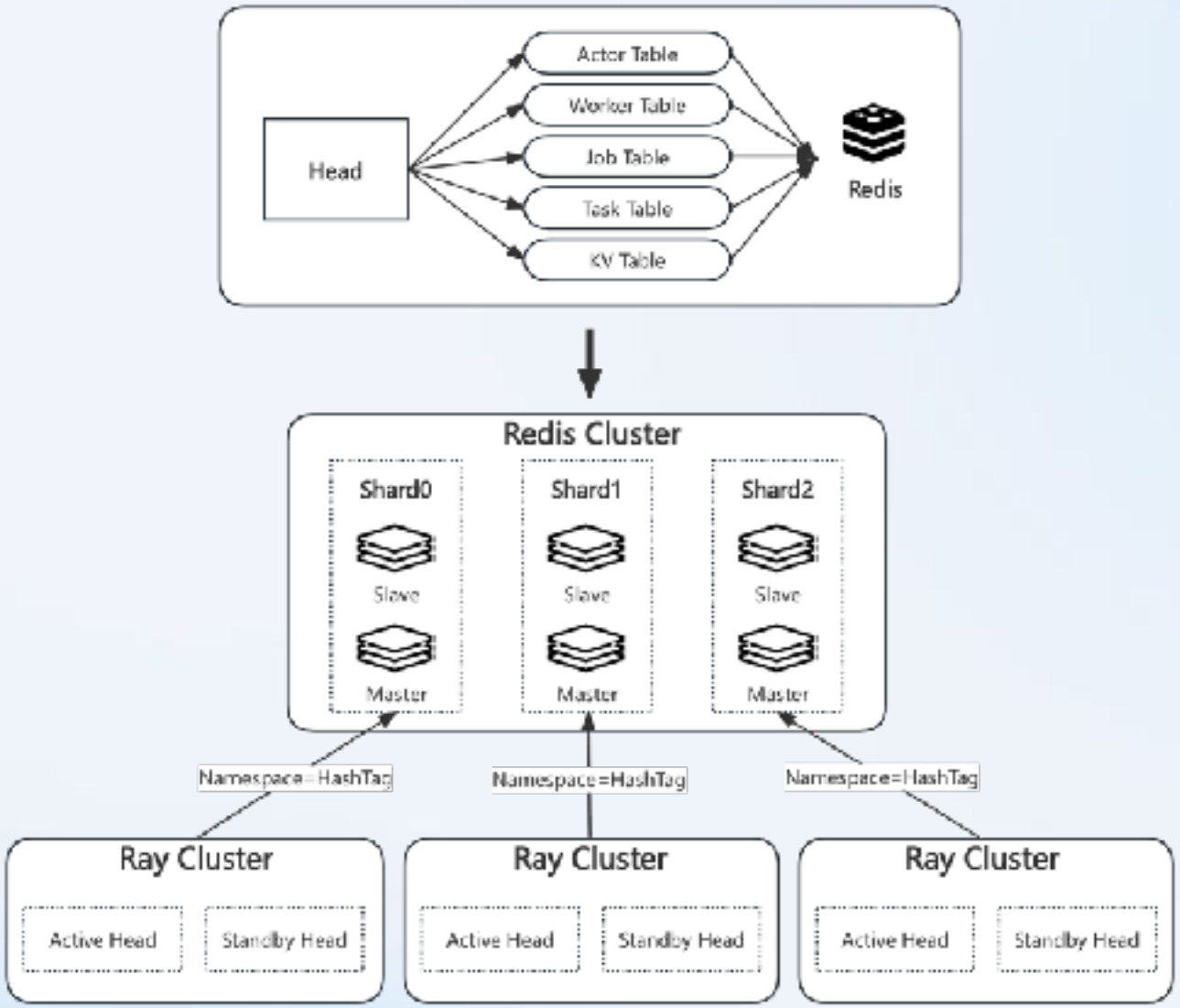
- Active和Standby争抢Token

容错增强

- 开启GCS On Redis
- Worker fetch log断点续传
- Client Connection失败重连

支持Redis Cluster

- 状态持久化管理，引入HashTag
- 一套Ray集群状态信息Hash到同一Shard
- 局部Node故障，自动寻址Hash对应副本Node



03

Ray+veRL
强化学习训练优化

强化学习RL – 概述

定义

- 让智能体通过与环境的交互来学习
- 目标是找到一个能够最大化累积奖励的策略

多角色

- 智能体(Agent) – 马里奥
- 动作(Action) – 往前走/跳跃
- 状态(State) – S0->环境变化>S1
- 奖励(Reward) – 活着/吃了蘑菇变大
- 环境(Environment) – 超级马里奥游戏

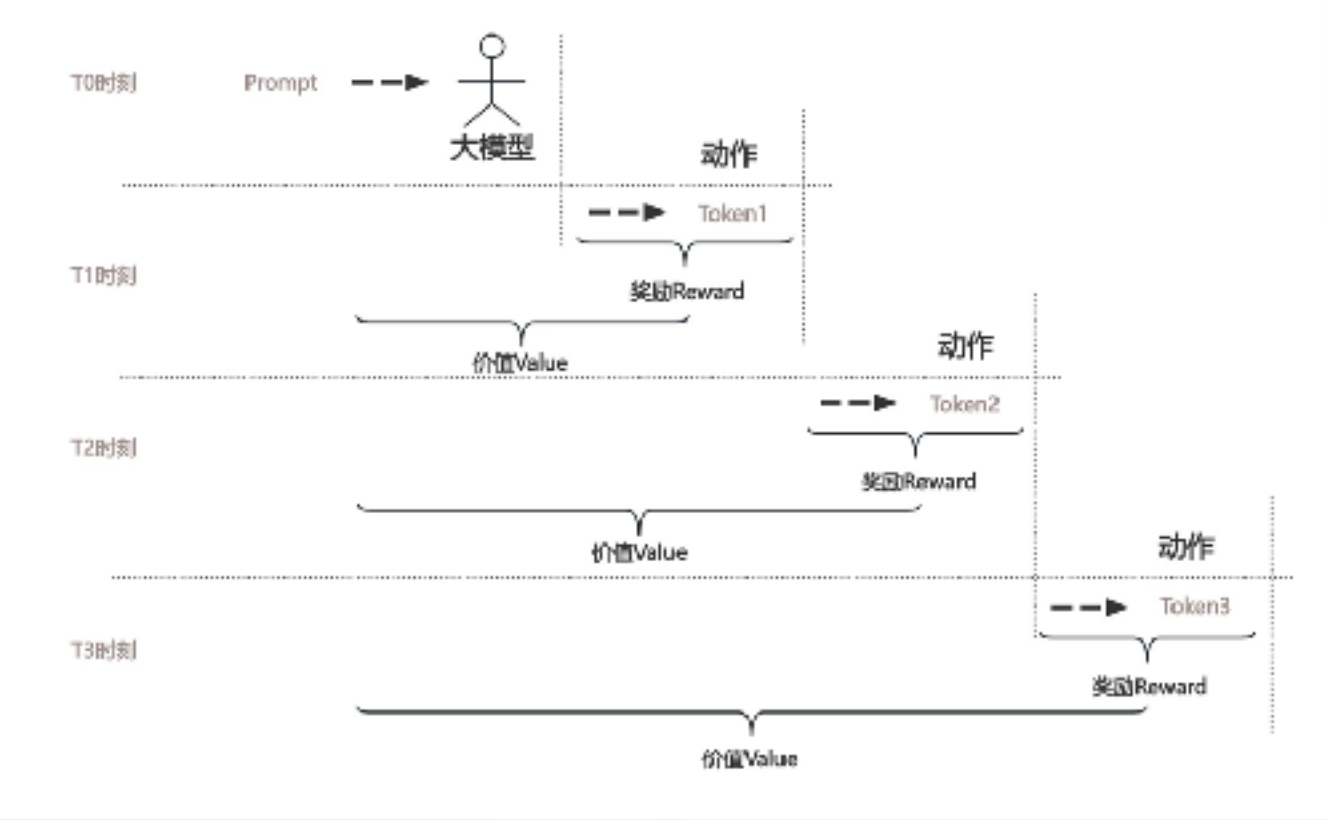
RL – 大模型训练

近端策略优化

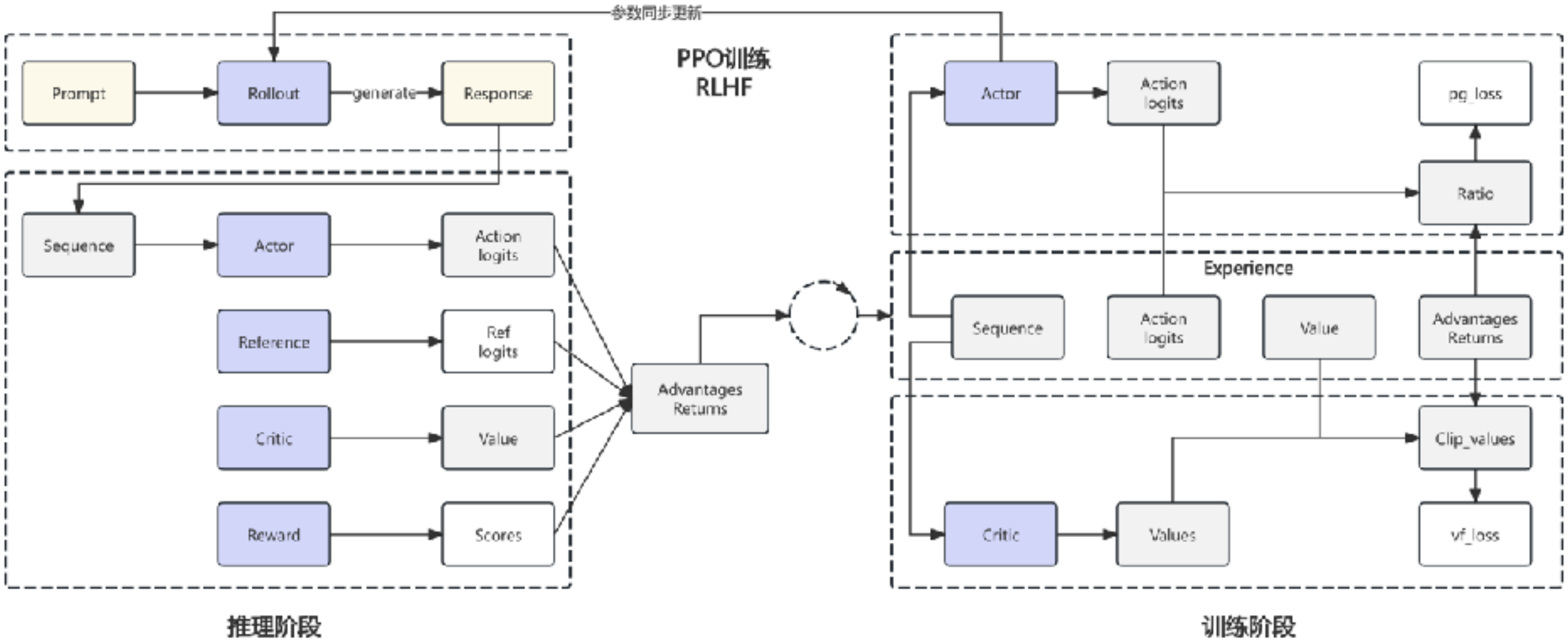
- PPO，OpenAI提出
- Proximal Policy Optimization
- 新旧策略差异不会太大下，寻找更优策略

角色说明

- Reward – 计算即时收益
- Value/Critic – 预估累计总价值
- Rollout – 生成动作，依赖于Actor
- Actor/Policy – 决策模型，训练对象
- Reference – 策略更新约束/重要性采样



RL – 训练流程


参数同步更新
PPO训练
RLHF
Prompt
Rollout
generate
Response
Sequence
Actor
Action logits
Reference
Ref logits
Critic
Value
Reward
Scores
Advantages Returns
Actor
Action logits
pg_loss
Ratio
Experience
Sequence
Action logits
Value
Advantages Returns
Clip_values
Critic
Values
vf_loss
推理阶段
训练阶段

veRL – 介绍

Controller

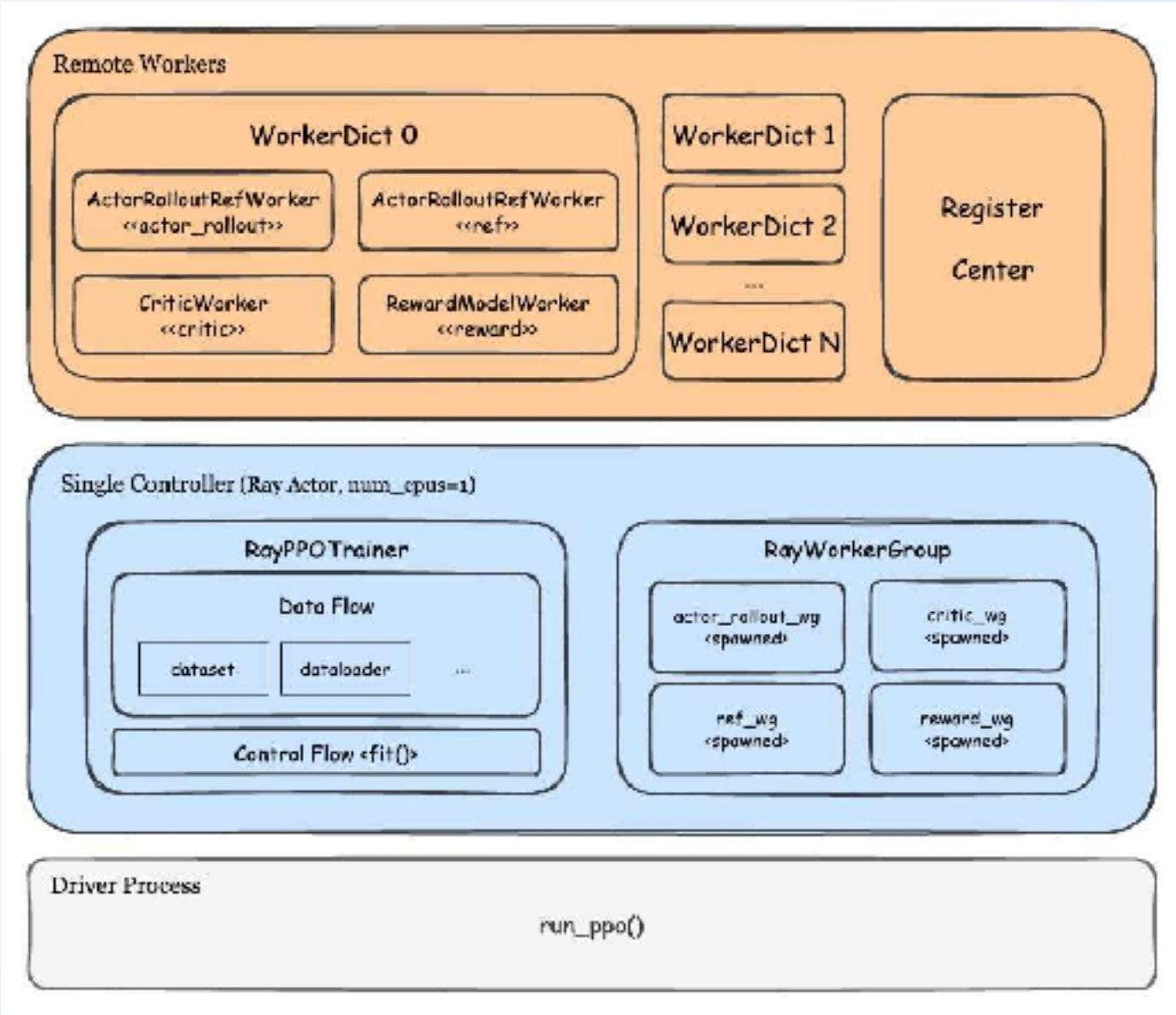
- Single+Multi
- 角色内，SPMD通信+组网
- 角色间，控制流基于Ray通信

生态融合

- 针对推理，集成vLLM+SGLang
- 针对训练，集成PyTorch+Megatron

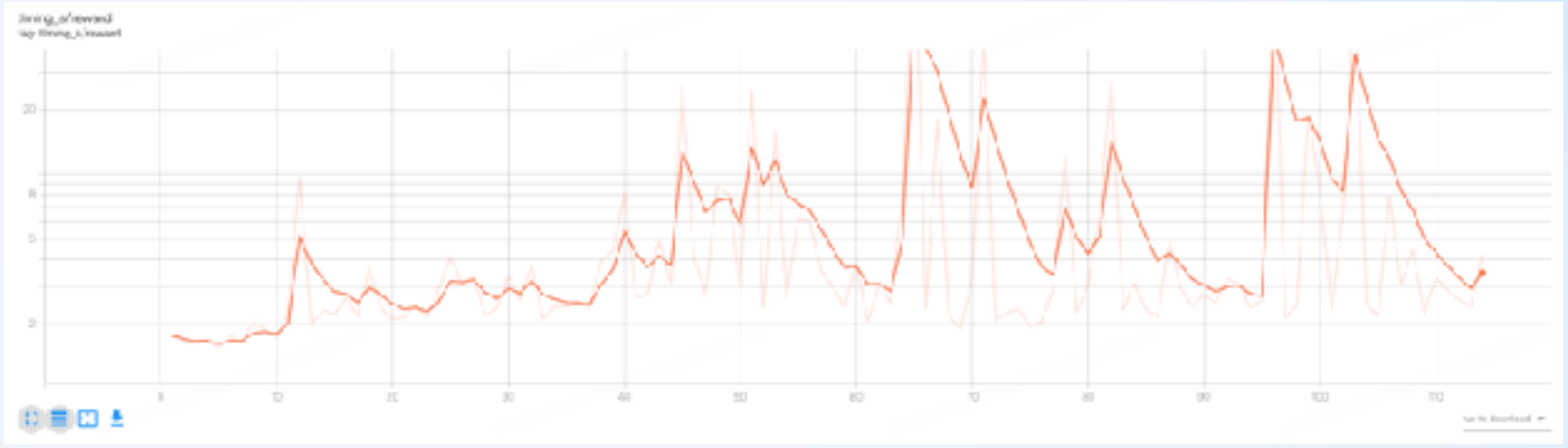
Hybird Engine

- Ray调度资源Pool，实现多角色共享

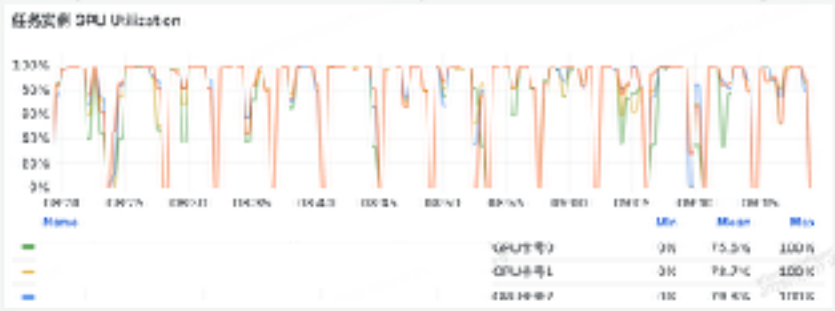


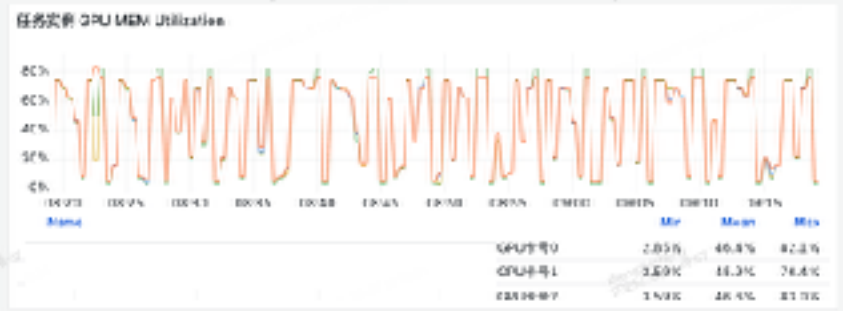
痛点 – 工程效率

Reward耗时高
导致GPU有空闲

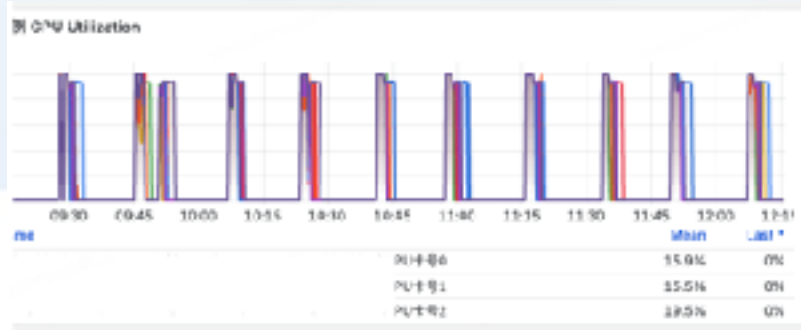


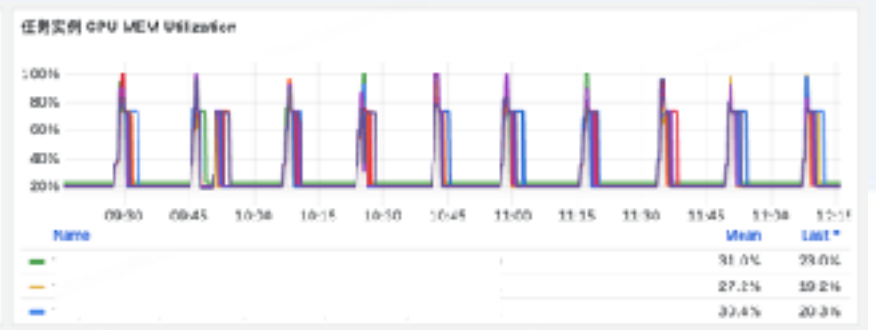
Rollout和Actor切换
GPU和显存利用率未打满



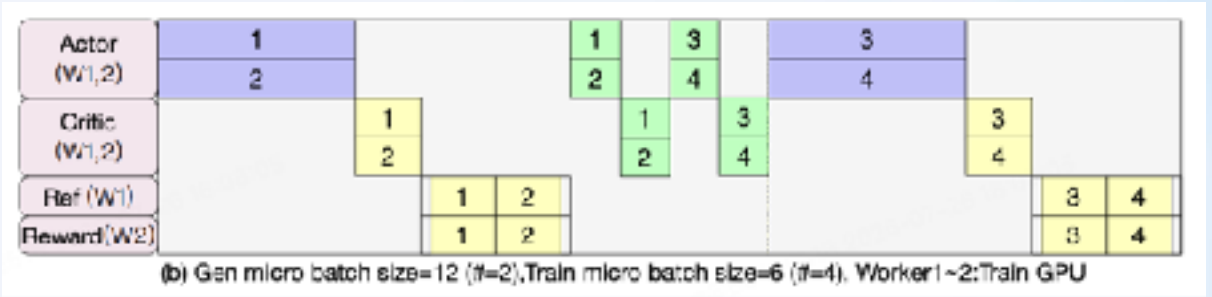


Reward Model On LLM
外挂资源翻倍，利用率减半





思路 – 异步+异构



目标

- 提升整体资源利用率，缩短训练时长

异构

- 内部很多异构卡，合理资源使用
- 训推资源差异大，如训练A100，推理A10
- 训练(权重/优化器状态/梯度)、推理(权重/kvCache)

异步

- 借鉴异步训练论文，[论文Paper]
- Rollout阶段和训练阶段并行解耦，类似Pipeline Parallel

An Adaptive Placement and Parallelism Framework for Accelerating RLHF Training

内容来自左侧Paper，出自Ant Group, Hangzhou, China

架构 – 异步流程

架构扩展

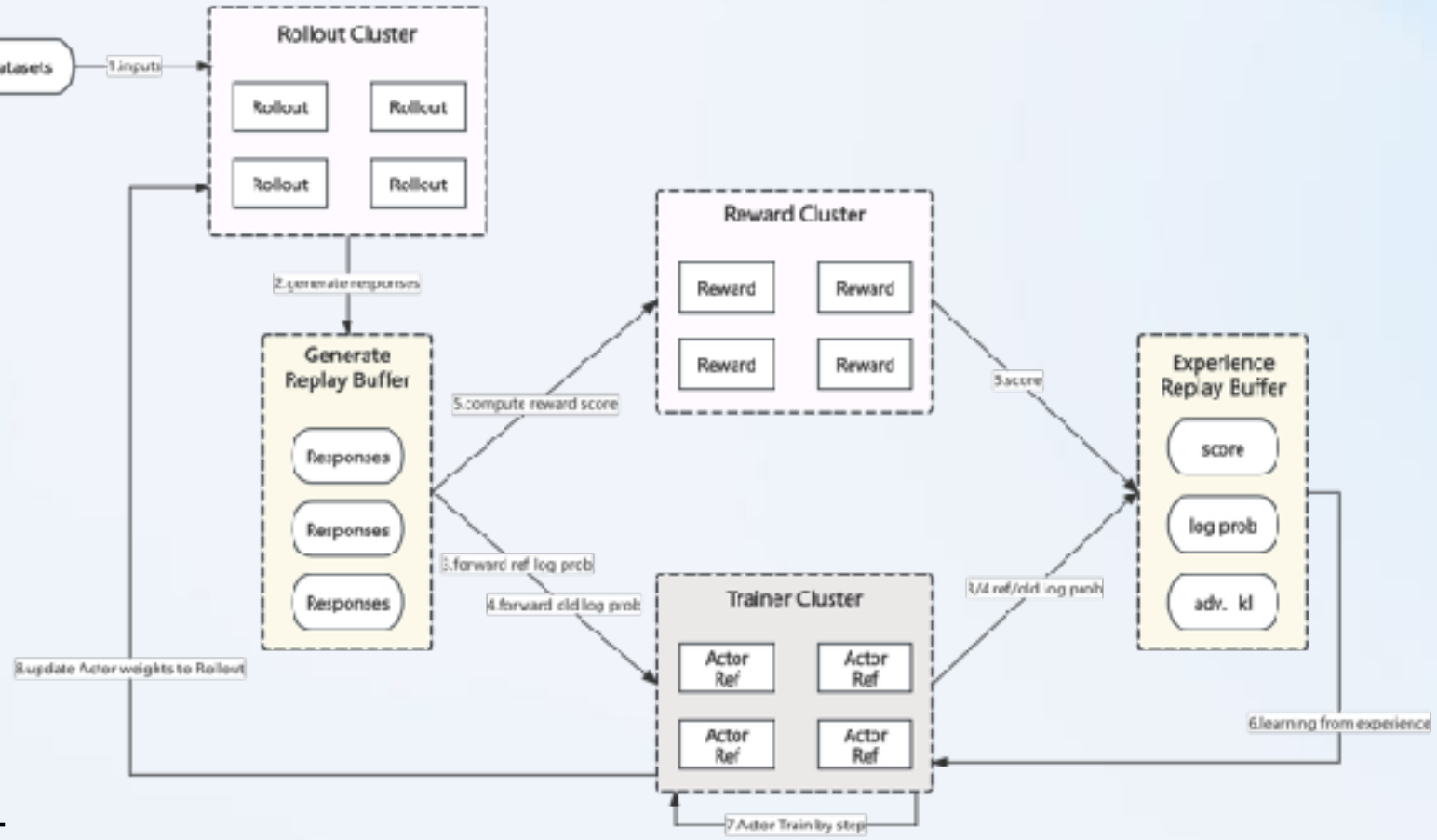
- 独立Rollout、Actor角色

Buffer队列

- 缓冲Generation和Experience

Async Flow

- Rollout基于Buffer Size反压
- Train异步从Buffer获取数据训练
- 如Buffer Size=1，就是原生的训练流程



架构 – 通信优化

权重同步

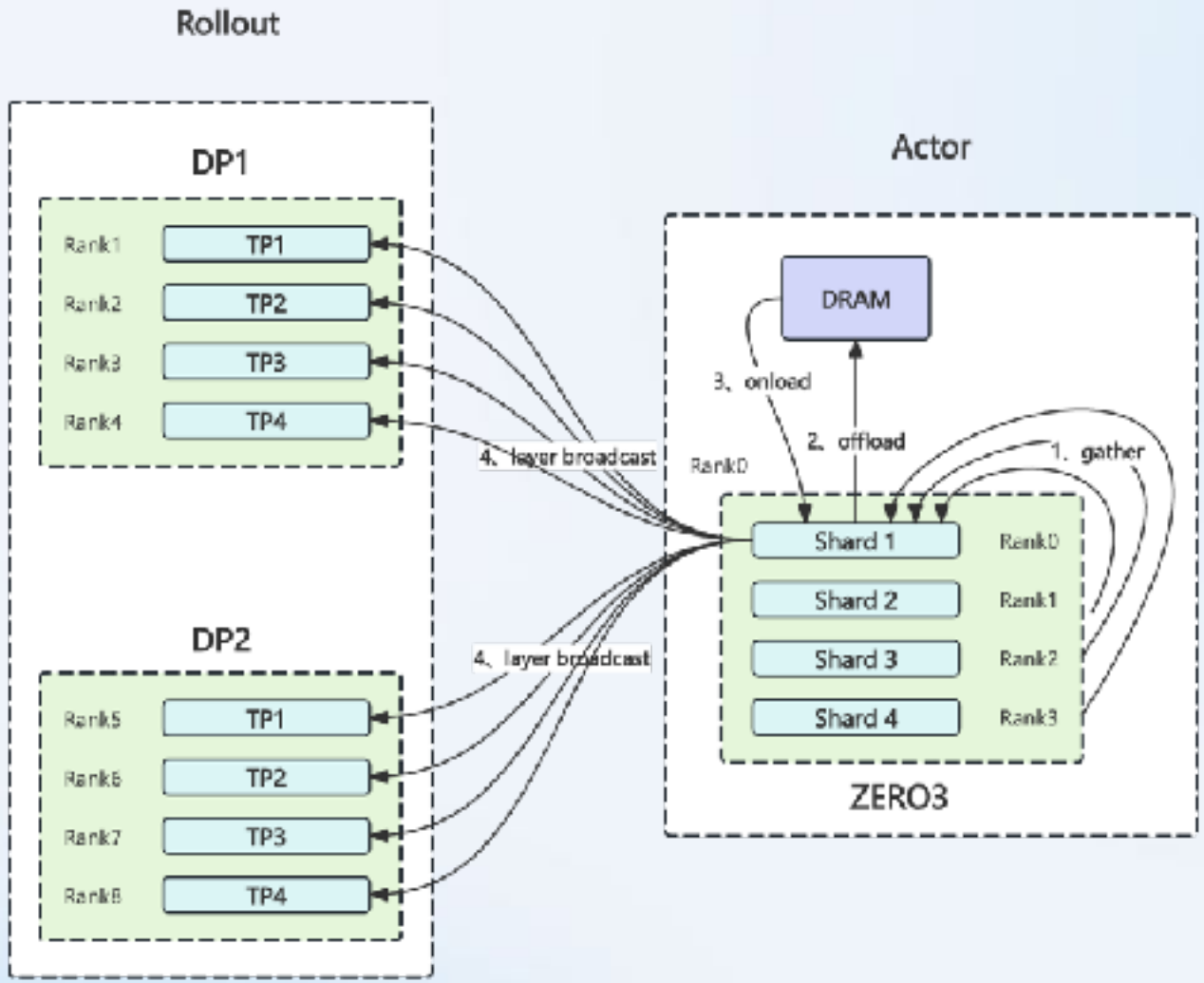
- Actor训练后权重同步到Rollout
- Rollout更新权重进行下一轮的推理

关键点

- Vllm/SGLang不支持暂停， 权重暂存
- Actor资源规格大于Rollout，权重卸载Actor侧
- Rollout完成Next Batch，触发权重的同步更新

工程效率

- CUDA Stream，通信流和计算流隔离
- NCCL同步权重按Layer，也可进一步Chunk切分



架构 – Pipeline

TBO

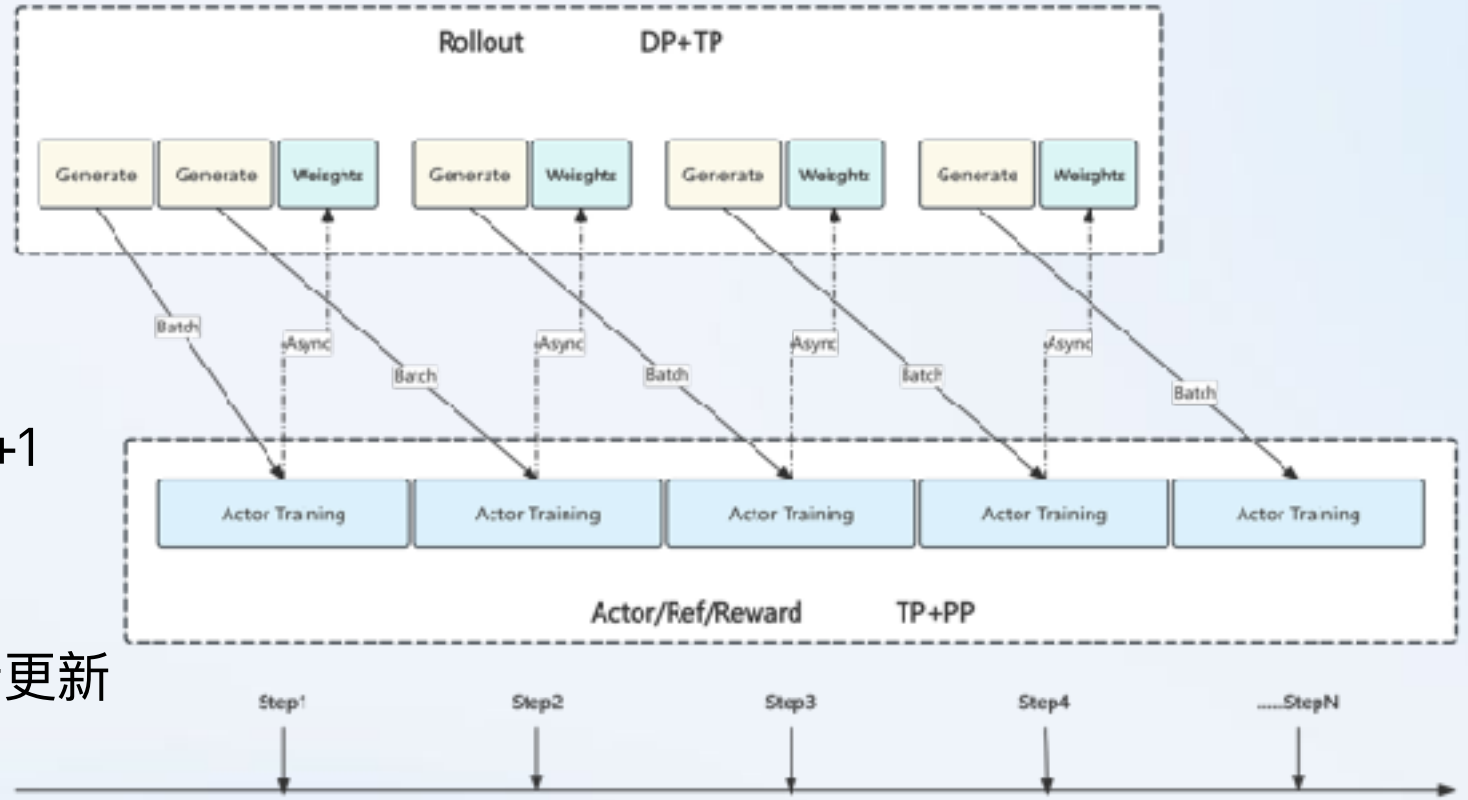
- Two Batch Overlap

Overlap

- Buffer Size = 2
- Actor训练Batch N，Rollout生成N+1

公式化

- Actor训练>=Rollout推理+权重同步更新



效果 – 工程效率ROI

部署

- Qwen2–7B–Instruct
- 训练4 A100，推理2*4 A10

效果

- 图2优化后Actor，图3优化后Rollout
- 显著缩短训练耗时，整体提升50%–100%

后续的扩展

- 弹性Rollout集群，进一步加速训练的效率
- Reward Model On Vllm，Colocate Actor/Ref

04

总结与展望

Ray – 多模态计算

数据IO成本

- 列式懒计算，减少不必要传输
- DAG的节点资源编排，减少Shuffle和IO
- 数据的IO成本，如多个Map节点的数据Moving

计算效率

- 自适应弹性计算，针对各个Map阶段，给出最合理并发配置
- 更细粒度算子的Checkpoint和Recovery，如视频分片或帧粒度恢复

稳定性

- 可观测增强、Worker节点的平滑升级、多租户（如Stage粒度镜像）、隔离性

更多场景 – 扩展

RL场景

- 基于VeRL+Ray，支撑Agent/MLLM RL
- 支持Reward Model on VLLM，进一步提升MFU

LLM Serve

- 基于Ray Serve，支持大模型的服务化
- 支持更多分布式推理模式，如P&D、EP、Attention–DP

多模态训练

- 基于Ray Data覆盖更多训练链路，并引入Lance解决向量/模态数据的存储

QCon
全球软件开发大会

InfoQ 极客传媒

THANKS

大模型正在重新定义软件

Large Language Model Is Redefining The Software