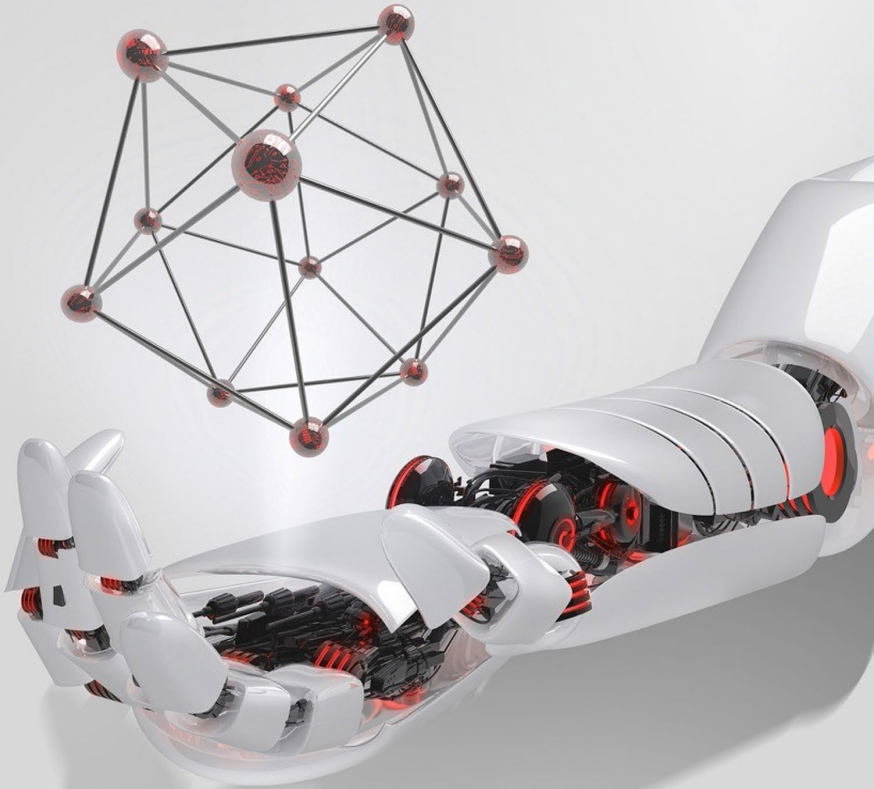
Data+AI融合场景下的分布式引擎探索与实践

演讲人：李志方

腾讯大数据基架高级研发工程师



01 AI时代下传统大数据引擎面临的挑战

02 基于PyIceberg的单机数据处理

03 引入Ray实现分布式数据处理

04 未来计划与展望

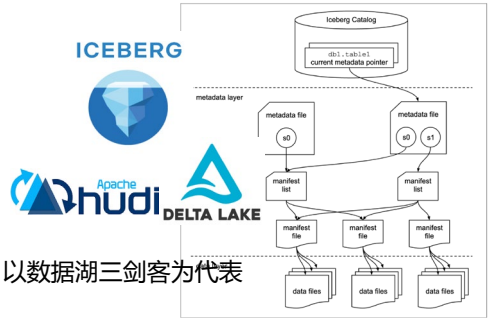
AI成为大数据的下一个发展热点

- **离线湖仓：T+1时效（03年~）**
  - MapReduce/Spark批处理引擎
  - Java/Scala作为主要语言
  - HDFS存储数据
  - 不支持ACID，静态schema

```
date=20180513/
 |- hour=18/
 |   |- ...
 |- hour=19/
 |   |- part-000.parquet
 |   |- ...
 |   |- part-031.parquet
 |- hour=20/
 |   |- ...
 |- ...
```

以Hive为典型代表

- **实时数仓：分钟级时效（18年~）**
  - 多引擎可插拔设计
    - 批处理：Spark
    - 流处理：Flink
    - 交互式查询：Starrocks/Pres
  - SQL作为主要语言
  - 对象存储S3/GCS/COS/OSS/MinIO
  - 支持ACID和schema evalution



以数据湖三剑客为代表

大模型等AI技术飞速发展（23年~），Data+AI融合场景成为未来趋势

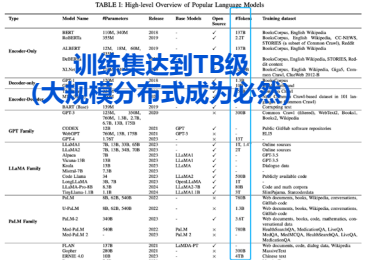
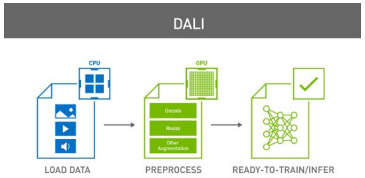


TABLE I: High-level Overview of Popular Language Models

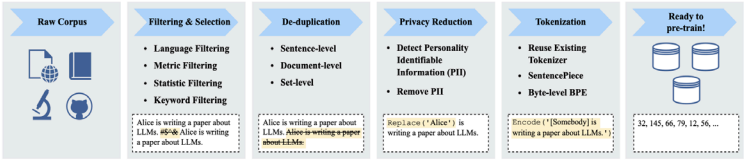
训练集达到TB级（大规模分布式成为必然）

CPU-GPU异构成为基本架构



https://developer.nvidia.com/zh-cn/blog/rapid-data-pre-processing-with-nvidia-dali/

pre-train前需要经历复杂的数据预处理流程



AI框架基本都是Python（传统大数据为Java/SQL）

Minaee S, Mikolov T, Nikzad N, et al. Large language models: A survey[J]. arXiv preprint arXiv:2402.06196, 2024.

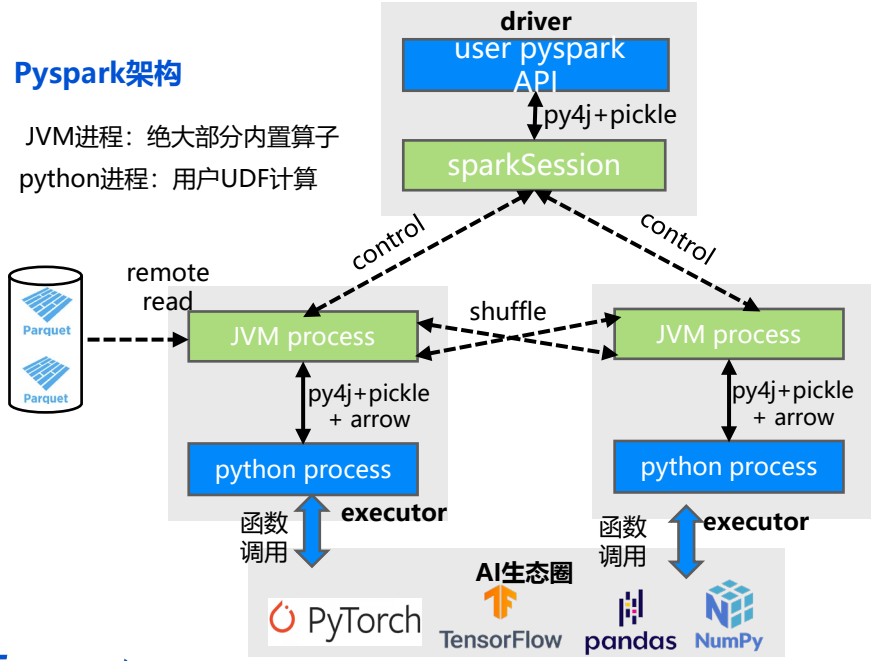
现有大数据引擎难以满足Data+AI融合场景

➢ 痛点1：Python交互能力受限

- 大数据引擎主要基于JVM生态
- 难以和AI生态无缝交互

Pyspark架构

JVM进程：绝大部分内置算子
python进程：用户UDF计算

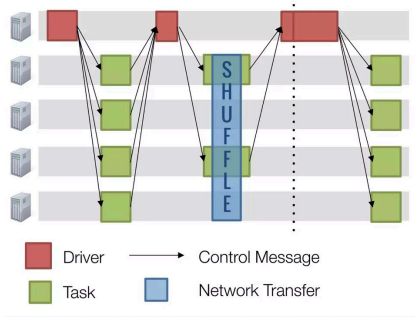


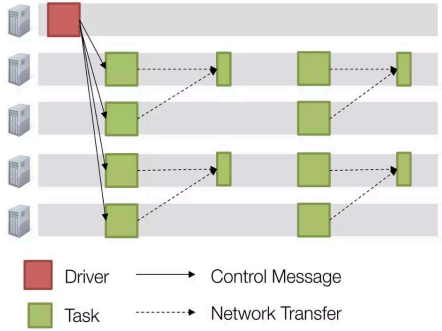
➢ 痛点2：不支持细粒度+异构的计算调度

- 大数据引擎只支持BSP和MPP/Dataflow两种范式
- Data+AI场景需要细粒度地使用CPU和GPU资源

BSP架构（Spark为例）

Stage-by-Stage方式，CPU预处理和GPU推理难以流水并行（（内部离线业务常用架构））





Dataflow架构（Flink为例）

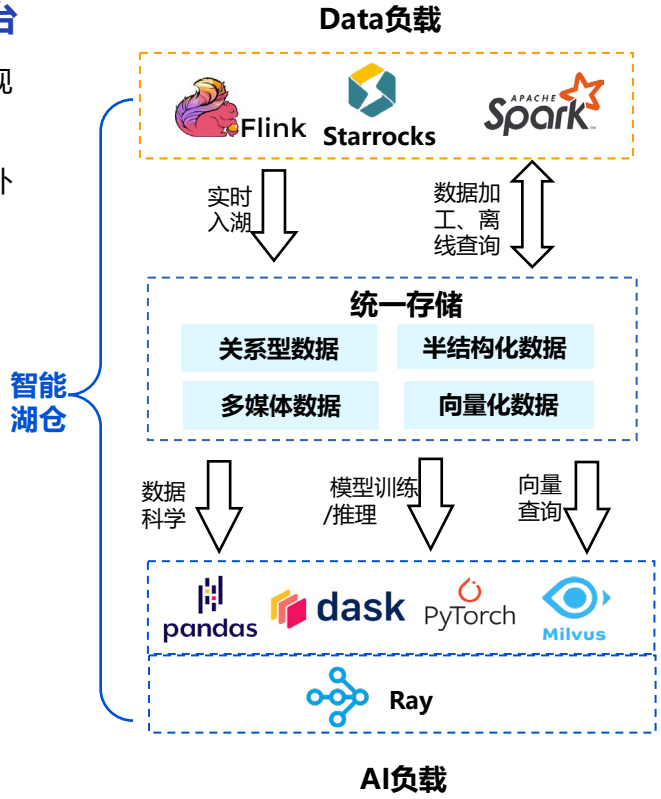
Record-by-Record方式，可以并行CPU和GPU，但无法动态变更DAG结构（内部在线业务常用架构）

Drizzle—Low Latency Execution for Apache Spark: Spark Summit East talk by Shivaram Venkataraman

智能湖仓成为Data+AI场景的重要基石

➢ 智能湖仓：Data+AI统一平台

- 无缝继承现有大数据框架，实现AI数据管理
- AI框架直接访问数据，无需额外ETL过程



实践与探索路线

➢ 阶段1：融入Python生态
- PyIceberg
- toTorch/toTF/toRay

➢ 阶段2：通过Ray实现分布式处理
- Ray Data Iceberg Source
- Dask/dask-ml on Ray

➢ 阶段3：打磨细分场景，优化性能
- Notebook on Ray
- 因果推断 on Ray
- Native dataframe on Ray

01 AI时代下传统大数据引擎面临的挑战

02 基于PyIceberg的单机数据处理

03 引入Ray实现分布式数据处理

04 未来计划与展望

Python原生的湖仓访问能力：Pyiceberg

优势

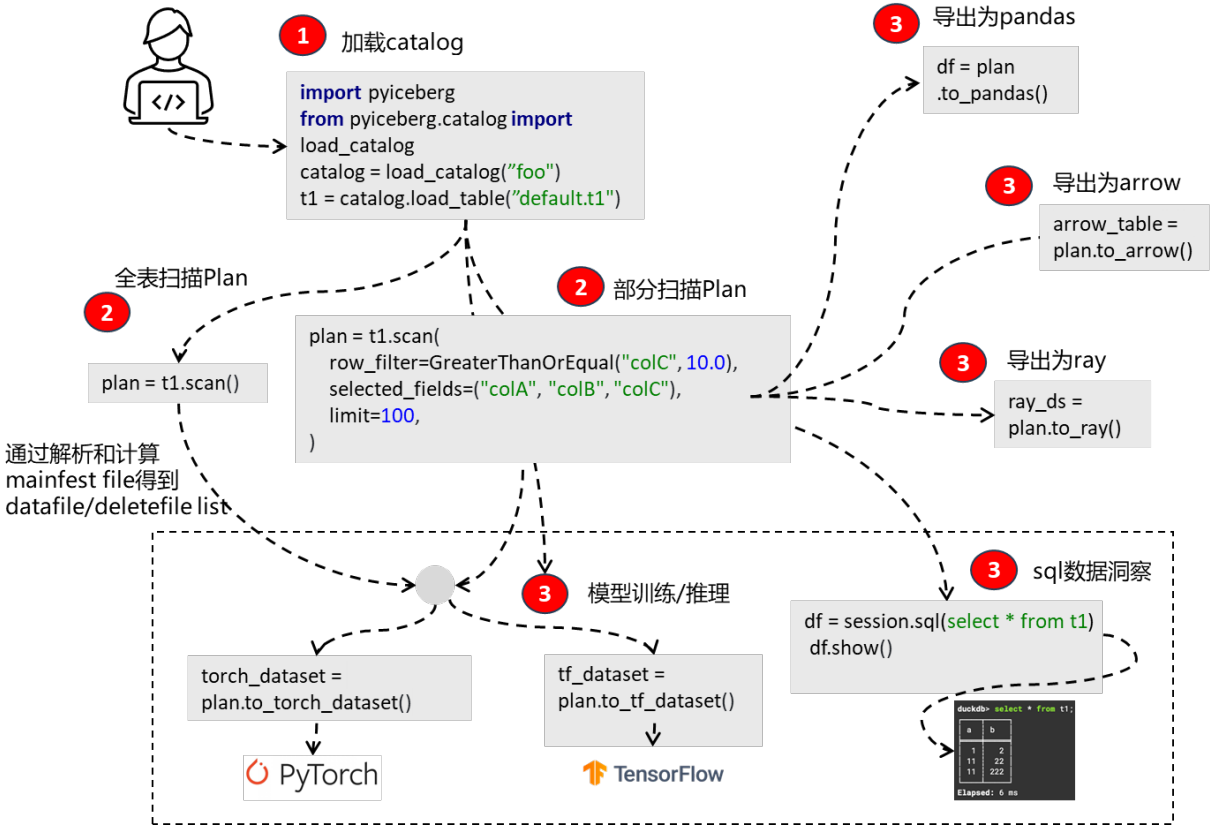
- 纯python接口，无JVM依赖
- 使用Arrow内存，天然亲和AI框架

不足

- 没有内置的分布式支持
- 单机lib难以服务化/平台化

自研特性

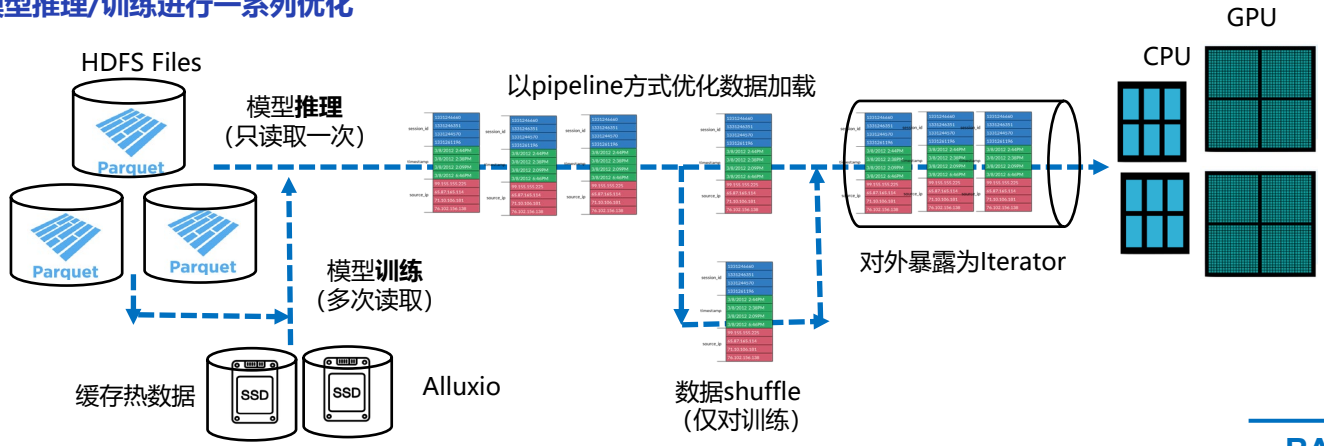
- HDFS支持（基于libhdfs）
- toPytorch/toTF loader
- 轻量级sql接口



训练/推理中的dataloader优化

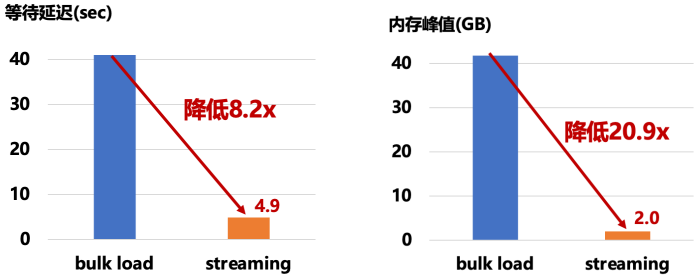
社区没有实现数据预处理和GPU计算的pipeline并行

内部实现为AI模型推理/训练进行一系列优化



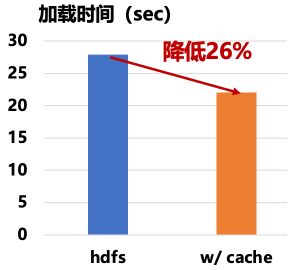
DataLoader性能测评

wikipedia-2022数据测试集



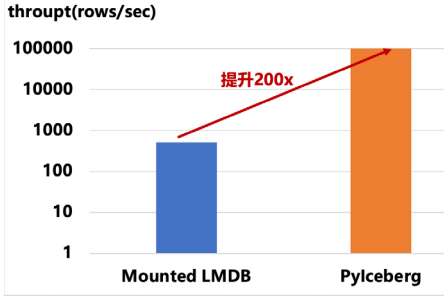
Dataloader的性能提升

配置：40 cores 256G mem

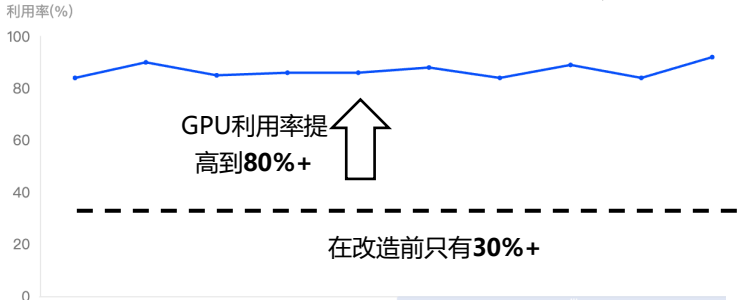


引入缓存的优化效果

某内部AI业务改造（离线推理）



batch size为64



数据链路准备时间从**2周**缩短到**3天**，
可以**快速迭代**模型效果

01 AI时代下传统大数据引擎面临的挑战

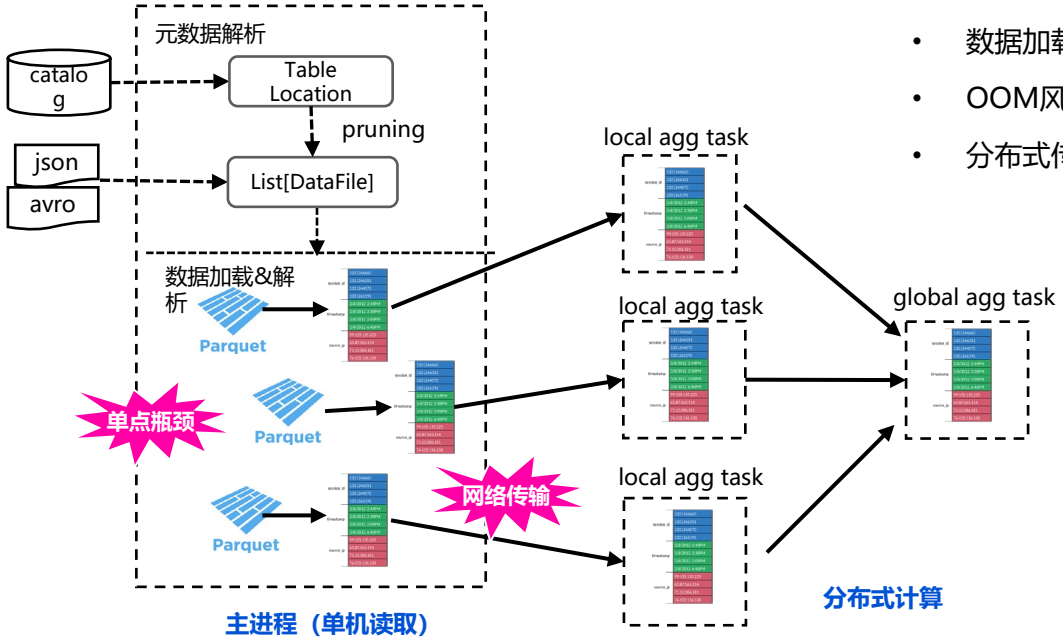
02 基于PyIceberg的单机数据处理

03 引入Ray实现分布式数据处理

04 未来计划与展望

PyIceberg toRay：单机读取，分布式计算

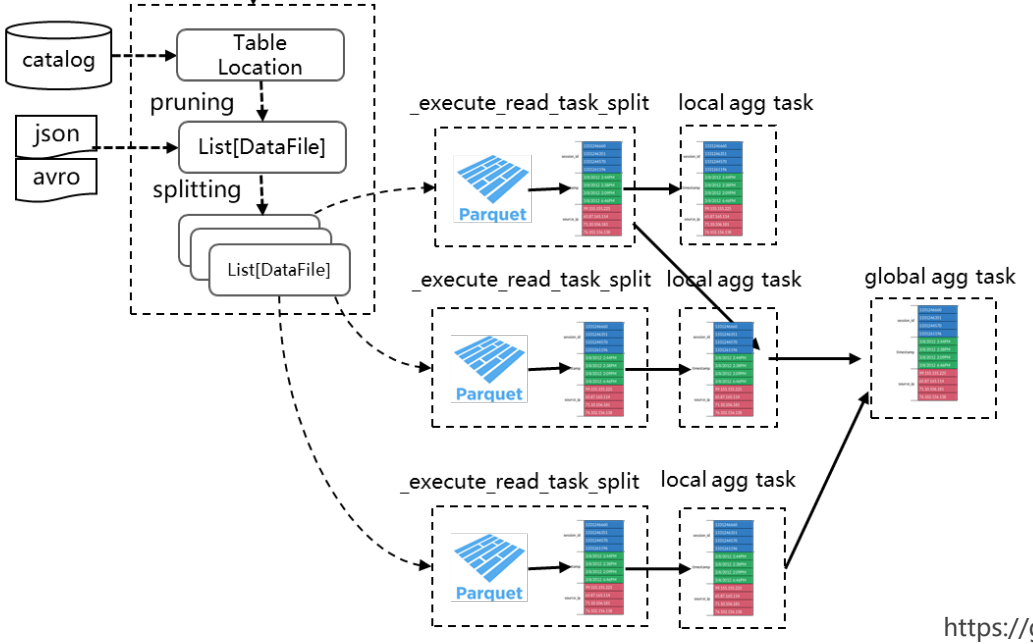
```
t1 = catalog.load_table(”default.t1")
plan = t1.scan(
   row_filter=GreaterThanOrEqual("colC", 10.0),
   selected_fields=("colA", "colB")
)
ds = plan.to_ray()
ds.groupby("colA").max("colB")
```



- **基于ray.data.from_arrow(xx)实现**
  - 数据需要在driver中物化成arrow格式
- **未充分发挥Ray分布式能力**
  - 数据加载单点瓶颈（CPU和网络）
  - OOM风险大
  - 分布式传输开销大

Ray Data IcebergSource：分布式读取+分布式计算

```
ds = ray.data.read_iceberg(
    table_identifier="default.t1",
    row_filter=GreaterThanOrEqual("colC", 10.0)
)
ds.groupby("colA").max("colB")
```



- **基于Ray Data的全分布式化**
  - planning和scanning解耦到不同task
  - DataFile列表split成片段（按file_size）

https://github.com/ray-project/ray/pull/46889

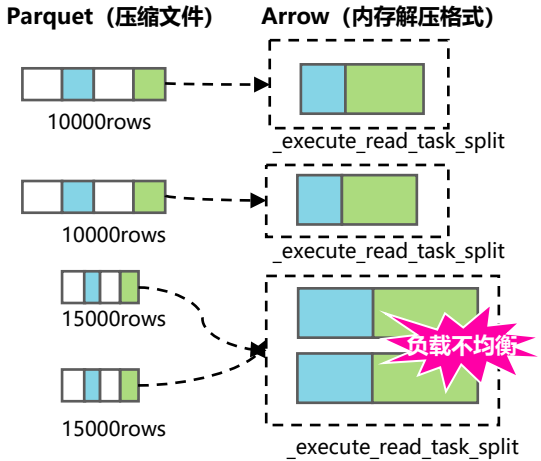
内部版Ray Data IcebergSource：支持大规模数据集

➢ 性能优化：自适应spiltting

- 社区版基于压缩后的文件size，未考虑解压后的内存大小（列存的压缩比和内容有关），容易负载不均衡
- 内部版基于read_schema估算解压后的内存大小，更接近最优的分布

➢ 线上bug修复

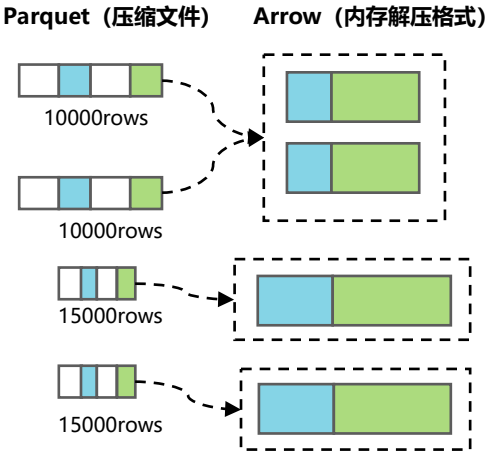
- 修复read task内部存在部分未解耦的元数据访问
- 修复read task内部存在CPU超用（PyIceberg底层的并行读取）



按预计解压后的memory size

inmemory_size += int(infer_row_size(schema) * total_row_count)

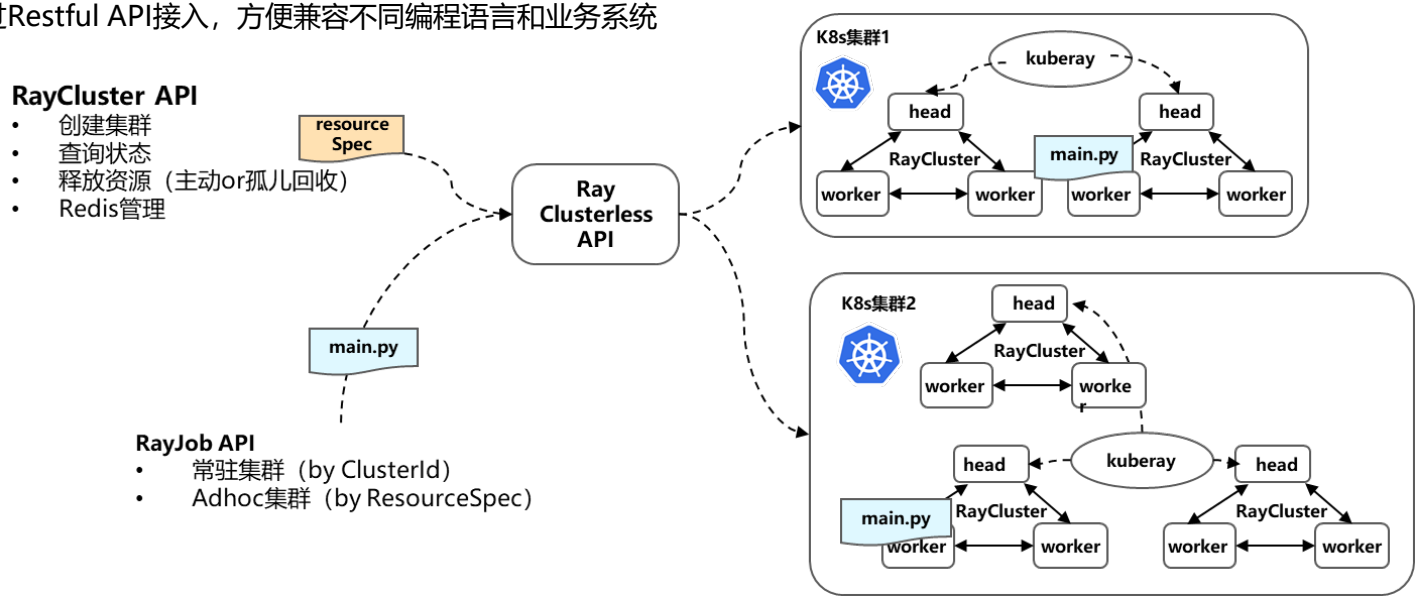
注：string, map, list 等变长类型只能假设一个固定长度



Ray Clusterless API：更易用的RayCluster管理/RayJob提交入口

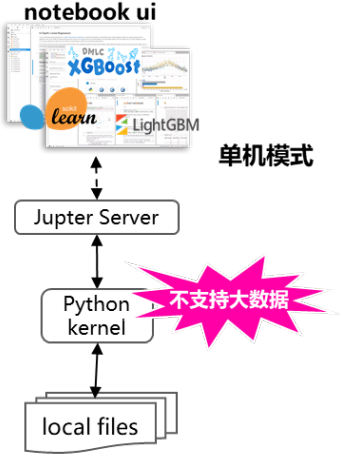
➢ Ray Clusterless API

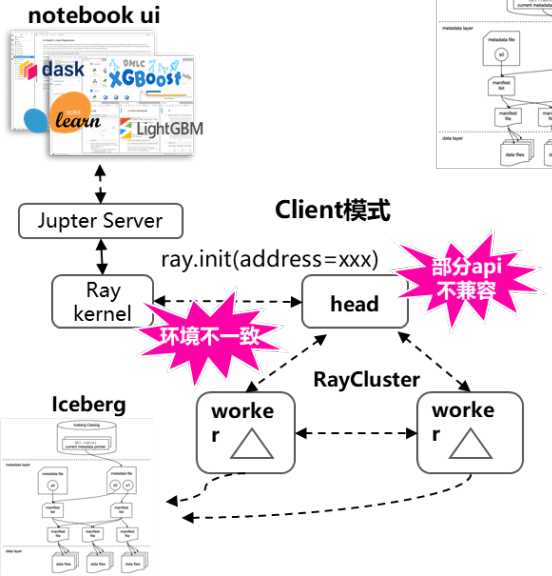
- 用户无需感知/维护k8s集群（kuberay需要用户有一定k8s背景）
- 支持多k8s集群提交和管理
- 同时支持Cluster/Job接口（现有ray sdk仅支持job）
- 通过Restful API接入，方便兼容不同编程语言和业务系统

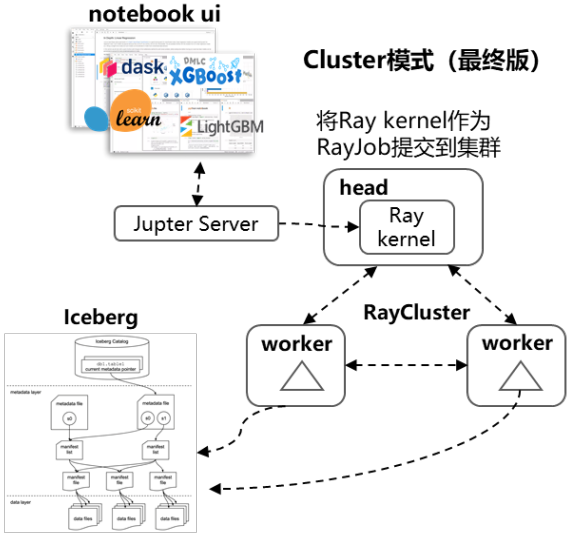


应用场景1：交互式数科分析 on Ray

- **基于Ray打造一站式数据科学Notebook**
  - Notebook kernel on ray（cluster模式）
    - 调试和运行环境保持一致
    - 快速启动（<10s）
  - 数科框架 on ray
    - 通过Ray Data分布式读取Iceberg海量数据，无需ETL准备
    - Dask/XGB/Sklearn on Ray实现分布式计算

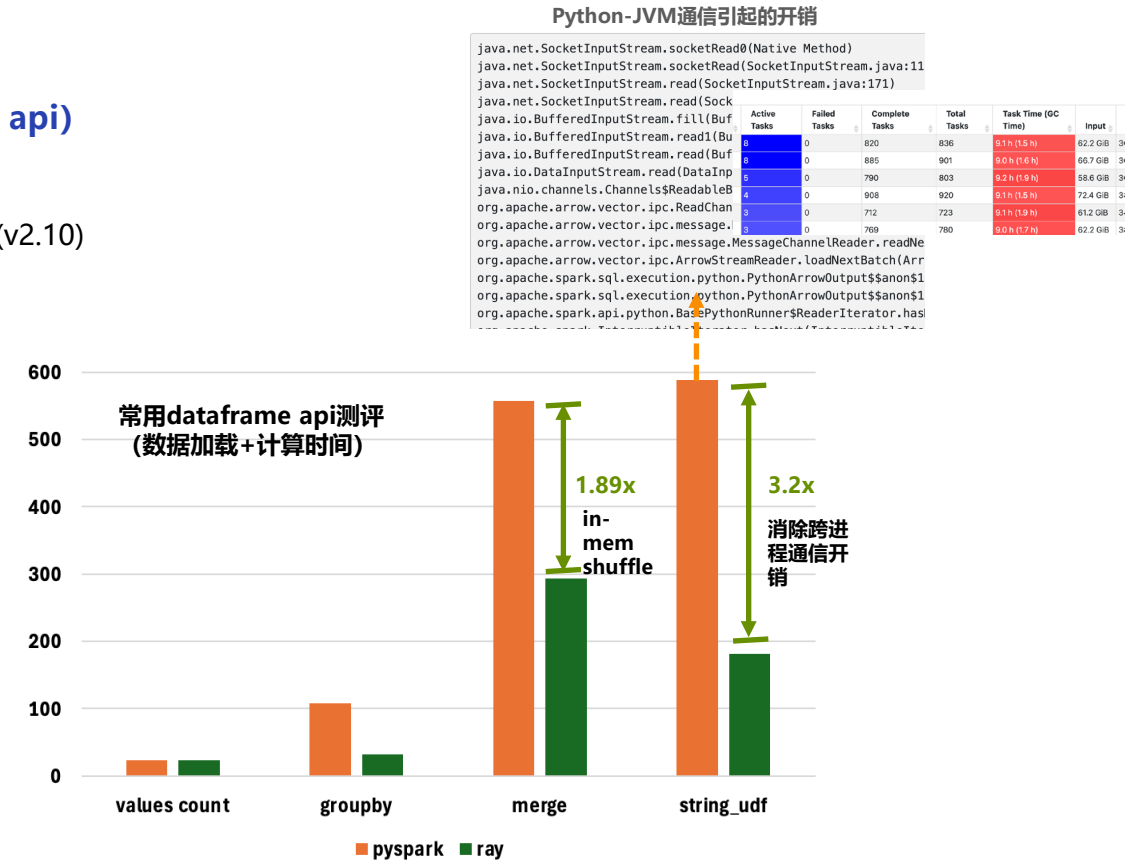






应用场景1：交互式数科分析 on Ray

- **线上性能测评（dataframe api）**
  - pyspark.pandas (v3.3)
  - ray data + dask on ray (v2.10)

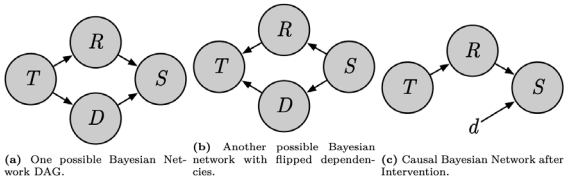


数据集：cudf测试集（100GB Iceberg存储，内存约1TB）
https://github.com/rapidsai/cudf/blob/branch-24.04/docs/cudf/source/user_guide/performance-comparisons/performance-comparisons.ipynb

资源配置：
分布式200核+750GB内存

应用场景2：因果推断 on Ray

因果推断是一种基于统计学中的方法，用于确定事件或变量之间的因果关系，属于机器学习的一个分支领域。在互联网中广泛应用于A/B测试、广告营销、用户行为分析等场景。



(a) One possible Bayesian Network DAG. (b) Another possible Bayesian network with flipped dependencies. (c) Causal Bayesian Network after Intervention.

Figure 2.1: DAGs of Bayesian and Causal Bayesian Networks.

Kaddour J, Lynch A, Liu Q, et al. Causal machine learning: A survey and open problems[J]. arXiv preprint arXiv:2206.15475, 2022.

业务痛点：主流因果推断框架主要是单机Python库，基于numpy/pandas计算，难以支持互联网海量数据，抽样又会减少效果

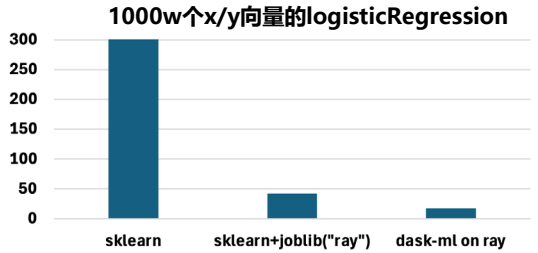
CausalML https://github.com/uber/causalml

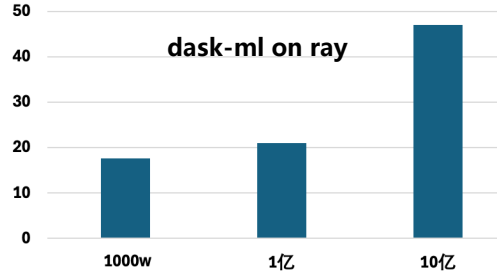
DoWhy https://github.com/py-why/dowhy

EconML https://github.com/py-why/EconML

➢ 通过Ray实现分布式因果推断算法

- 使用dask df/arrary (on ray)替换numpy/pandas接口
- 使用dask-ml (on ray)中成熟的经典机器学习算法





资源配置：4核24G + 8核32G*8

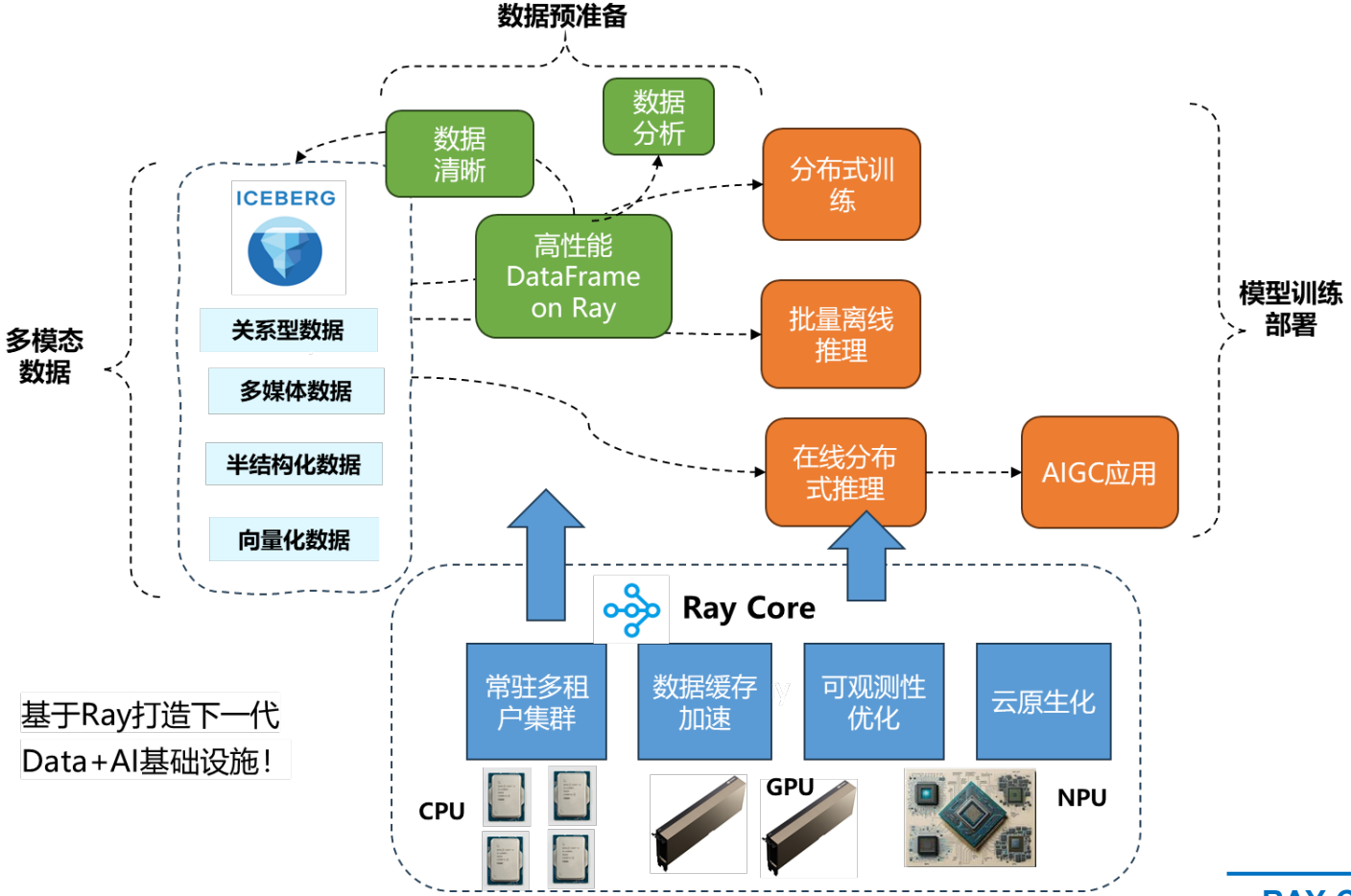
01 AI时代下传统大数据引擎面临的挑战

02 基于PyIceberg的单机数据处理

03 引入Ray实现分布式数据处理

04 未来计划与展望

未来计划与展望


数据预准备
数据清晰
数据分析
分布式训练
高性能DataFrame on Ray
批量离线推理
在线分布式推理
AIGC应用
模型训练部署
多模态数据
ICEBERG
关系型数据
多媒体数据
半结构化数据
向量化数据
Ray Core
常驻多租户集群
数据缓存加速
可观测性优化
云原生化
CPU
GPU
NPU
基于Ray打造下一代Data+AI基础设施!

We are hiring!

基于Ray打造下一代Data+AI基础设施

联系方式：zhifgli@gmail.com

非常感谢您的观看

Q & A