开源社
kaiyuanshe

Vineyard

加速大数据分析工作流中的跨引擎数据共享

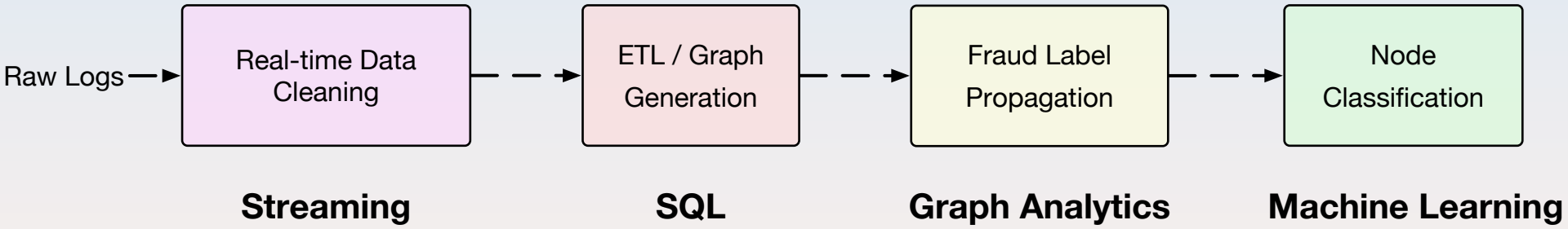
何涛，阿里巴巴

真实世界中的大数据分析

Vineyard

单个数据分析应用往往有很多个不同的任务组成

一个典型的风控场景

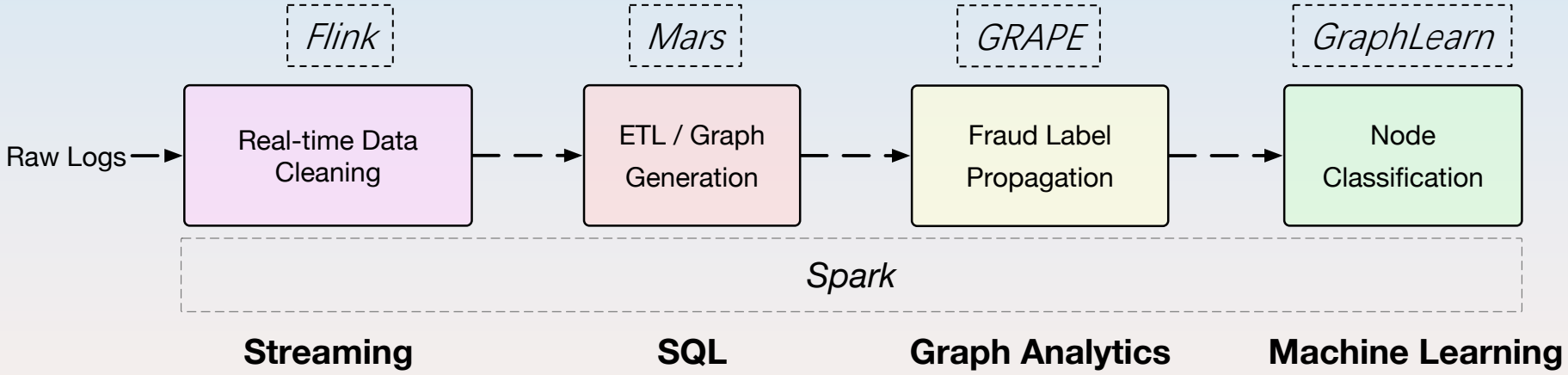


真实世界中的应用通常包含真多个跨领域的任务.

单系统 vs. 多系统

Vineyard

不同领域的计算任务通常使用不同的计算系统完成



很多任务都可以用Spark这样的系统完成，

但是，Spark对于很多数据分析的场景并不总是最好的选择

GRAPE[1] 在大规模图的分析算法上性能可以超出 Spark *200倍*.

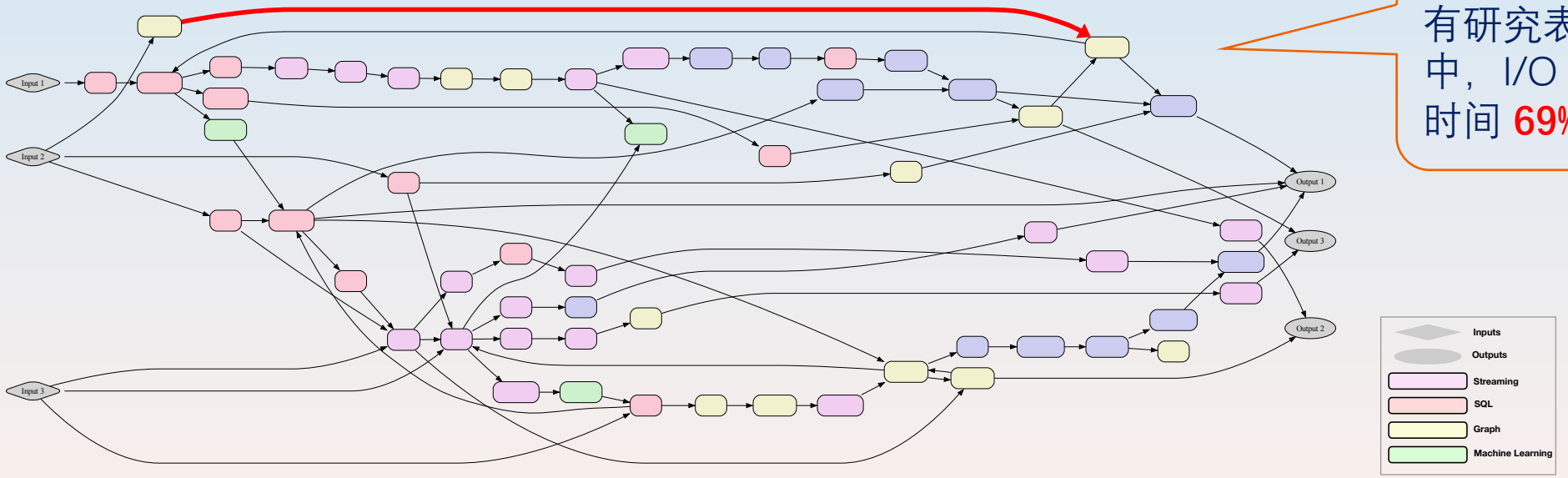
在生产实践中，人们往往更倾向于部署多个系统来应对复杂应用的需求

[1]: Parallelizing Sequential Graph Computations, Wenfei Fan et al., SIGMOD' 2017

多系统工作流中的中间数据共享

Vineyard

多系统数据分析工作流需要繁重的中间数据共享



有研究表明，大数据分析工作流中，I/O 阶段消耗了**79%** 的执行时间 **69%** 的计算资源。[1]

工作流中的任务往往是分布式的，实践中可以扩展上千个工作节点

Kubernetes Clusters

数据共享是数据密集型分析工作流中最大的额外开销的来源

[1]: PACMan: Coordinated Memory Caching for Parallel Jobs, Ganesh et al., 2012

工作流中低效的中间数据共享

Vineyard

中间数据的共享往往是通过在外部文件系统上以公共文件格式存取实现的

- 这个过程需要序列化/反序列化的开销，会造成很多额外的内存拷贝，需要文件I/O和网络传输
- 对于一些复杂类型的数据结构(例如 *属性图*) ，由于缺乏高效的文件存储格式，进一步放大了序列化/反序列化过程的开销（需要复杂的额外计算，e.g., Sort、Shuffle等等）

使用外部存储服务来共享任务之间的中间结果效率低、成本高

为什么不使用高效的内存？

Vineyard

使用内存来共享中间结果在实践中有很多挑战

事实上，我们已经很熟悉在同一个进程的 library 之间内存共享中间结果：传递一个变量

```
>>> a = numpy.array([1, 2, 3])
>>> t = torch.from_numpy(a)
>>> t
tensor([ 1,  2,  3])
>>> t[0] = -1
>>> a
array([-1,  2,  3])
```

但是，

- **跨进程/运行时，就不再容易做这样的事情**
  - 需要 M x N ad-hoc 的、专门的集成用来连接用到的计算库和计算引擎
- **缺乏对分布式数据和内存中放不下的数据的支持**
  - 使用内存时，需要仔细地设计和实现来应对远程数据访问、数据迁移、数据在内存和外存之间置入置出、调度等各种难题

如何设计一个系统，既能享受到内存的高效，同时又能满足大数据分析系统和应用带来的挑战?

Vineyard：中间数据的高效共享

Vineyard

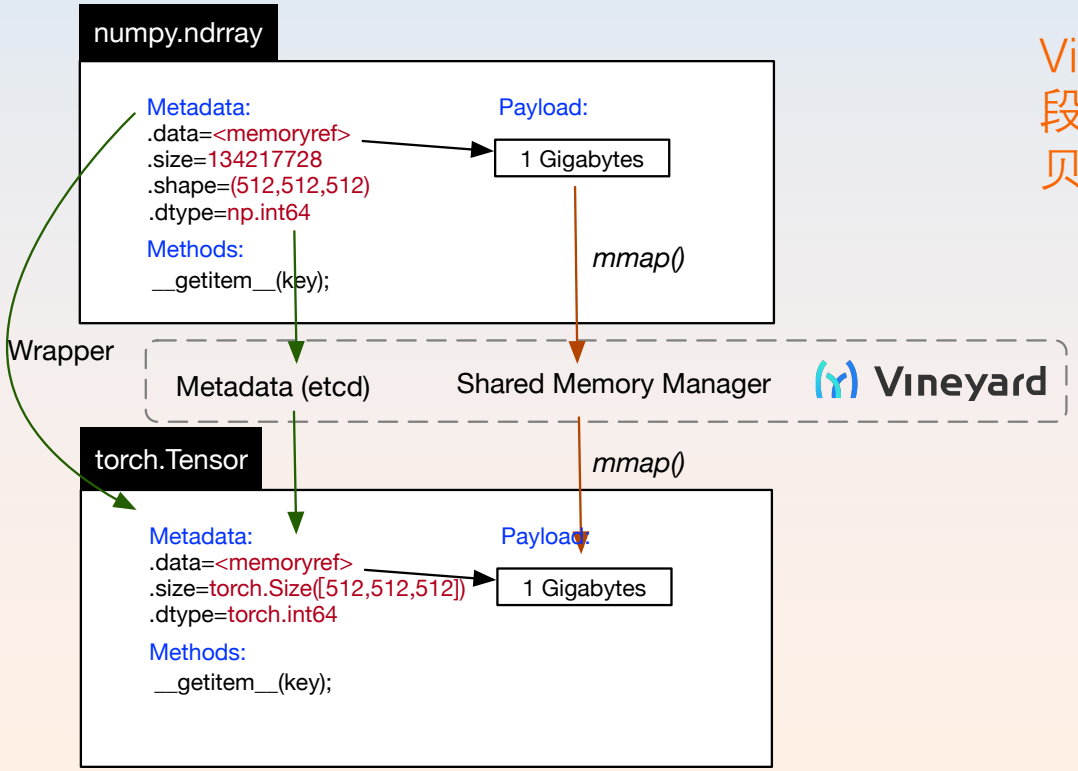
Vineyard 是一个为高效共享大数据分析工作流中任务之间中间结果传递设计的数据管理引擎

- **使用内存**：尽可能地避免不必要的内存拷贝、序列化/反序列化，文件I/O和网络传输；
- **容易被集成**： 基础计算引擎之间数据结构之间存在共性的观察，避免 *M x N 的集成开销；*
- **分布式**： 支持不同系统之间共享分布式数据和内存中放不下的数据；
- **云原生**：针对 Kubernetes 上部署大数据分析应用特别设计和优化。

Vineyard：抓住不同系统间数据结构的共性

如何使用内存在不同的系统之间共享数据？

机遇: 很多数据结构在不同的系统中有不同的实现，但是，**它们往往有着相同的内存布局**，而仅仅是在一些元数据/属性上有区别



Vineyard 将数据对象描述成元数据（metadata）和一组连续内存片段（payloads），使用内存映射（mmap）实现对 payloads 的零拷贝共享。对集成来说，只需要一个很直观的wrapper来重构对象。

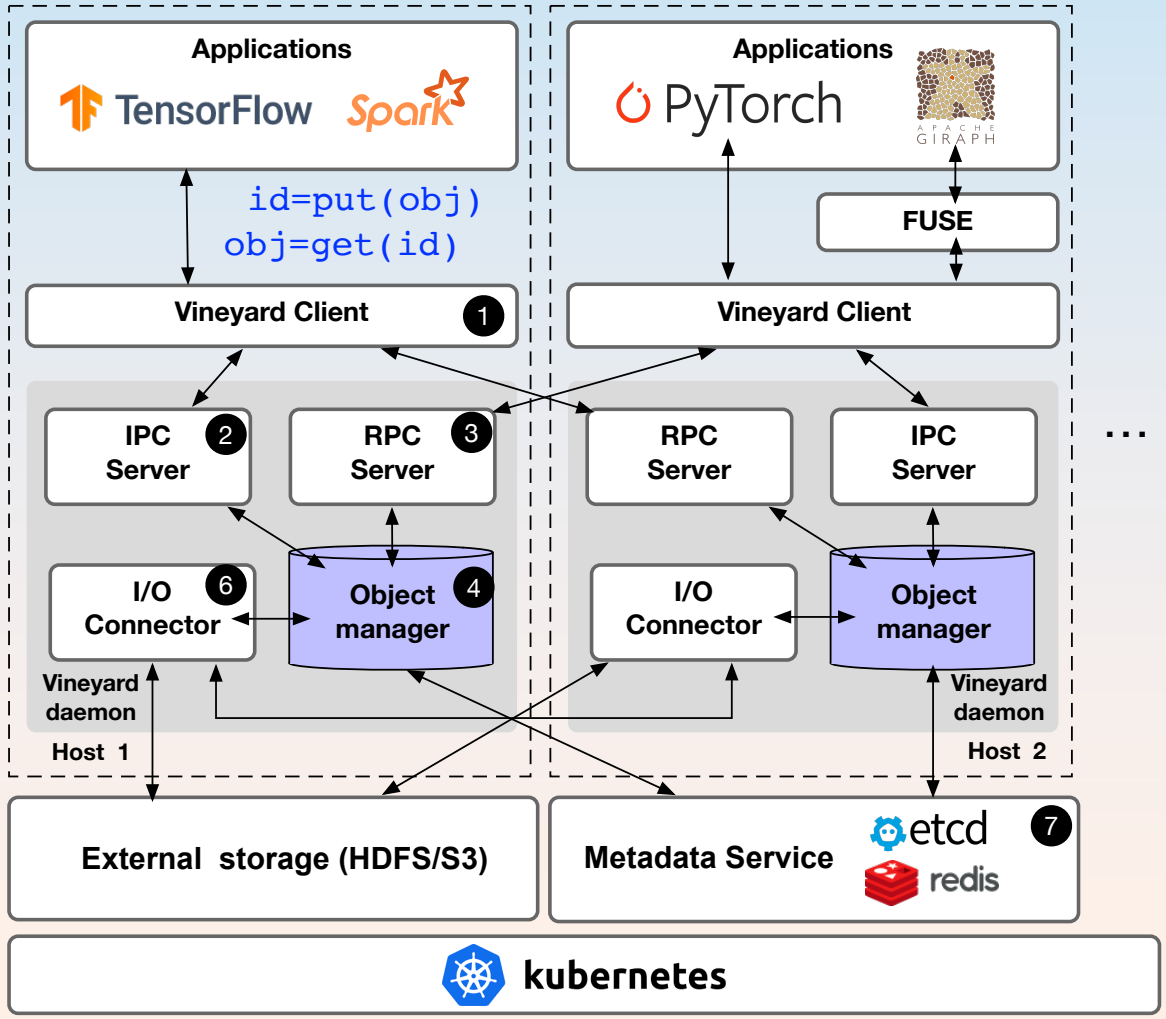
例如，“.data”字段可以在 PyTorch 和 NumPy 之间零拷贝地跨进程高效共享。

元数据被组织成支持嵌套的属性树，存在KV存储中，这一设计是的Vineyard中的对象具有可组合行，进一步减少了复杂对象需要的昂贵的序列化/反序列化。

Vineyard 实现了跨系统、跨进程的 payloads 的零拷贝高效共享

Vineyard：内存数据管理引擎

Vineyard

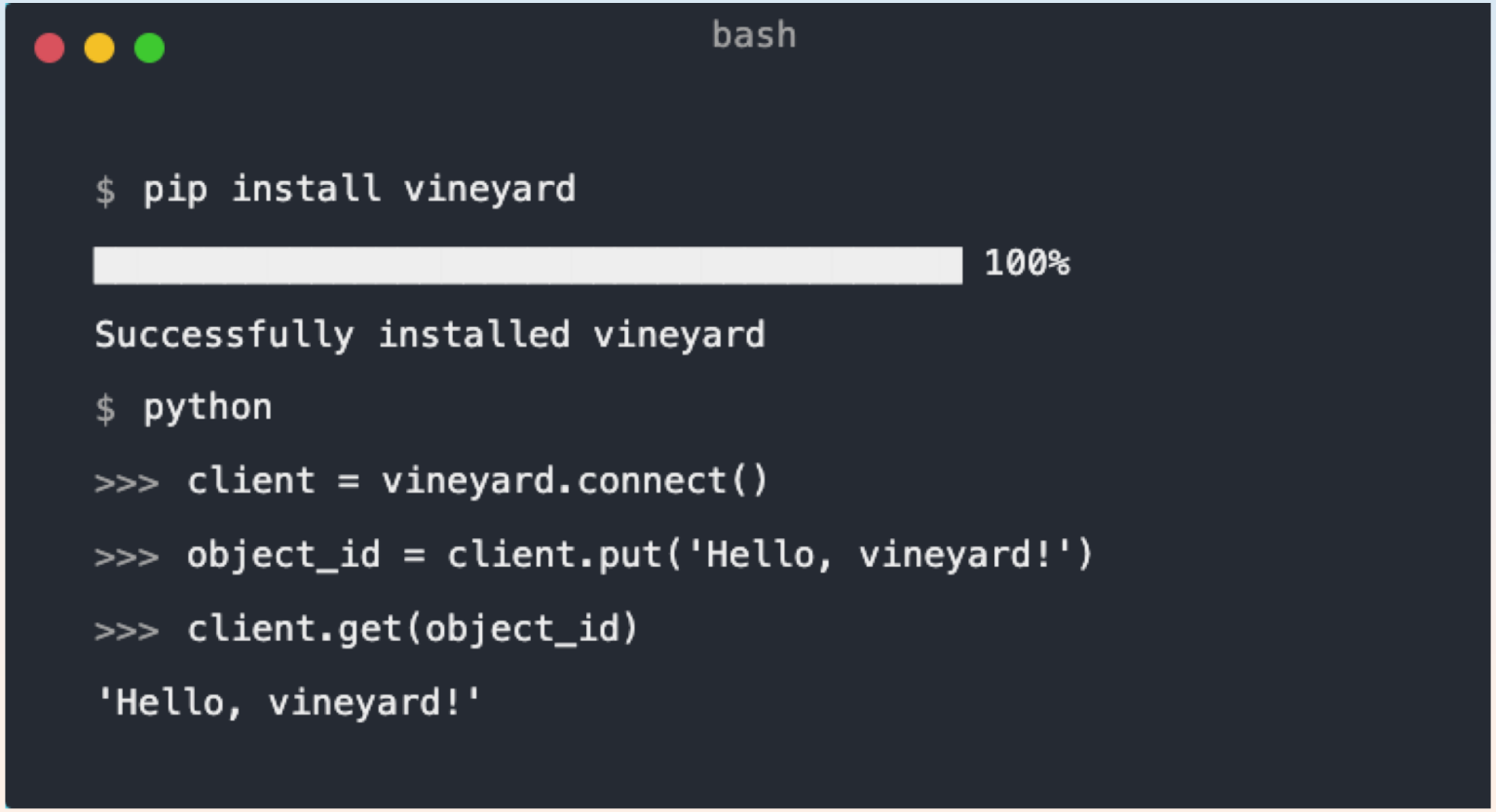


- 零拷贝的数据共享
- 开箱即用的数据结构和计算引擎集成
- 支持分布式数据和内存中放不下的数据
- 面向云原生设计

Vineyard：内存数据管理引擎

Vineyard

Vineyard 呈现给应用程序一个简单的对象存储的抽象



Vineyard：内存数据管理引擎

Vineyard

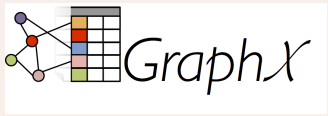
Vineyard 提供了各种编程语言开箱即用的SDK

Vineyard

Vineyard 与大数据生态集成

Vineyard

在很多常用计算引擎中Vineyard都能对中间数据存取取得显著的加速效果



表示对于一个get/put操作，vineyard相比于使用对象存储服务（e.g., OSS）能够取得的加速效果.

Vineyard achieves big performance gain with small integration efforts.

案例分析

Vineyard

通过 Vineyard 实现**零拷贝跨计算引擎**共享内存

GraphScope

Vineyard 如何做到**无需代码修改**即可加速 Python 数据分析工作流

Kedro Pipelines

如何使用 Vineyard 优化 **Kubernetes** 上机器学习应用中任务节点之间的I/O开销

Kubeflow

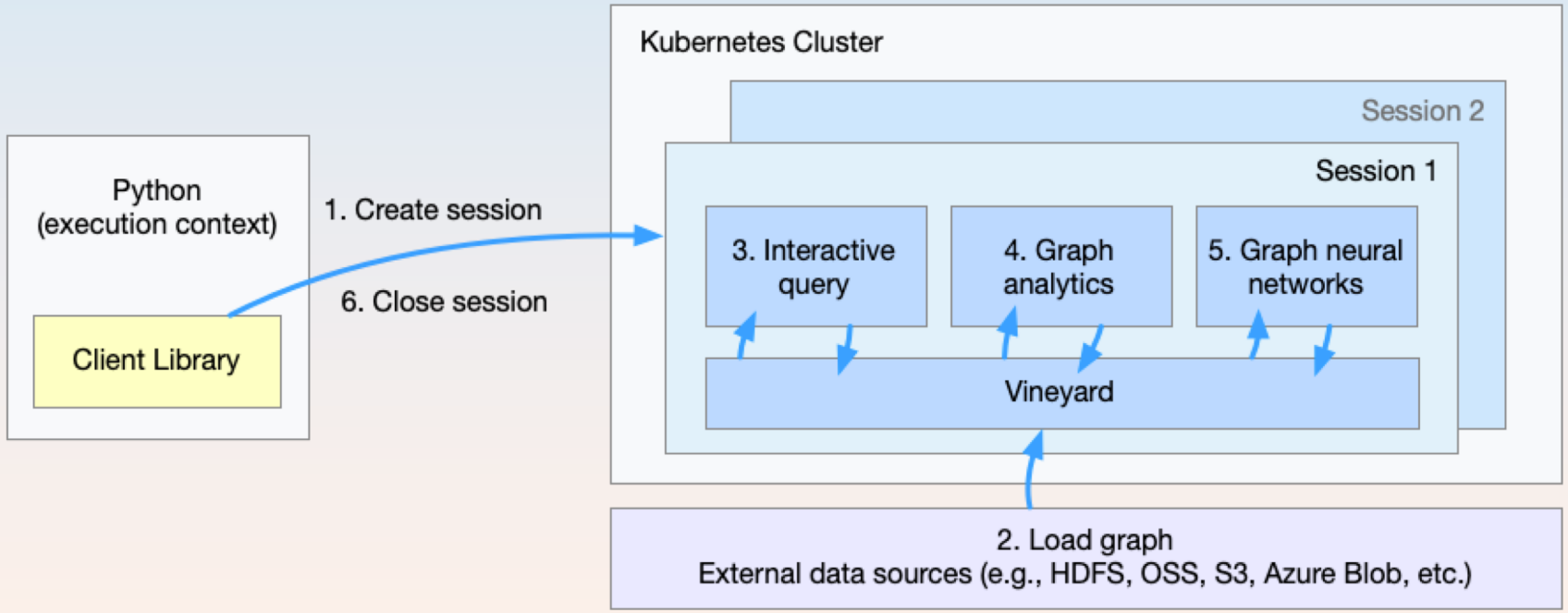
借助 Vineyard 打通**数仓大数据分析与新兴应用**的最后一公里

Apache Hive

Vineyard in GraphScope

Vineyard

- GraphScope 是一个支持图分析、图查询、图学习的**一站式**图计算系统



Vineyard 实现了 GraphScope 中图分析（ C++/MPI ）、图查询（ Rust/DataFlow ）、图学习（ C++&Python/PyTorch ）引擎之间零拷贝的内存共享

Vineyard + Kedro

Vineyard

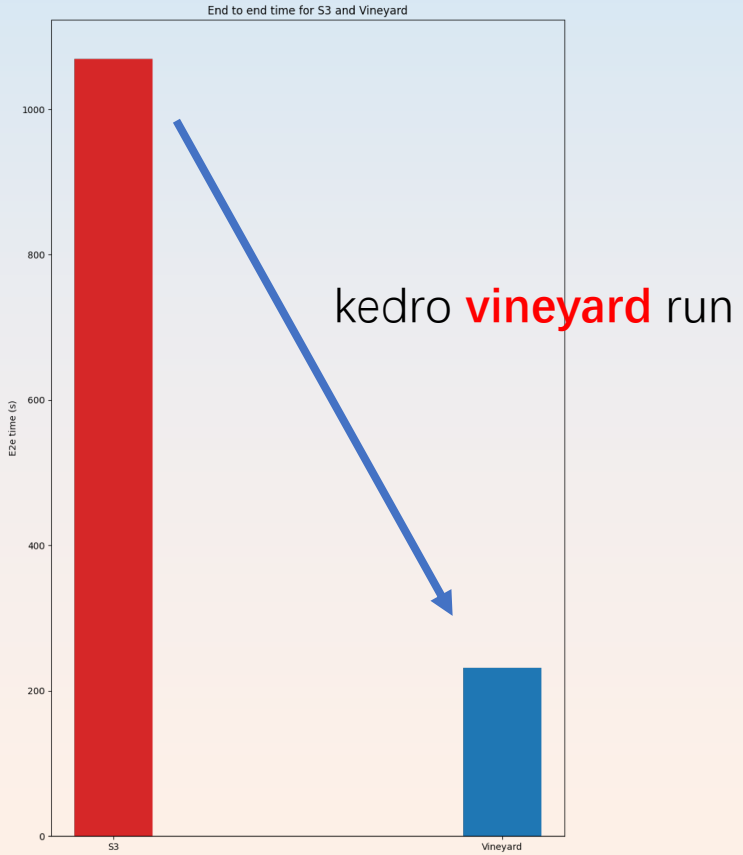
- Kedro 是一个面向 Python 的数据处理流水线编排引擎

```python
def split_data(
    data: pd.DataFrame, parameters: Dict[str, Any]
) -> Tuple[pd.DataFrame, pd.DataFrame, pd.Series, pd.Series]:
    X_train = ...
    X_test = ...

    return X_train, X_test, y_train, y_test

def make_predictions(
    X_train: pd.DataFrame, X_test: pd.DataFrame, y_train: pd.Series
) -> pd.Series:
    y_pred = ...

    return y_pred
```

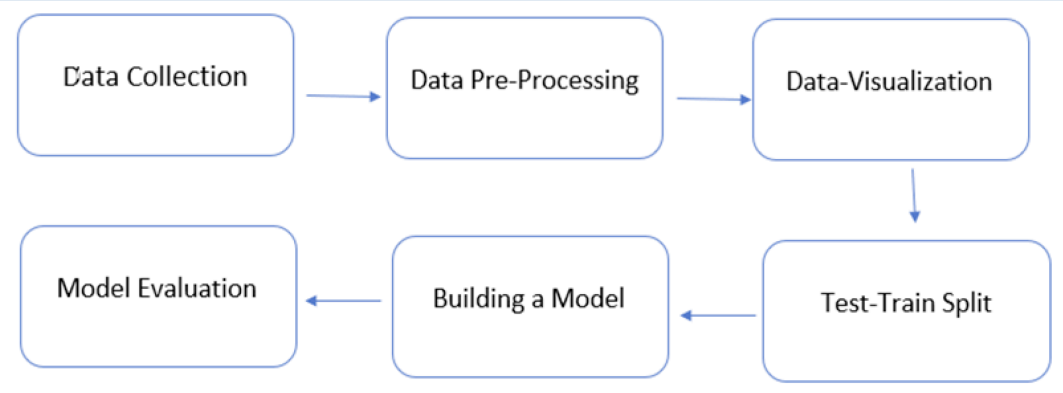


使用 Vineyard，在不修改用户已有代码的前提下，就能实现减少 75% 的端到端执行时间

Vineyard + Kubeflow Pipelines

Vineyard

- Kubeflow Pipelines 是 Kubernates 上的工作流编排引擎（特别是机器学习应用）



Kubernetes scheduler plugin

Vineyard

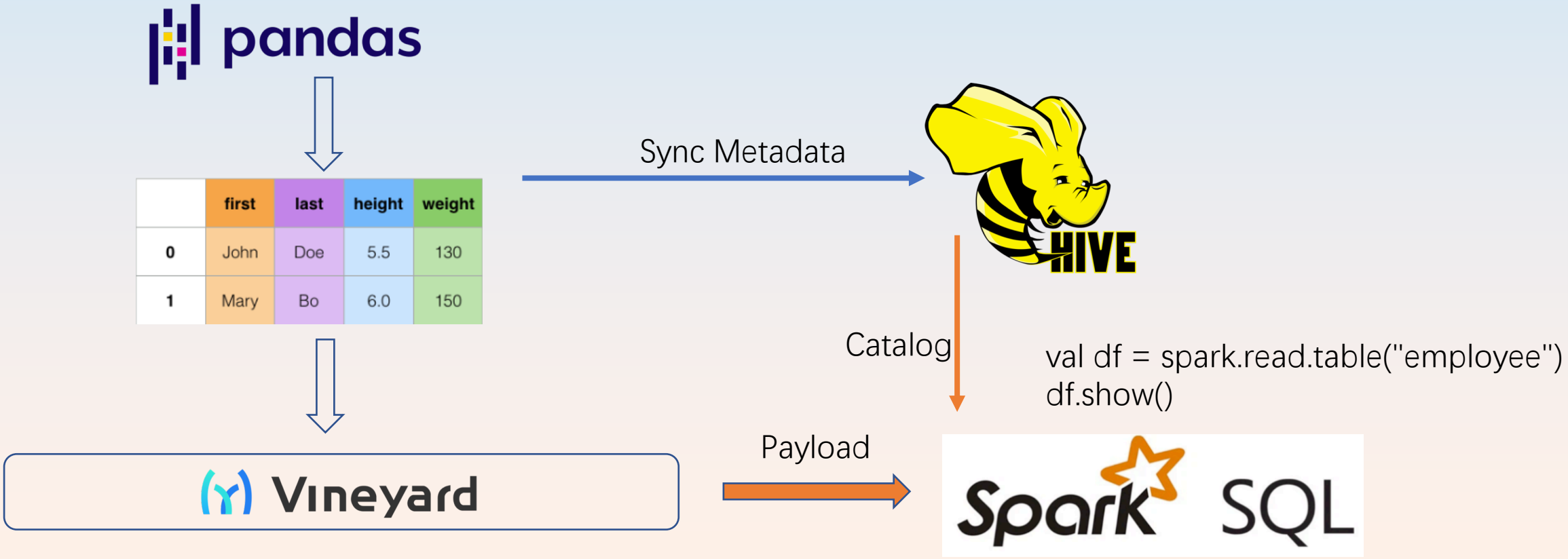
440 seconds → 352 seconds

Vineyard 借助调度器插件实现了 Kubernetes 上分布式工作流的数据感知的调度策略

Vineyard + Hive：连接数仓和应用程序

Vineyard

- 如何实现比SQL import/export更快的方式连接 Hive/Spark 和应用程序（PyData）

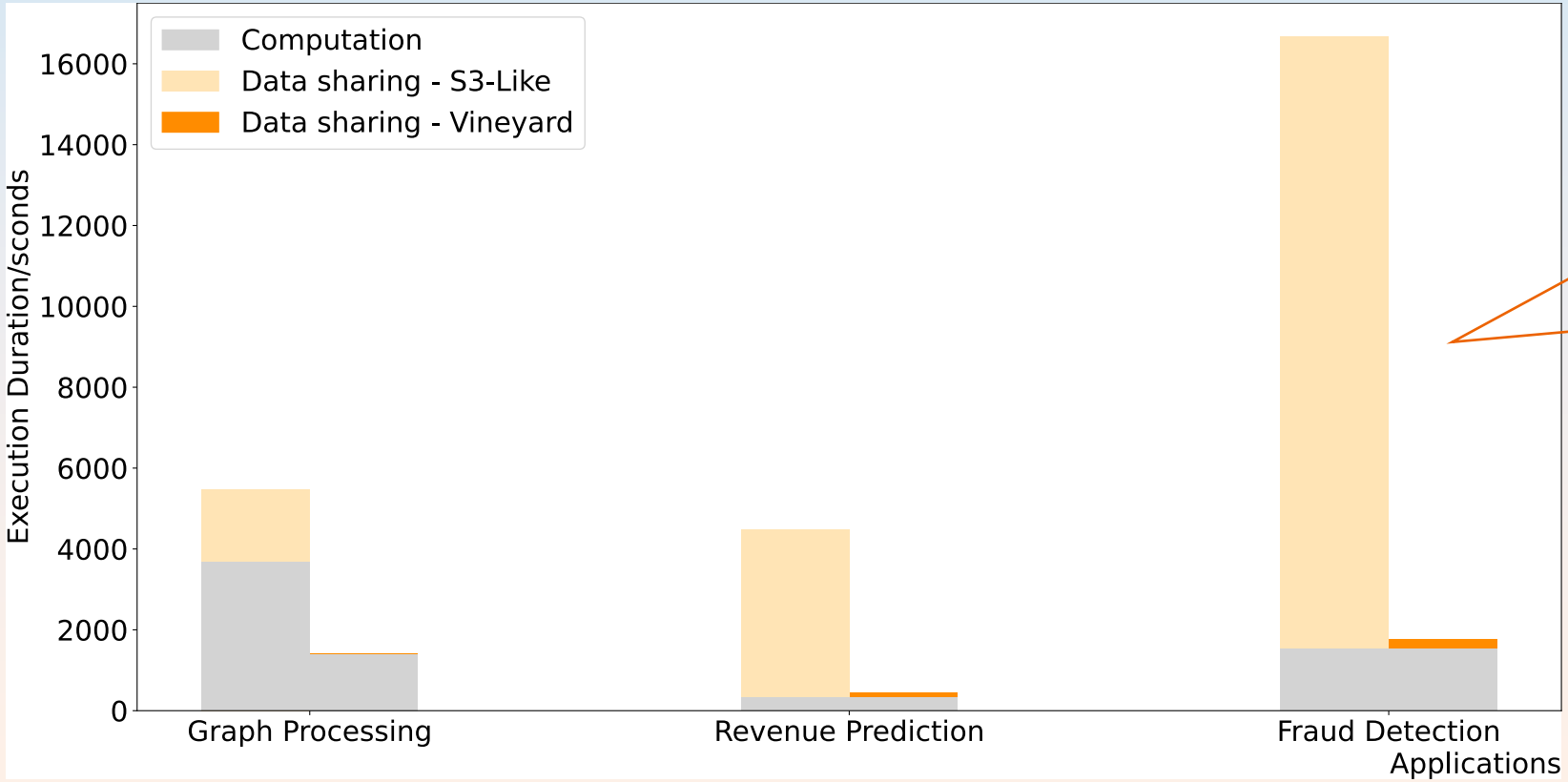


Vineyard 通过将metadata service和hive metastore的集成，实现Hive/Spark和PyData其他生态的高效无缝衔接

真实场景中的性能加速

Vineyard

几个不同领域的真实应用中对Vineyard的性能评估



从图中可以看到，中间结果共享的使用外部文件存储服务效率低、额外开销高在真实世界的大数据分析中是普遍现象

Vineyard 对于不同类型的应用都能起到很好的优化效果

总结

Vineyard

- **Vineyard 是一个为高效共享大数据分析工作流中任务之间中间结果传递设计的数据管理引擎**
  - *通过共享内存实现零拷贝数据共享;*
  - *容易被计算引擎集成;*
  - 支持分布式数据和数据内存中放不下的场景;
  - 对实际应用加速效果显著。
- **Roadmap**
  - *发布 Hive Connector;*
  - *GPU 内存共享以及更好的机器学习应用的支持;*
  - CXL 与分布式内存等前沿技术。

- Vineyard is **open-sourced** and is part of ***CNCF.***

扫描上方二维码关注我们！
https://github.com/v6d-io/v6d.

欢迎试用、共建和star!

CLOUD NATIVE SANDBOX

开源社
kaiyuanshe

THANK YOU

QUESTIONS?

欢迎扫码打卡
积分可兑换对应礼品哟！

微信公众号：开源社KAIYUANSHE
视频号：开源社KAIYUANSHE
新浪微博：开源社
B站：开源社KAIYUANSHE

简书：开源社
头条：开源社
Facebook：KaiyuansheChina
Twitter：开源社KAIYUANSHE

扫码关注开源社公众号

扫码添加讲师联系方式