先进计算产业发展联盟
Advanced Computing Industry Alliance

大数据云原生技术发展研究报告
（2023 年）

先进计算产业发展联盟

2023 年 12 月

研 究 报 告 要 点

随着行业的快速发展，数据量也呈爆炸式增长，大数据已成为决策的基本工具，企业面临着数据管理和处理的巨大挑战。围绕 Hadoop 的传统大数据架构在交付运维，资源利用率，系统迭代与兼容性，安全等方面存在诸多不足。

随着以 Kubernetes 为代表的云原生概念的兴起，越来越多的企业投身于云原生转型的浪潮，以解决传统应用面临的弹性能力不足、资源利用率较低、迭代周期较长等问题。当前云原生的发展已比较成熟，成为数字化转型的重要支撑技术。

大数据和云原生技术的融合，逐渐成为企业数字化转型的重要演进方向，目前还处于高速发展，百家争鸣的阶段，一些企业已经在大数据云原生之路上砥砺前行，而国内大部分企业依然处于观望状态。

同时可以看到，业界在大数据与云原生结合的定义和方向上，有一些不同的声音，不同企业融合的方式也有所不同。大数据和云计算要不要融合？如何融合？都是人们所关心的话题。

带着这个问题，我们希望基于之前积累的经验，结合工作中的痛点，调研并产出一份**尽量中立、客观、完整的**大数据云原生技术发展报告，希望能为相关企业、研发团队和需要大数据的客户提供参考。也希望能抛砖引玉，吸引更多企业专家的参与，引发后面更专业，更大范围有关大数据云原生技术的讨论，最终能促进一些共识，提升大

数据云原生的技术普惠，“旧时王谢堂前燕，飞入寻常百姓家”，为国家数字化转型，做一点贡献。

本报告将从以下几个方向来阐述：

1.**大数据与云原生技术的发展与演进：**介绍大数据技术的演进，云原生技术的发展，以及它们融合的情况。

2.**传统大数据平台的痛点：**探讨传统大数据平台交付运维成本高、资源利用低效、迭代兼容性及安全性问题等关键痛点。

3. **云原生技术解决思路：**详细说明云原生技术解决这些关键痛点的思路，以及云原生技术带来的其它好处和引入的新挑战。

4.**大数据云原生技术架构简述：**简述大数据云原生技术的设计思路和参考架构，包括弹性伸缩、资源隔离、容器化、统一资源调度、多计算引擎管理、统一数据湖管理以及智能化运维等方面。

5.大数据云原生的未来发展和建议：最后简单提出对大数据云原生技术未来发展和建议，以提升大数据云原生的技术普惠。

由于时间仓促，水平所限，错误和不足之处在所难免，欢迎各位读者批评指正，意见及建议请发送至 wangxiaogang@chinatelecom.cn。

研究单位：天翼云科技有限公司

课题负责人：王小刚

课题参加人：侯圣文，张桐，李冰，金喆，李启蒙，黄俊慧，李庆森

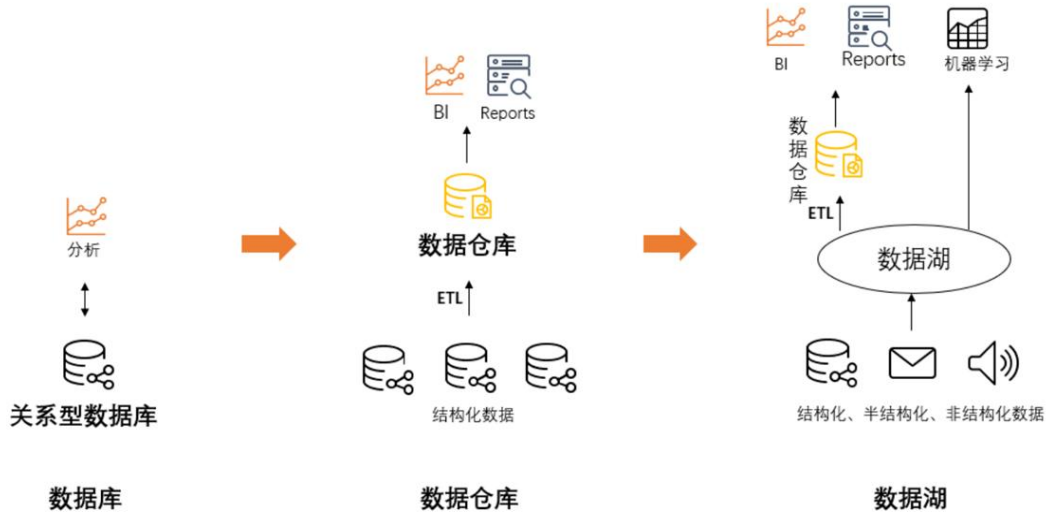
完成日期：**2023** 年 **12** 月

目　录

一、大数据平台与云原生技术的发展与演进

（一）数据平台的发展与演进

需求催生技术革新，在海量数据需求的推动下，数据平台架构持续演进，经过数十年的发展，历经了数据库、数据仓库、数据湖、湖仓一体等概念。这里按出现顺序简述：（其中关于数据湖和湖仓一体目前业界有多种不同的定义，这里我们采用其中一种定义说明）



来源：CCSA TC601 大数据技术标准推进委员会

图 1：数据分析技术演进图

数据库（Data Base）：

自 1980 年代初至中期起，数据管理工具主要呈现为数据库形式，以面向事务交易的 OLTP 场景为主，数据分析功能则作为辅助。这些数据库主要用于向管理层提供固定报表，支持宏观管理决策。它们通过标准 SQL 提供数据分析能力，主要代表产品包括 Oracle、Sql Server、Mysql 等。

- **以前**：提升单机性能：IBM小型机、EMC企业级存储、Oracle企业级数据库

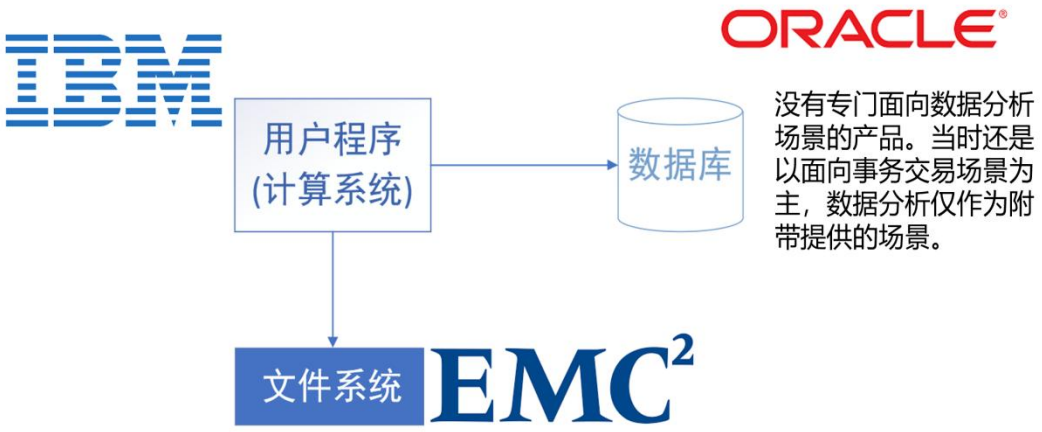
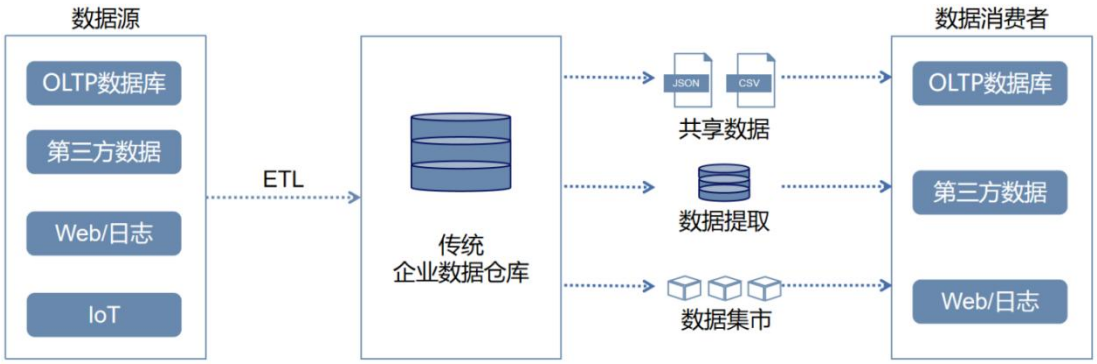


图 2 早期数据库阶段系统架构

数据仓库（Data Warehouse）：

随着互联网的快速普及，门户、搜索引擎、百科等应用快速增长，数据量呈爆发式增长，原有的单个关系型数据库架构无法支撑庞大的数据量。20 世纪 90 年代数据仓库理论被提出，核心是基于 OLTP 系统的数据源，根据联机分析处理 OLAP 场景诉求，将数据经过数仓建模形成 ODS、DWD、DWS、DM 等不同数据层，每层都需要进行清洗、加工、整合等数据开发（ETL）工作，并最终加载到关系型数据库中。



来源：云原生产业联盟

图 3：OLAP 系统建设

数据仓库架构是为了解决单个关系型数据库架构无法支撑庞大数据量的数据存储分析问题。传统数据仓库多为 MPP(Massively Parallel Processor)架构，代表产品有 Teradata、Greenplum 等，当前 MPP 架构依然为新型数仓的重要选择，比如 ClickHouse，Doris，StarRocks 等。

随着 Hadoop 技术的成熟与普及，基于 Hadoop 自建离线数据仓库(Hive)是常见的大数据平台之上数据仓库方案，在目前依然发挥着重要的作用。

数据湖（Data Lake）：

随着移动互联网的飞速发展，半结构化、非结构化数据的存储、计算需求日益突出，对数据平台提出了新的要求。

以开源 Hadoop 体系为代表的开放式 HDFS 存储(或 S3)、开放的文件格式、开放的元数据服务（Hive Metastore 等）以及多种引擎（Hive、Spark、Flink、Presto 等）协同工作的模式，形成了数据湖的雏形。

来源：云原生产业联盟

图 4：Hadoop 生态系统重要组件

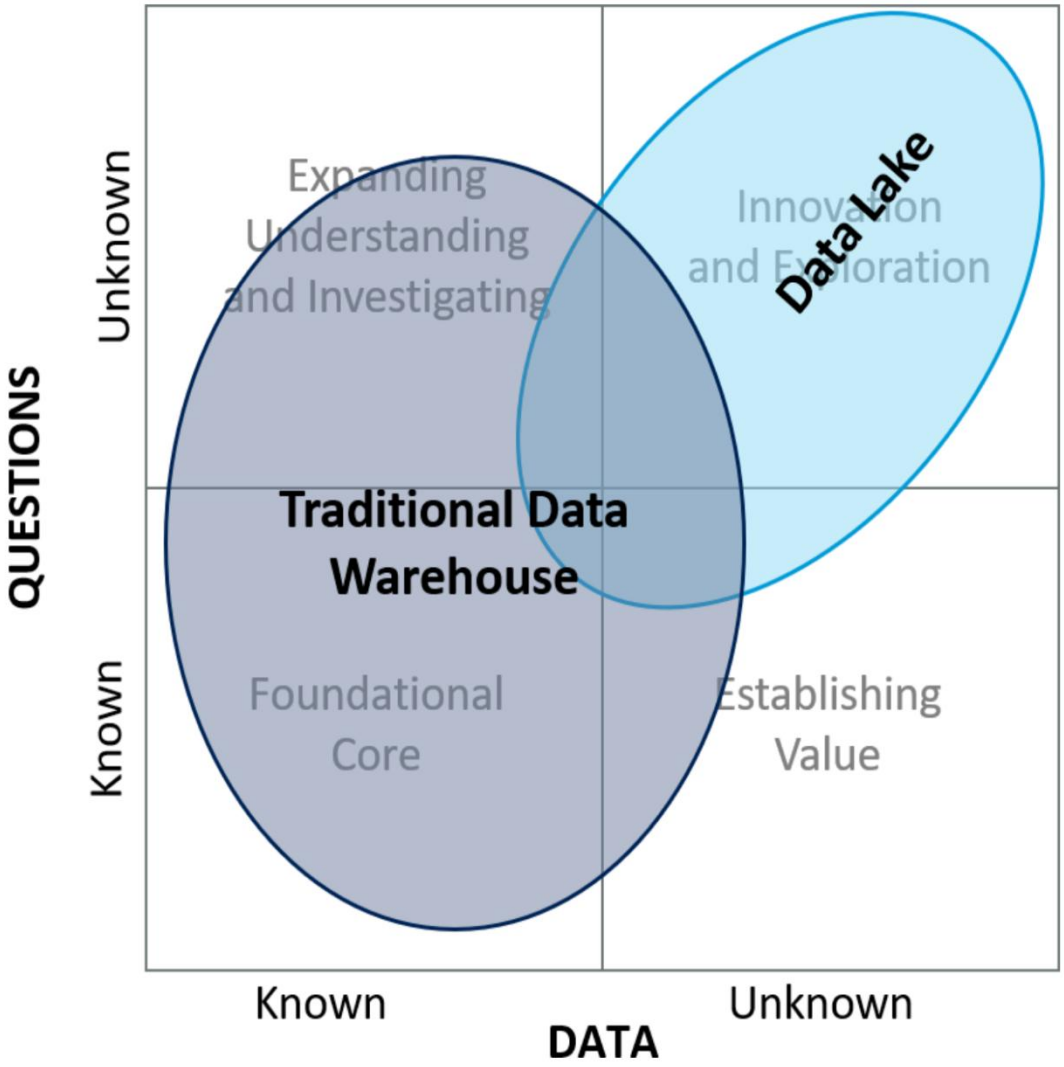
2010 年，数据湖概念被提出，数据湖是一种支持结构化、半结构化、非结构化等数据类型大规模存储和计算的系统架构。数据湖与数据仓库的主要区别在于数据仓库中数据在进入仓库之前是需要实现归类，而数据湖是把大量原始数据通过廉价存储保存下来。数据湖架构的特点可总结为：低成本、原始数据、需灵活使用、面向任务数据绑定、不提前定义数据模型。

表 1 数据湖与数据仓库对比表

差异项	数据湖	数据仓库
数据类型	所有数据类型	历史的、结构化的数据
Schema	读取型 Schema	写入型 Schema
计算能力	支持多计算引擎用于处理、分析所有类型数据	处理结构化数据，转化为多维数据、报表，以满足后续高级报表及数据分析需求
成本	存储计算成本低，使用运维成本高	存储计算绑定、不够灵活、成本高
数据可靠性	数据质量一般，容易形成数据沼泽	高质量、高可靠性、事务隔离性好
扩展性	高扩展性	扩展性一般，扩展成本高
产品形态	一种解决方案，配合系列工具实现业务需求，灵活性更高	一般是标准化的产品
潜力	实现数据的集中式管理，能够为企业挖掘新的运营需求	存储和维护长期数据，数据可按需访问

来源：CCSA TC601

从解决场景的角度来看，数据仓库和数据湖各有其适合覆盖场景，基本上属于互补关系，前者更多是解决固定的、明确的数据问题；后者则为应对随机性、探索式的数据问题。下图是一个示意图。

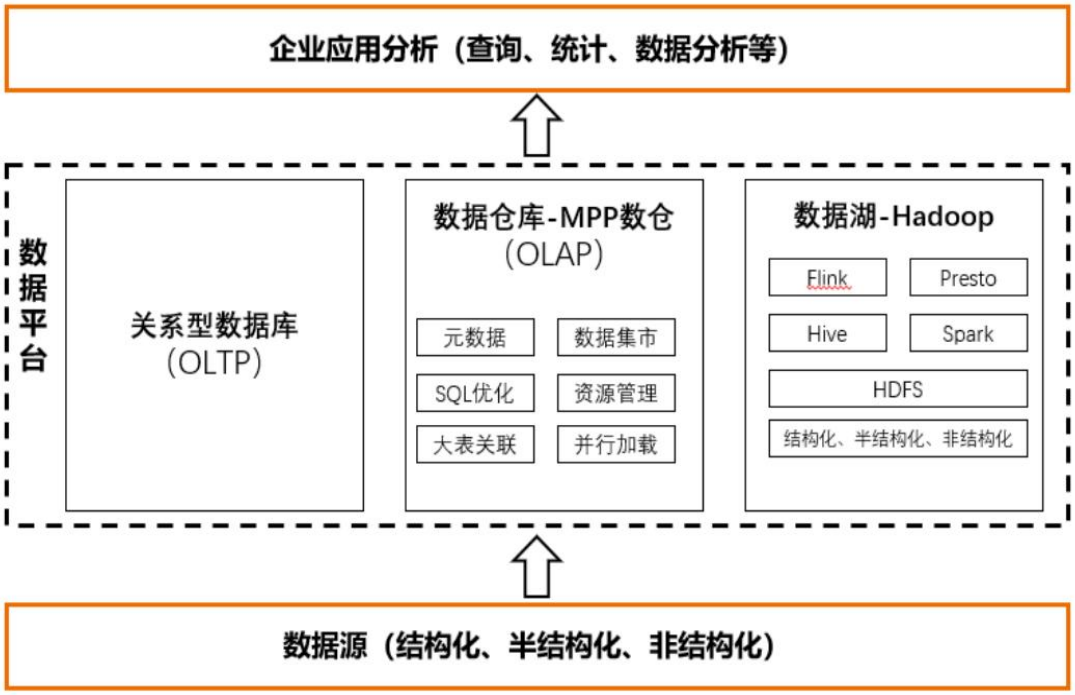


来源：Gartner

图 5：数据仓库和数据湖的场景

湖仓一体（LakeHouse）：

为满足多种数据类型存储、多场景分析等业务诉求，企业的数据采用混合部署模式，其中数据湖和数据仓库通过 ETL 进行数据交换，数据湖和数据仓库是两套独立的体系。



来源：CCSA TC601

图 6：湖+仓混合架构图

“数据湖+数据仓库”混合架构满足了结构化、半结构化、非结构化数据高效处理需求，解决了传统数据仓库在海量数据下加载慢、数据查询效率低、难以融合多种异构数据源进行分析的问题，但混合架构是技术向业务妥协的一个产物，存在数据冗余，增加存储成本，两个系统间额外的 ETL 流程导致时效性差，数据一致性保障低，混合架构开发运维复杂等弊端。

2020 年 Databricks 提出“湖仓一体”的概念，到目前技术和概念侧依然在持续演进。湖仓一体是指融合数据湖与数据仓库的优势，形成一体化、开放式数据处理平台的技术。通过湖仓一体技术，可使得数据处理平台底层支持多数据类型统一存储，实现数据在数据湖、数据仓库之间无缝调度和管理，并使得上层通过统一接口进行访问查询

和分析。总的来看，湖仓一体通过引入数据仓库治理能力，既可以很好解决数据湖建设带来的数据治理难问题，也能更好挖掘数据湖中的数据价值，将高效建仓和灵活建湖两大优势融合在一起，提升了数据管理效率和灵活性。

湖仓一体目前没有统一的架构，在企业需求的驱动下，各开源技术和厂商基于原有架构演进，数据湖与数据仓库在原本的范式之上扩展。逐渐形成了“湖上建仓”与“仓外挂湖”两种湖仓一体实现路径。如图 7 和表 2 所示。湖上建仓和仓外挂湖虽然出发点不同，但最终湖仓一体的目标一致。

表 2 两种实现路径对比表

实现路径	优势	劣势	需解决的问题	实现方向
湖上建仓 （Hadoop 体系）	支持海量数据离线批处理	不支持高并发数据集市、即席查询、事务一致性等	1.统一元数据管理 2.ACID 3.查询性能提升 4.存储兼性问题 5.存算分离 6.弹性伸缩	1.提升查询引擎、存储引擎能力
仓外挂湖 （MPP 体系）	事务一致性，结构化数据 OLAP 分析	不支持非结构化/半结构化数据存储、机器学习等	1.统一元数据管理 2.存储开放性 3.扩展查询引擎 4.存算分离 5.弹性伸缩	1.计算引擎不变，只扩存储能力。 2.查询引擎扩展，提升查询引擎效率

来源：CCSA TC601

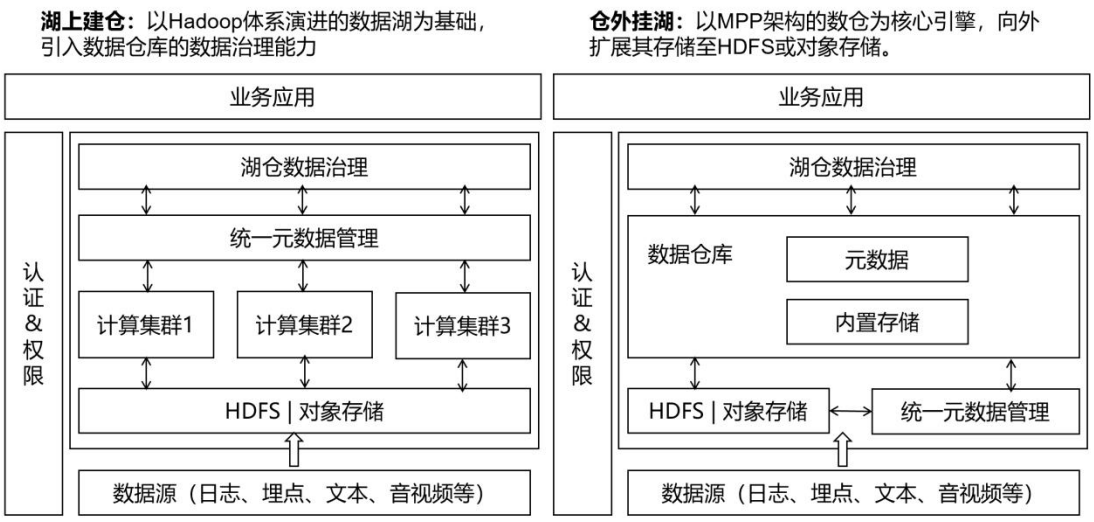
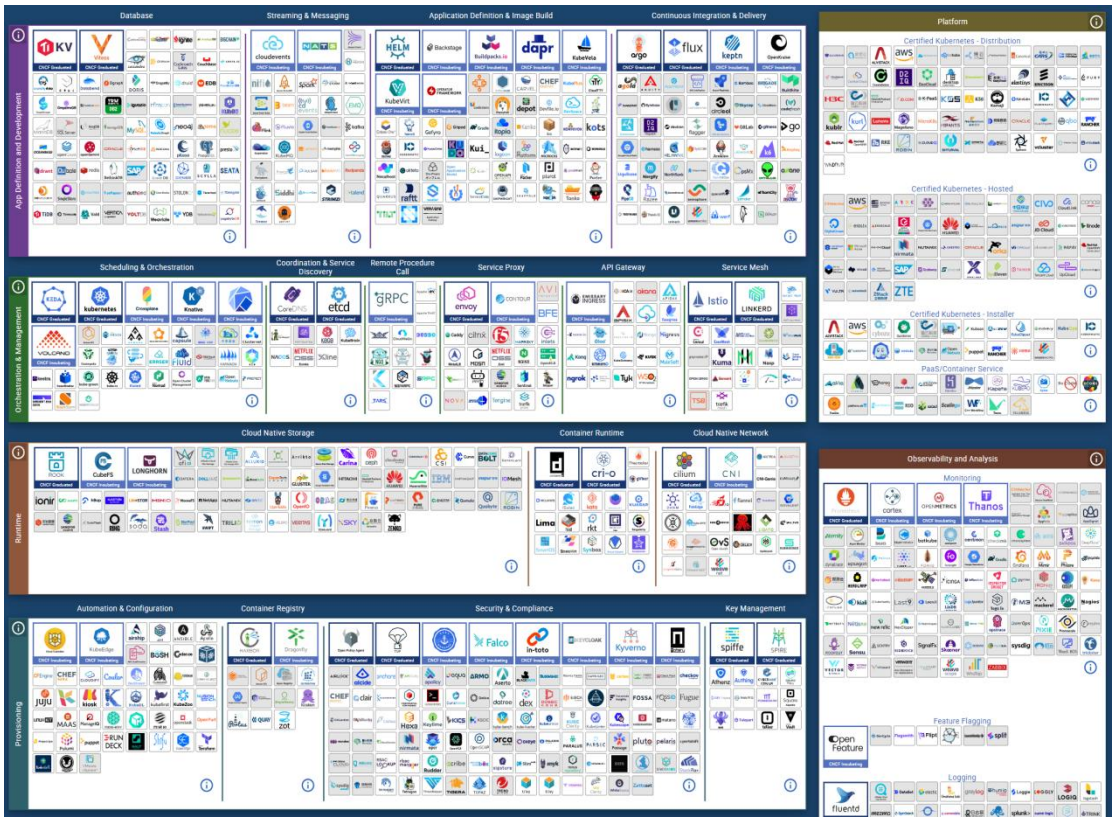


图 7：湖仓一体架构模块图

（二）云原生技术简述

云原生的发展简述：

云原生(Cloud Native)，最初由 Pivotal 公司的 Matt Stine 在 2013 年提出，随后 Linux 基金会在 2015 年成立了云原生计算基金会(CNCF)。CNCF 不仅推广了云原生这一概念，还逐步构建了以云原生为核心的技术生态工具。到 2018 年，Kubernetes 成为 CNCF 的首个毕业项目。目前，Kubernetes 已经确立了在容器编排领域的领导地位，并推动了云原生技术的广泛应用。



来源：https://landscape.cncf.io/

图 8：云原生 Landscape（景观）指南

云原生的核心思想：

云原生普遍被认为包含四大核心要素：DevOps、微服务、持续交付和容器化。

DevOps：DevOps 是开发(Development)和运维(Operations)的结合，它推动了开发和运维团队的紧密协作。在 DevOps 文化中，软件的开发、测试、部署和监控过程是连续的，不断循环，旨在加快软件交付速度并提高质量。

持续交付：持续交付是一种软件工程方法，它允许软件在短时间内且持续地被交付到生产环境。它通过自动化开发、测试和部署流程来支持频繁的版本发布，旨在减少发布新功能和修复的时间。

微服务：微服务架构是一种设计方法，将应用程序分解为一组较小、相互独立的服务，每个服务都围绕特定业务功能构建，并可独立部署。

容器：容器技术，如 Docker，提供了一种轻量级、可移植的方法来封装、部署和运行应用。Kubernetes(K8S)则发展为容器编排和管理的领导者，它提供了高级的部署、扩展和运行容器化应用的能力。

CNCF 重新定义云原生：

随着云原生生态的不断壮大，CNCF 基金会容纳的项目越来越多，到了 2018 年，原来的定义已经限制了云原生生态的发展，CNCF 为云原生进行了重新定义：

“云原生技术有利于各组织在**公有云**、**私有云**和**混合云**等新型动态环境中，构建和运行**可弹性扩展**的应用。云原生的代表技术包括**容器**、**服务网格**、**微服务**、**不可变基础设施**和**声明式 API**。

这些技术能够构建**容错性好**、**易于管理**和**便于观察**的**松耦合系统**。结合**可靠的自动化手段**，云原生技术使工程师能够轻松地对系统作出**频繁和可预测的重大变更**。

云原生计算基金会(CNCF)致力于培育和维护一个厂商中立的开源生态系统，来推广云原生技术。我们通过将最前沿的模式民主化，让这些创新为大众所用。”

新的定义继续保持原有的核心内容：容器和微服务，加入了服务网格、不可变基础设施和声明式 API 这三个重要设计指导理念，并且强调了公有云、私有云和混合云等新型动态环境。

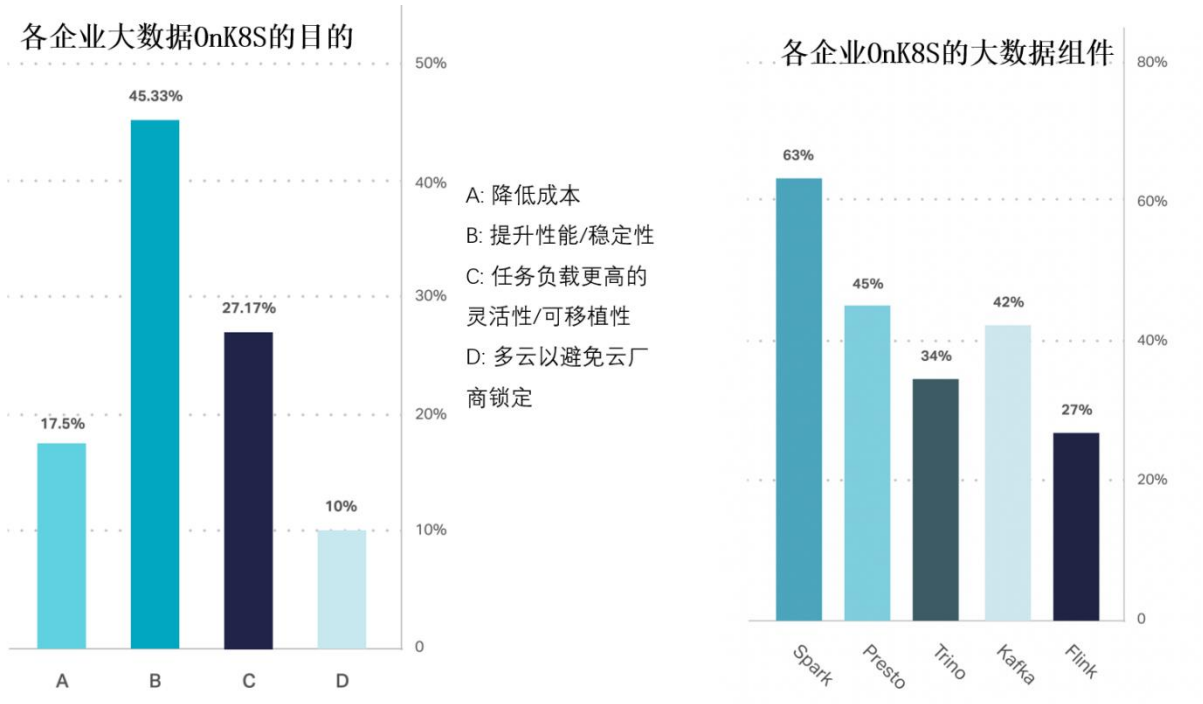
（三）大数据与云原生结合分析

从 2018 年起，一些开源大数据引擎陆续开始了 on Kubernetes 的探索：

- 2018 年 3 月，Spark v2.3.0 开始探索 on Kubernetes
- 2018 年 6 月，Kubernetes Airflow Operator 在正式发布
- 2019 年 8 月，Starburst Presto 宣布支持 K8S
- 2020 年 2 月，Flink v1.10.0 发布 Native Kubernetes beta 版
- 2020 年 2 月，Hive 探索 MR3 运行在 Kubernetes 上

而到了 2021 年 3 月，Apache Spark 3.1 正式支持了 kubernetes，越来越多的企业开始在生产环境使用大数据云原生融合的技术。

根据 Pepperdata 关于“2022 BigData on Kubernetes Report”显示：



来源：Pepperdata 2022 BigData on Kubernetes Report

图 9：大数据云原生融合情况与上云目的

50%+的受访者正在将大数据应用迁移至 Kubernetes 中，以降低整体成本。报告显示，迁移原因排名前四的是：

1. 超过 45%的受访者为了提高应用程序的性能和稳定性而选择将任务迁移至 Kubernetes。
2. 为任务负载具备更高的灵活性和可移植性。
3. 降低成本。
4. 多云解决方案避免被一个云厂商绑定。

从任务分布上可以看出：占比最高的大数据作业依次是 Spark、Presto、Kafka、Trino、Flink，可以看到基本上涵盖了大数据领域的批处理、交互式分析和流处理场景。

国内外云厂商也一直加大产品技术在云原生(Serverless)方向的投入和引导。亚马逊云科技在 2022 年 re:Invent 中宣布其大数据核心

产品已全面 Serverless 化，阿里云在 2023 年云栖大会上将 Serverless 作为大会主题之一，包括大数据在内的多款云产品均发布了其 Serverless 相关的产品能力。

单纯从技术趋势的角度来看，无论是技术还是产品，大数据云原生化都是发展的必然趋势。

与云原生融合的一些其它声音，也值得我们关注：

值得一提的是，在数据库领域，对于“数据库是否应该放到 K8S”这个话题有一些讨论，虽然大数据有一些不同，但可以参考：

反方认为：**维持已经成熟和可靠的系统不需要 K8S：**

认为将数据库放入 K8S 中会导致“双输”——K8S 失去了无状态的简单性，不能像纯无状态使用方式那样灵活搬迁调度销毁重建；而数据库也牺牲了一系列重要的属性：可靠性，安全性，性能，以及复杂度成本，却只能换来有限的“弹性”与资源利用率，但虚拟机也可以做到这些。

整体论点分为以下四点：

1. **增加复杂度和不可靠性**：K8S 增加了额外的架构复杂度和潜在失效点。
2. **性能挑战**：在 K8S 上运行的数据库可能面临性能问题。
3. **安全和合规风险**：多租户环境和更多的组件依赖，增加数据库的安全威胁，使得审计和合规更加复杂。
4. **成本和维护问题**：尽管 K8S 在一定程度上简化了数据库管理，但可能无法抵消其自身引入的复杂度和维护成本。

正方认为，**数据库 on K8S，是专业能力的普及化：**

这里的“专业能力”指的是高可用性、弹性伸缩、容错等能力，它们通常需要复杂的技术实现和专业知识。通过将数据库部署在 K8S 上，这些能力可以通过 K8S 的自动化和标准化机制更容易地实现和普及，降低了对专业数据库管理技能的依赖。

整体论点主要也是下面四点：

1. **资源弹性和伸缩性**：K8S 提供强大的资源弹性和伸缩性，适应业务需求的波动，在高峰期自动扩展资源，在低谷期收回，有效节约成本。
2. **容器技术的优势**：Docker 等容器技术提供了轻量级和标准化的部署方式。容器对数据库性能影响很小。
3. **K8S 的运维能力**：包括路由网关、水平扩展、监控、备份和灾难恢复等，这些能力有助于数据库的高可用性和持续运行。
4. **高可用性等解决方案**：固化高可用方案，提供主从秒级切换和数据一致性等特性。

“它山之石，可以攻玉”，虽然数据库与数据平台在场景和技术架构上会有一些不同，但大数据云原生的融合在这个问题上依然值得借鉴思考。

二、传统大数据平台的需求与痛点

传统围绕 Hadoop 生态构建的大数据平台，存在着交付运维成本高、资源利用率低、系统迭代与兼容性挑战和安全相关挑战。用户期

望大数据云原生技术能够在这些方面为传统的大数据平台带来优化和改进。

（一）交付运维成本高

传统大数据平台的建设和维护目前是一个重要且复杂的任务。这些平台通常包括多种不同的组件，它们在技术栈、功能和架构上存在显著差异。这种多样性不仅使得平台的部署和配置变得极为复杂，也大幅增加了整体的运维成本。

组件多样性导致较高的人力需求：

大数据平台包含多种组件，这些组件在技术栈（如 Java、C++）、功能（如流处理、批处理，OLAP）和架构（如 C/S、MPP）方面各不相同。部署，配置和维护如此多样的技术栈需要大量的专业人力，特别是在部署和交付新的大数据平台时，人力资源的需求和成本会显著增加。

运维专业性与效率：

大数据组件的复杂性，具有较高的运维知识门槛。这不仅增加了运维团队的工作负担，还可能导致效率低下和更频繁的解决问题需求，从而增加成本。

工具与管理挑战：

许多大数据组件缺乏开箱即用的日志、监控和告警功能，导致运维团队需要为每个组件单独开发和适配这些工具。

每个组件可能有各自的集群和管理界面，使得整个平台的统一管理和问题排查变得困难。这种分散性不仅降低了运维效率，还可能导致问题解决的延迟，增加了管理成本。

云原生技术可以有效缓解传统大数据平台的运维挑战。容器化通过屏蔽不同组件间的技术栈和基础设施差异，简化了运维流程。工具如 Operator 实现了服务部署和运维的标准化与自动化，降低了复杂性和人力成本。在云原生架构下，应用和组件的更新仅需拉取新镜像并重启容器，确保环境一致性，加速应用发布。

此外，云原生环境提供统一管理界面，集中处理不同服务的发布和运维，提高问题监测和定位的效率。集成在 Kubernetes Pods 和节点的监控与告警工具，使得运维团队可以通过统一界面清晰监控基础设施和服务状态，有效跟踪系统性能和健康状况。

（二）资源利用率低

组件混部困难：传统的大数据平台，为了避免组件间的相互影响，组件的集群往往是相互独立部署的。这样虽然对于整体的编排来说相对简单，但是会降低资源利用率。

业务波动性：由于大数据业务的特点，大数据组件在高峰期的资源利用率可以很高，但是在业务低峰期则会有较多闲置，此时集群整体资源利用率可能只有 20%-30%，综合平均起来集群整体的资源利用率偏低。

弹性扩缩容难度高：在业务高峰期时，如果现有的资源已经无法支撑业务，这个时候可能需要通过较为繁琐的运维流程扩容节点。到了业务低谷期时，这部分机器又很难通过运维流程快速下线，扩缩容的效率不高。

云原生领域在基础资源基础上抽象出了“资源池（计算、存储、网络等的组合）”的概念，资源池被所有的大数据组件公用，按需申请，可以避免重复规划造成的资源浪费。资源池可以承载不同类型的大数据集群，可以是批处理、也可以是流处理、MPP 等，得益于容器化较好的隔离性，不同的业务可以在一起混部，形成资源的分时复用；同时云原生领域有着丰富的工具来对资源池进行弹性扩缩容，甚至当业务不存在的时候降到零副本，做到随需启停，做到 Serverless 化。

（三）系统迭代与兼容性挑战

传统的大数据平台，组件开发迭代不敏捷，周期长。组件版本往往比较固定，版本升级难度比较大。在部署新的大数据组件到现有的 Hadoop 集群时。首先，必须确保这些新组件与现有的 HDFS 和 Yarn 版本兼容。由于许多新的大数据组件不支持旧版本的 Hadoop，升级 Hadoop 可能导致现有组件失效。其次，部署时还需考虑到不同 Linux 操作系统间的兼容性问题，这增加了额外的复杂性。因此，整合一个新的计算和存储组件到现有架构中通常是一个耗时的过程，可能需要几天甚至几周的时间。

云原生的定义中先天具有 DevOps、CI/CD 的思想，可以比较容易地做到IaC(Infrastructure as Code)和CaaS(Configuration as a Service)。组件的配置或代码在 Git 中进行保管，一旦发生变化，就会经过 CI 的环节进行质量验证，然后通过 CD 的管道打包成容器镜像发布到运行环境中。由于云原生中都是微服务架构，你可以只单独发布组件的一部分，不必整体组件一起发布；在组件的发布过程中，也可以更方便地实现 A/B Test、灰度发布、发布回滚等操作。同时云原生中都是容器，启停容器是轻量级动作，效率高。

（四）安全相关挑战

传统大数据平台，在安全方面面临着一系列挑战。包括数据隔离、综合访问控制、数据加密和保护、审计与合规性、网络安全、灾难恢复与数据备份等。针对这些挑战，大数据开源社区已经有一些解决方案，如使用 Kerberos 进行认证、Ranger 或 Sentry 进行访问控制等，但这些解决方案往往伴随着配置复杂、灵活性不足、自动化程度低等缺点。

相比之下，云原生技术社区也提供了面对安全问题更全面、灵活且自动化的安全解决方案。可以做到更好的网络隔离，数据隔离，权限隔离，灾备等。

三、云原生技术解决大数据问题的思路

（一）云原生技术提升运维交付质量与效率

1、容器化

云原生中部署的都是容器，容器的不可变基础设施属性，屏蔽了基础设施层对应用的影响：如应用的依赖包是否安装好，应用所在机器的 CPU 架构等，提高了应用程序的可移植性，大大降低了平台交付过程中环境标准化工作的成本。

2、Operator 机制

大数据组件种类繁多，每个大数据组件的安装和运维方式不尽相同，人力维护成本较高。Kubernetes 中提出了使用 Operator 的方式来管理复杂应用。通过 Operator 的方式标准化了大数据组件的安装和运维，将复杂的安装步骤、大数据组件升级、高可用配置等运维操作自动化，减少了人工成本，同时也提高了运维工作的及时性和效率。

3、统一的日志收集与监控报警

在云原生技术栈中，日志收集展示和监控报警都有标准化的实现方式，如采用 sidecar 容器收集业务容器中的日志，采用 Prometheus 技术体系收集监控指标与配置报警规则。这样就可以用统一的方式将所有组件的日志和监控汇总到统一页面来查看，方便了问题的跟踪与定位。

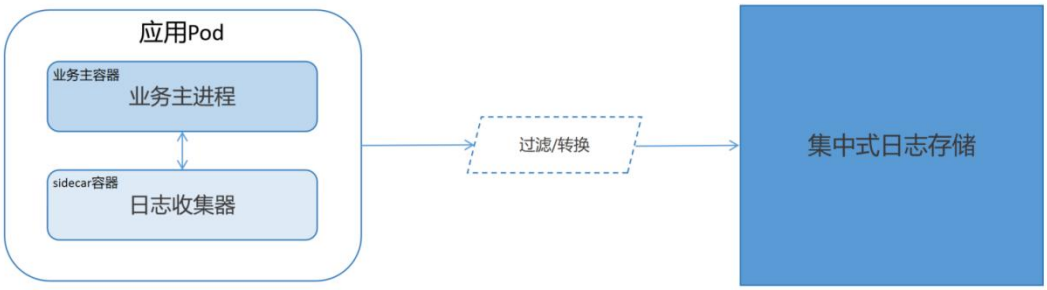


图 10：kubernetes 中常见日志收集方式

4、声明式的 API

在云原生中，API 都是声明式的，声明式会为大数据组件带来两个好处:

资源抽象程度高：如果组件需要一个 1G 容量大小的块存储，不用运维人员感知然后手工去创建和配置，只要在 API 中声明你需要的配置即可，整个流程都是自动化的；

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
spec:
  storageClassName: standard
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

图 11：kubernetes 声明式 API 样例-存储

面向终态：声明式 API 是不断向着终态逼近的，假如我们声明了一个服务包含 3 个副本，如果其中某几个副本意外退出了，声明式 API 会及时发现并且迅速拉起一个同样副本，始终保证服务的副本数

的稳定，确保服务的整体可靠性。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deployment
spec:
  replicas: 3 # 副本数
```

图 12：kubernetes 声明式 API 样例-计算资源副本数

（二）云原生技术提升集群资源使用率和弹性

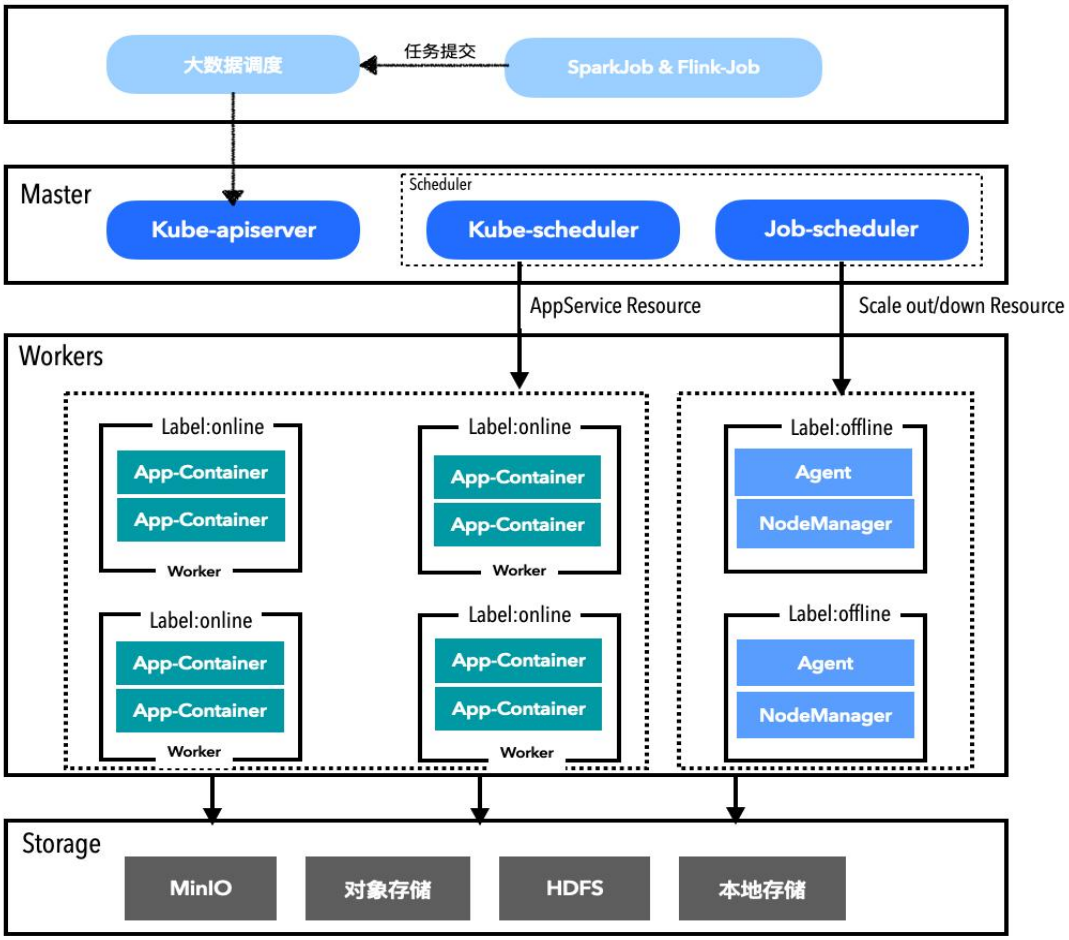
1、混部

云原生技术中，将整个基础资源层抽象成了一个大的资源池（计算、存储、网络等资源的组合），被所有大数据组件所共享。组件可以按需从这个大资源池中申请资源，这样避免了资源的重复建设，减少了资源浪费。同时这些资源池可以承载不同类型的大数据组件，如批处理、流处理、OLAP 等，加上云原生提供的多种调度技术，可以将不同大数据组件一起混部，结合算法模型，可以预测各个组件的流量峰谷，从而形成了资源的分时复用，整体上提高了总体资源利用率。云原生提供的单机隔离技术 Cgroup、KVM、Kata 等可以有效的防止混部进程间的相互干扰，使得混部更加安全。

大数据混部主要指离在线业务的混部，有如下几种常见思路：

（1）在线离线业务完全容器化混部

在该种思路中，新业务完全使用 K8S 调度，同时兼容在线集群，离线和在线大数据业务根据分时调度的策略，选择不同集群/节点运行。整体架构如图所示：



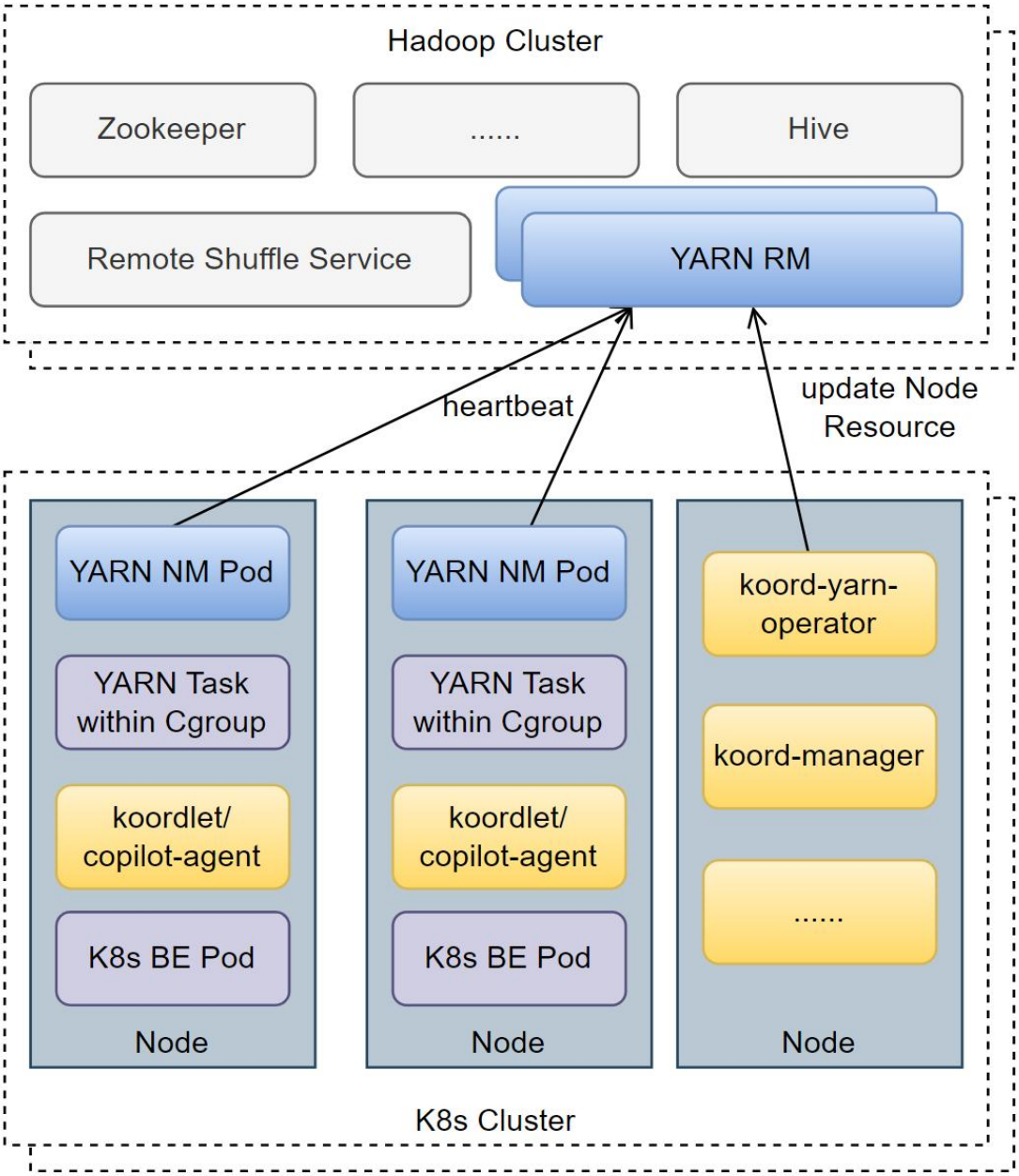
来源：https://xie.infoq.cn/article/812c3183d4a2507892534de76

图 13：在线离线业务混部示意图

该思路的优点是管控集中，统一 K8S 调度管理，便于实现在线离线任务的灵活调配。缺点是目前社区大数据方案还不够成熟，旧的基于 Yarn 的大数据作业的兼容性无法保证，需要企业自身具备较强的大数据及云原生技术实力，且无需考虑历史作业的兼容。

（2）Hadoop Yarn on Kubernetes 模式

在该种思路中，作业提交入口依然为 YARN 保持不变，整个计算资源的调度还是由 YARN RM 来执行，计算任务还是运行在 Yarn Node Manager 中。整体架构如图所示：



来源：https://koordinator.sh/

图 14：Hadoop Yarn on Kubernetes 集群模式混部示意图

从上面架构图中，可以看到大数据集群和容器集群依旧是相对独立的，容器集群中会启动一个 Yarn Node Manager（以下简称为 Yarn NM）的 Pod，这里的 Yarn NM 会自动连接到大数据集群的 Resource Manager 服务。这样的话，所有的业务代码和提交方式都不需要改变，等同于单独扩容了几台 Yarn Node Manager。该思路的优点是计算任

务层无感知，兼容性好。缺点如下：

- 离线集群可扩展差，Yarn 镜像中需绑定大数据集群的 Host 列表。
- 会有额外的资源消耗，Node Manager 占用部分服务资源。
- 需要高版本的 Yarn，低版本 Yarn 不支持 Liunx Container Cgroup 设置，可能会对在线资源有影响，需要考虑好 Cgroup 目录划分和驱逐策略。
- 额外的节点 Yarn 标签管理工作。

该种思路适用于大部分企业场景，改造成本更低、能够较好兼容现有大数据集群、效果明显。

除此之外，会有企业在上述两类混部思路的基础上，实现更多组合优化方案。

2、弹性伸缩

云原生技术栈提供了 HPA、KEDA 等弹性扩缩容技术，结合统一监控收集到的组件业务指标和性能指标，可以实现大数据组件的资源弹性伸缩。在业务高峰期时，按需动态增加资源容量（CPU、内存、存储等），当业务低谷时，动态减少资源容量，甚至做到当业务无流量时，将资源容器缩容到 0，从而进一步提高整体资源的弹性和利用率。同时由于容器的轻量级，弹性伸缩的速率很高，通过镜像预热等技术，可以更快的提高弹性扩缩容的速率。

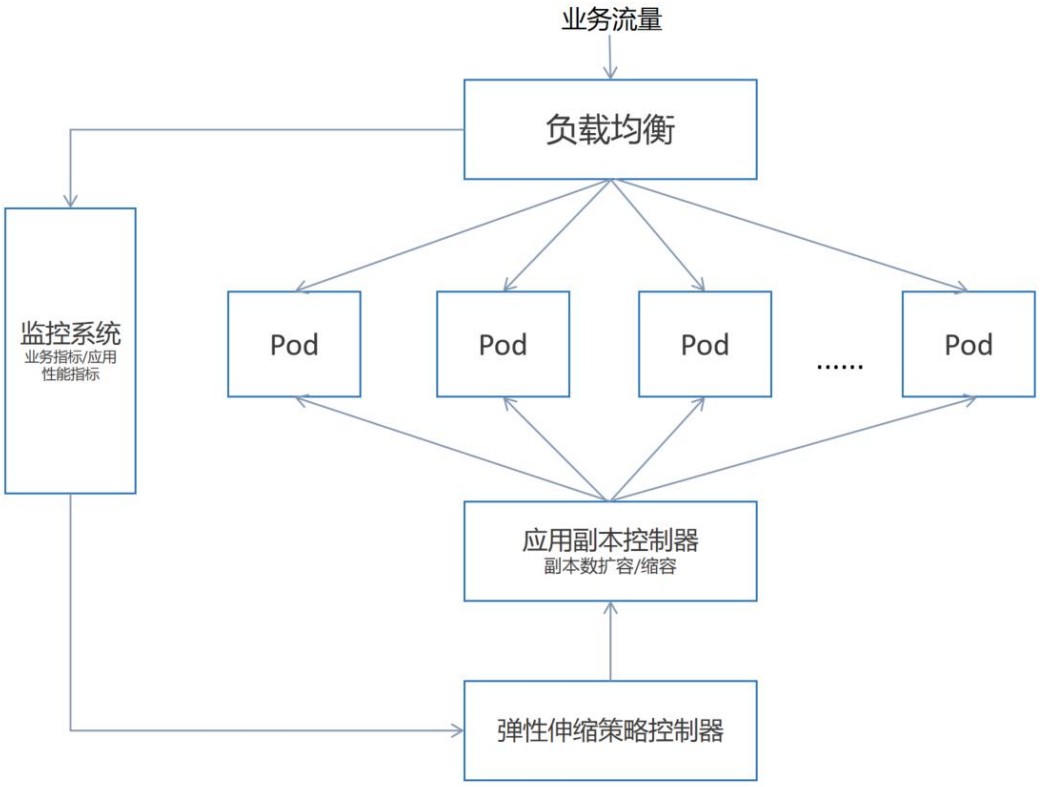


图 15：弹性伸缩示意图

（三）云原生技术提升大数据平台迭代效率

利用云原生 DevOps 和持续交付的特性，基于现在比较流行的 GitOps 方法论，可以方便得做到 IaC(Infrastruce as code)和 CaaS (configuration as a service)：将代码和配置存储在 Git 中，Git 中的每一次提交会产生一个新的 CommitID，所以天然具有版本的概念。当 Git 中的文件发生变化，会产生一个新的版本(CommitID)，同时通过 CI/CD 机制可以触发对这个版本的测试，当测试通过，会进行打包和部署；部署时可以结合灰度发布和服务网格等技术实现 A/B test，金丝雀发布安全发布方案。整体上提高从研发、验证到决策的效率。

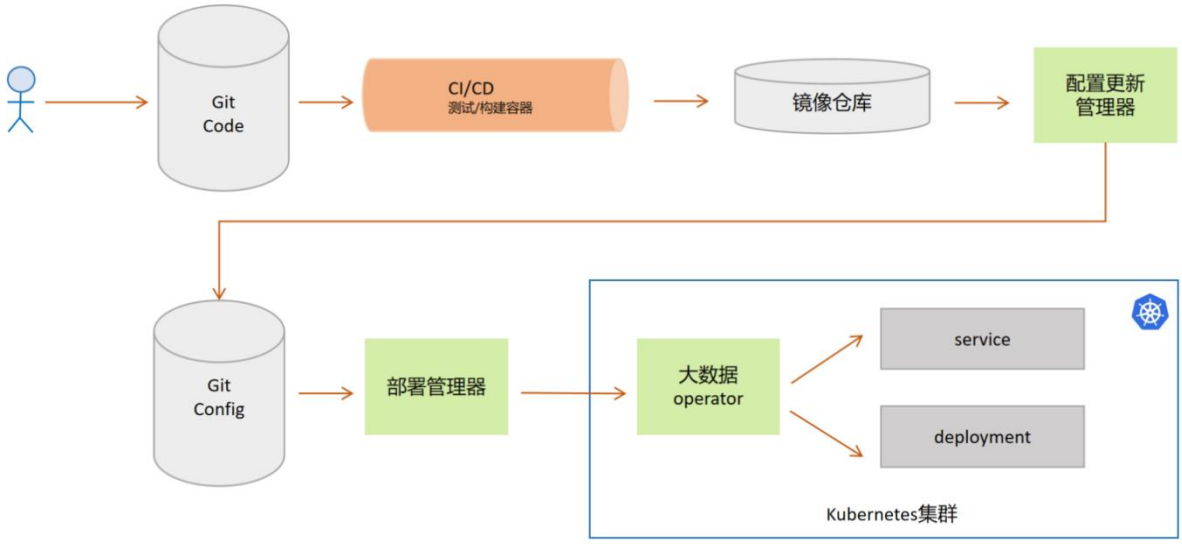


图 16：GitOps 流程示意图

常见的云原生发布策略有滚动更新与回滚、蓝绿发布、金丝雀发布策略，这些策略可以使应用发布的过程中实现应用的平滑升级，以减少对业务的影响。

- 滚动更新：确保应用程序在更新过程中保持可用。
- 滚动回滚：允许回滚到之前的版本。
- 蓝绿部署：一种在新旧版本之间进行切换的策略。
- 金丝雀发布：一种渐进性更新策略。

相较于传统的大数据和分布式系统发布，云原生的容器发布提供了一种更为灵活、轻量、可移植性和隔离性的发布形式，使得大数据和分布式系统的部署和管理更为方便。

（四）云原生技术提升大数据安全和隐私保护

1、数据隔离与多租户安全：

云原生技术栈中，有健全的多租户隔离机制来隔离不同用户的资源，防止不同用户间的恶意干扰。通过虚拟集群技术可以实现云原生

API 使用层面上的隔离、通过 Kata 等安全容器技术，可以实现容器运行时的强隔离，防止普通容器的逃逸问题发生；通过 VPC 等虚拟网络技术实现网络层面的隔离，防止端口扫描等恶意探测行为；云存储可以实现存储介质和 IOPS 上的隔离。

多租户隔离可以分为硬隔离和软隔离，硬隔离为每个租户底层单独分配一个集群，从而达到隔离不同租户的目的，适用于为企业外部客户服务的场景，软隔离是多个租户共享底层计算，存储，网络资源。云原生强大的生态能力在网络层面，数据层面，控制面提供了多种隔离解决方案，多租户软隔离=网络隔离+数据隔离+权限控制。

网络隔离：VPC 网络（Kube-router），Network Policy（Calico，Cilium）。

数据隔离：安全容器 Kata，运行时 Runc Cgroup, Resource Quotas, Limit Range, Request/limit, Namespace。

权限隔离：基于 RBAC 的访问控制+鉴权+授权

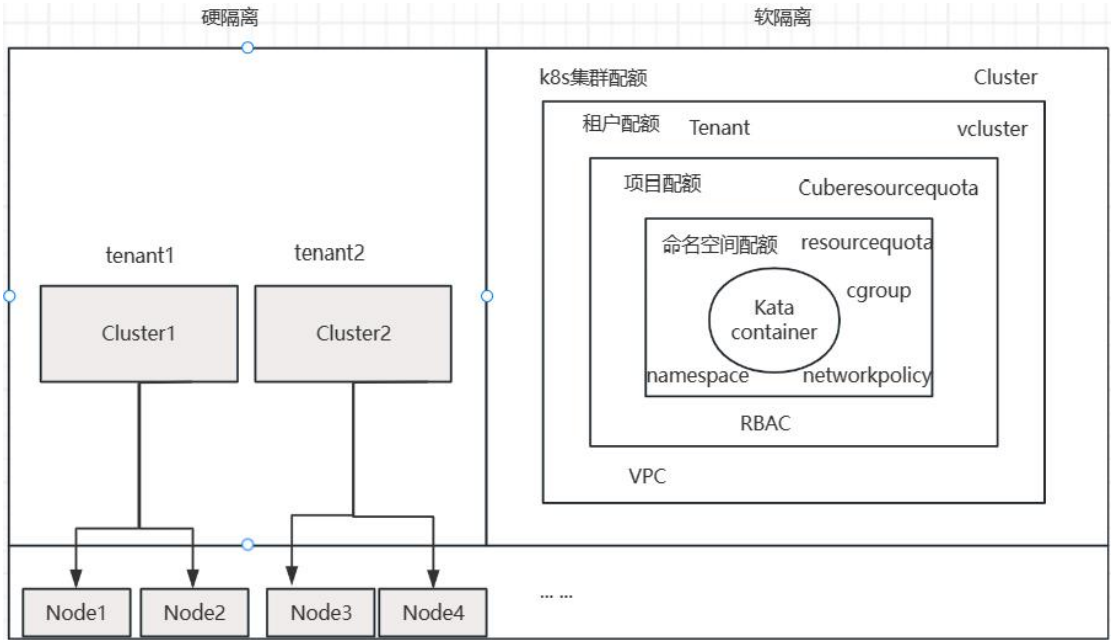


图 17：云原生安全隔离技术图示

2、网络通信安全：

K8S 内部微服务和 K8S 的通信使用非对称加密方式，服务之间通信使用证书/SA，具有双向认证功能，K8S 内针对 K8S 对象的访问也基于 RBAC 准则，即访问控制+鉴权+授权，针对不同的用户限制了对集群资源的访问范围和权限，极大的减少了由于越权访问带来了安全问题。

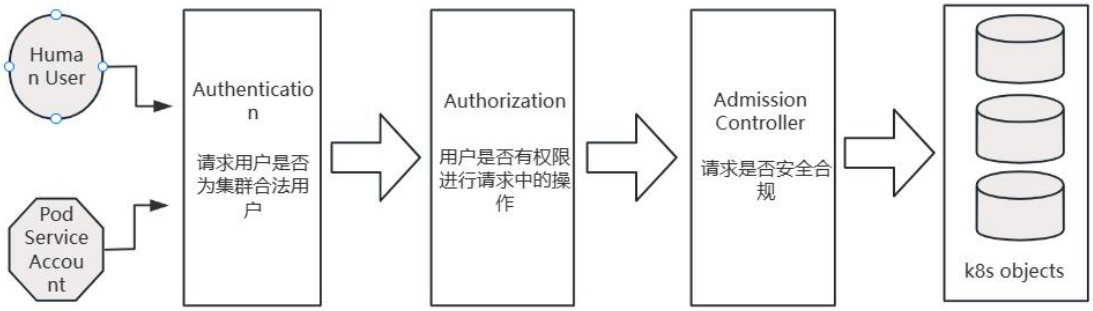


图 18：双向认证与权限控制

3、数据加密与保护：

想要实现端到端的数据加密在传统大数据平台上实施起来较为复杂。开源的方案有 HDFS 加密，TLS/SSL 用于数据传输等，但加密配置可能复杂，且在某些场景下可能影响性能。而云原生解决方案可以提供内建的、透明的加密功能，能够在性能较小损耗的情况下实现端到端加密：

数据完整性:云原生存储中，会通过校验码等机制来验证数据在传输和存储过程中没有损坏和篡改，保证数据的完整性。同时云原生技术栈中可以使用 RAM/IAM 等鉴权系统对资源进行精确的访问控制，可以细粒度到 PV 级别。同时当数据不用时，云存储会自动擦除

云盘上的数据，防止残留。

数据机密性:云原生技术栈中，从数据传输、计算、到存储都有完整的全链路加密方案，数据在网络传输时可以采用 Https、SSL、VPN 等加密，使用 HSM(密钥托管)等组件来实现证书的获取和管理；在数据的处理和计算过程中，可以采用数据脱敏技术对隐私数据进行隐藏，同时可以采用隐私计算等软硬结合技术实现安全沙箱容器，在内存级别实现对隐私数据的加密和隔离。在数据存储时，可以选择不同算法对数据进行加密存储，算法可以在申请 PV(Persistent Volume, 持久化存储)的时候进行指定。下图是全链路加密的示意图:

图 19：全链路加密示意图

4、审计与合规性：

云原生解决方案通常可以提供更全面和集成的审计日志功能，以及与合规性标准（如 GDPR、HIPAA）的更紧密集成。这些平台能够自动记录更详细的操作日志，并提供更高级的数据保护和隐私控制功能，从而更有效地满足复杂的合规性要求。

5、灾难恢复与数据备份

在灾难恢复和数据备份方面，云原生解决方案可以提供自动化的、高效的备份和恢复解决方案：

云原生技术基于微服务架构，每一个微服务一般都是多活架构，微服务架构将整体组件的复杂性进行了解耦，避免了服务的单点故障，提高稳定性和容错性；

云原生存储在技术实现上都是多副本机制，可以避免少量副本意外丢失造成的数据不可用情况，保证了数据的高可用性；

云原生技术栈中自带的负载均衡、服务发现、共享存储等机制，可以更好的实现组件的同城双机房，两地三中心等容灾部署。

6、丰富的安全工具提升网络安全

在网络层面的安全防护是传统大数据平台的薄弱环节。云原生解决方案有着丰富的安全类云产品或工具，如 WAF、安全组、网络 ACL 等来对组件的安全进行防护，实现防止网络攻击、异常行为检测、异常代码扫描等。

（五）云原生技术带来的其它好处

除了解决上述传统大数据平台的痛点，云原生技术与大数据平台融合，还带来了以下好处：

1、对跨平台的支持更加简便

云原生架构的大数据平台对于适配不同的 CPU 架构和操作系统来说会更简便，得益于如下两个特点：

首先，云原生技术栈及其配套工具普遍采用 Go 语言开发。作为一种编译型语言，Go 提供了优秀的交叉编译能力和对不同平台的原生支持，这降低了针对多操作系统的适配和部署难度。其次，大数据

组件在云原生环境中通常以容器形式运行。容器技术的不可变性和隔离性能有效地屏蔽了底层操作系统差异，加快了在不同环境中的适配速度。这两大特性共同增强了云原生大数据平台的跨平台能力，提升了其灵活性和可移植性。

2、更便于实现混合云、多云架构

随着公有云的更加普及，未来企业会有更多的混合云、多云的需求，遵循云原生规范的基于 Kubernetes 的大数据云原生架构，将极大地促进跨多云和混合云部署的便利性。这主要得益于以下几个方面：

- **统一的管理和编排：**Kubernetes 提供了一个统一的平台来管理和编排容器化的大数据应用。无论是在公有云、私有云还是混合云环境中，Kubernetes 有希望提供一致的操作体验，简化跨环境部署和管理。
- **灵活的资源调度：**Kubernetes 强大的资源调度能力使得在不同云环境中部署和扩展大数据应用变得更加灵活和高效。我们可以尝试根据应用的资源需求和可用性要求，智能地在多云或混合云环境中分配和优化资源。
- **容器化的一致性：**由于大数据组件在 Kubernetes 平台上以容器的形式运行，这保证了应用在不同云环境中的一致性和可移植性。容器化的应用更好地在多个云环境之间迁移，无需针对特定云环境进行重大修改。
- **网络策略和安全：**Kubernetes 提供了强大的网络策略和安全机

制，这有助于在多云和混合云环境中保护数据和应用。通过这些策略，可以有效地隔离和保护跨云环境中的网络流量，确保数据安全和合规性。

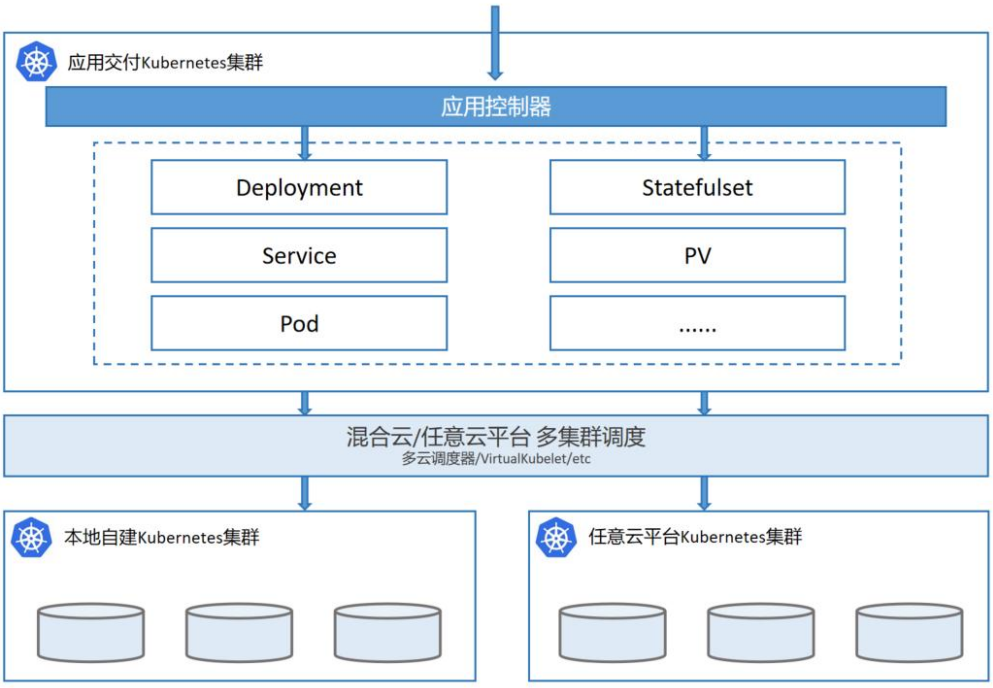


图 20：多云/混合云部署示意图

3、更便于与 AI 平台融合

传统的大数据平台和 AI 平台往往存在一定程度的割裂，往往是两个独立的团队，它们依赖于不同的技术栈。大数据平台通常基于 Java 技术栈，而 AI 平台则更倾向于使用 Python 技术栈。这种技术栈的差异在一定程度上导致了两个领域之间的协作障碍。然而，随着云原生架构的普及和采纳，这种局面正在发生变化。

- **技术栈的统一：**云原生架构提供了一种更加统一的方法来构建和部署应用，无论它们是基于 Java 还是 Python。在云原生环境中，不同技术栈的应用可以容器化，并在统一的平台上进行管

理和调度。这极大地简化了不同技术栈之间的集成和协作。

- **灵活的服务交互：**云原生架构通常采用微服务的方法，这允许不同的服务（无论是大数据处理还是 AI 模型）作为独立的组件被开发和部署。这种方法不仅提高了系统的灵活性和可维护性，还促进了不同技术栈之间的互操作性。
- **统一的数据开发能力：**大数据平台在处理和分析大规模数据方面的能力可以直接支持 AI 应用。通过在同一云原生平台上紧密集成，AI 模型可以更方便地访问和处理存储在大数据平台上的数据，从而提高数据处理的效率和效果。
- **统一的管理运维：**云原生平台如 Kubernetes 提供了高度灵活的资源管理和调度能力，可以同时纳管 CPU 和 GPU，更好的支撑大数据和 AI 作业。在云原生环境中，无论是大数据应用还是 AI 应用，都可以利用相同的 CI/CD 流程、监控工具和运维实践。这种统一性减少了跨技术栈协作的复杂性，使得团队可以更专注于业务逻辑的实现，而不是环境的差异。

通过采用云原生架构，大数据平台和 AI 平台之间的融合变得更加自然和高效。这种融合不仅打破了技术栈的障碍，还为企业提供了一个更加灵活、协同和创新的环境，以支持其数据驱动和智能化的业务需求。

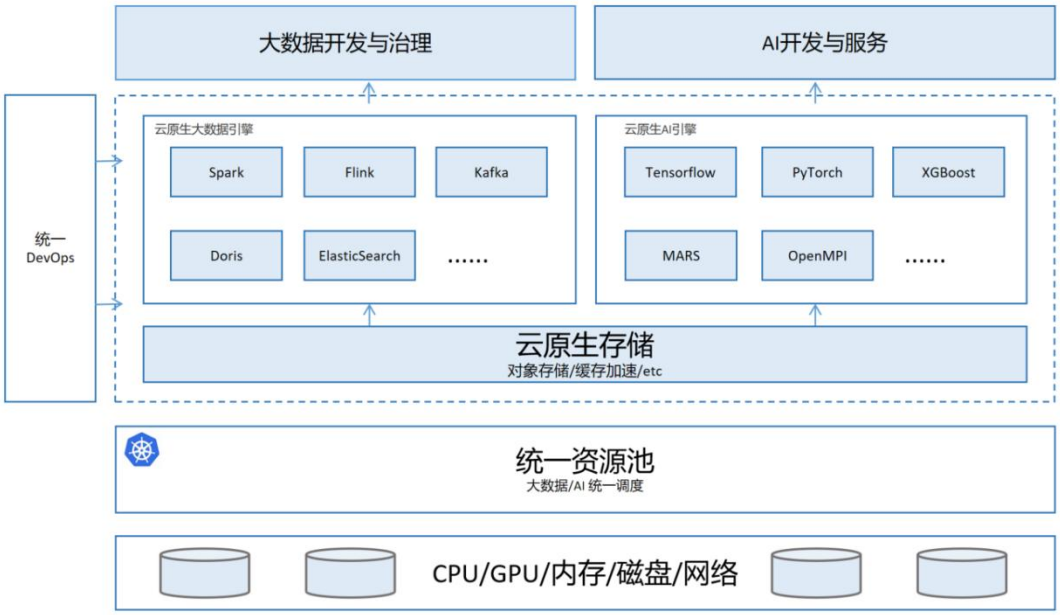


图 21：大数据/AI 混合部署

（六）大数据云原生引入的新挑战

借助云原生技术对大数据平台带来好处的同时，也引入了一些新的挑战，需要业界共同去解决，简要列举如下：

1、如何提升大数据作业调度的稳定性和性能？

由于大数据的离线计算跟传统云原生应用的场景相比，对调度的时效性和并发度要求都会比较高，传统的调度机制难以满足这种动态和大规模的资源管理需求，因此需要更符合大数据场景的调度方案，可以考虑以下措施：

(1) 更为智能和灵活的调度算法

最为关键的是开发更为智能和灵活的调度算法，例如基于 Kubernetes 的自定义调度器，能够根据作业的特性和资源的可用性进行优化调度。同时，通过实现高级调度功能，如对于大数据批处理作业来说，针对 All-Or-Nothing 类型任务的 Gang Scheduling，确保相关

任务能够同步执行，减少资源的闲置和作业的等待时间。在实现 Gang Scheduling 方面，有一些开源项目比如 Volcano、YuniKorn 等提供了优秀的案例。Volcano 是一个为云原生环境设计的高性能批处理系统，专注于优化大数据和 AI 工作负载的调度，而 YuniKorn 则提供了轻量级且适应大规模和异构环境的调度解决方案，同时支持多租户和容量调度。这些系统将 Gang Scheduling 的概念与自身特有的调度策略相结合，为云原生环境中的大数据作业提供了更为稳定和高效的调度解决方案。

(2) 针对不同作业类型的精细化调度

在 Kubernetes 环境中，根据作业的特性和需求，作业被分为不同的 Quality of Service (QoS) 类型，如 Guaranteed, Burstable, 和 BestEffort。这种分类允许系统根据作业的重要性和资源需求进行更细致的资源分配。例如，对于关键任务，可以保证资源的稳定供应，而对于非关键任务，则可以在资源紧张时进行调整。这种策略有助于在保障关键作业稳定运行的同时，提高整体系统资源的使用效率和灵活性。

(3) 分布式调度支持大规模集群

随着集群规模的扩大，单个调度器可能成为性能瓶颈。分布式调度系统，如 Kubernetes 中的多调度器架构，允许将集群资源分区分组，并进行并行调度。这种方法降低了单个调度器的压力，提高了调度的吞吐量，减少了作业的等待时间，从而提升了大数据作业的运行效率。

(4) 多集群联邦调度

多集群的联邦方式，整合不同部门不同组织的集群资源。根据不同集群中业务实际运行情况，跨集群进行资源共享（主要包括资源空闲时借出及资源紧张时回收等），既提升多集群资源使用效率，同时又保证高优先级大数据作业的及时响应。

2、如何做好旧系统兼容性，降低旧作业迁移成本？

传统的大数据应用大多是运行在 Hadoop 之上的，资源管理方面是 Yarn，存储对接的主要是 HDFS，而云原生的大数据平台，资源管理变成了 K8S，存储可能是对象存储，这导致将旧系统迁移到云原生平台时，需要进行大量的改造和适配工作，增加了成本和复杂性。

为了降低迁移成本，可以考虑提供与旧系统兼容的接口和服务，如支持 Yarn API 的云原生调度器。同时，开发与传统存储系统（如 HDFS）兼容的解决方案，确保无缝迁移，减少对现有工作流的影响。目前有一些开源的框架可以支持文件格式转换等工作，比如 Alluxio，JuiceFS 等。同时也需要针对各自具体的场景，开发一些辅助迁移工具；

3、如何解决云原生场景下中间临时数据存取的问题？

大数据计算场景下有很多中间临时数据，也就 Shuffle 数据。传统的 Shuffle 机制可能因为跨节点动态资源调度的特性，在资源重新调度后无法读取前面 Shuffle 的数据，造成任务异常。

解决这一问题的一种方法是指定 Shuffle 存储，防止数据随着 Pod 的失效而丢失；业界比较流行的演进方案是采用新型的 Shuffle 处理技术，如 Remote Shuffle Service(RSS)，来优化在云原生环境中的数

据传输和处理流程。通过这种方式，可以有效提高数据处理的效率，降低延迟。

4、如何缓解存算分离对大数据组件的影响？

云原生环境中普遍采用的存算分离架构，数据普遍会考虑存在成本更低的对象存储中，对数据访问和处理模式提出了新的要求。在这种架构下，传统的大数据处理组件可能无法高效处理远程存储的数据，影响整体性能。

针对这一挑战，需要优化大数据处理组件，比如改进 Spark 和 Hive 等计算引擎的数据缓存机制，使用 Alluxio、JuiceFS 等中间缓存层，以及 Hudi、Iceberg 等表格式存储，实现更适合存算分离场景的统一元数据管理，以提高对远程数据的处理效率。

5、Hadoop 需要进行云原生改造吗？

Hadoop 的关键组件 HDFS 和 YARN，在云原生场景下，都有相应的平替，比如对象存储和 K8S，因此 HDFS 和 YARN 本身的容器化，相比 Spark、Flink 等计算引擎，一直是一个比较低调的话题，这是否意味着，我们不需要考虑 Hadoop 的云原生改造了？ 这个问题目前没有标准答案。

然而，我们还需具体场景具体分析，在国内大量的私有化大数据场景下，Hadoop 本身的可靠性和兼容性在较大规模的场景下已经经过了大量的检验，对于很多用户来说，可靠性和兼容性大于一切，Hadoop 依然有其存在的价值，私有化场景下的对象存储也会失去公有云即开即用的优势，与 HDFS 在哪些场景下孰优孰劣也是一个值得

探讨的话题。在传统的 Hadoop 大数据平台到完全 Serverless 化的云原生大数据平台之间，我们也需要讨论是否需要过渡方案，以期望通过较少的投入，获得部分云原生技术的红利。目前开源社区有一些 Hadoop On K8S 的方案，也有围绕 Hadoop 体系演进的开源对象存储方案，同时也有云厂商推出了兼容 HDFS 的 Serverless 云原生方案。

6、传统大数据平台如何向云原生演进？

大量的企业未来都会面对如何从传统的大数据平台向云原生演进的问题，在从传统架构过渡到云原生架构的过程中，我们希望既能缓解大数据平台的问题，又尽量降低迁移过程中的改造成本，规避迁移风险，还需要考虑两种架构的共存问题。

目前这一问题还没有标准的解法，业界一些常见的经验是，为解决这一问题，可以采用分布式迁移策略，研究相对平滑渐进的解决方案，目前也有一些厂商，在大数据云原生演进方面进行了一系列探索，也希望业界积累更多的经验和共识。

7. 引入 Kubernetes 的复杂度与不可靠性如何权衡？

Kubernetes 的引入虽然为云原生大数据环境带来了诸多优势，如资源的动态管理和服务的自动化部署，但同时也带来了更高的系统复杂度和潜在的不可靠性风险，排查大数据故障需要同时涉及大数据知识和云原生知识。

在不同场景下，还需要用户根据实际需求进行权衡，为了有效利用 Kubernetes 的优势，同时控制风险，需要实施综合的系统监控、自动化运维和容错设计。

四、大数据云原生技术的架构简述

（一）云原生大数据平台的架构原则

1、弹性伸缩与资源隔离

针对不同的大数据组件以及同一组件的不同租户，要利用多租户技术与编排技术做到相互隔离，使它们可以安全地运行在各自的资源池内，并且不同组件要能实现弹性的扩缩容来应对业务的波峰波谷以提高资源利用率。

2、容器化与统一资源调度

假设是完全的大数据云原生方案，所有大数据组件要容器化，接入云原生底座操作系统 Kubernetes。通过可插拔调度体系对底座资源池进行统一的、细粒度调度。

- 实现在离线业务混部，提高资源利用率。
- 根据业务优先级来分配对应的资源，确保资源不发生不合理竞争。
- 通过不同的专业调度器实现 Gang Scheduling、Capacity Scheduling 等不同的调度效果。

3、多种计算引擎联合管理

云原生大数据平台要有统一的页面和接口管理所有的云原生大数据组件，包括管理所有大数据组件的 Operator、统一为所有的大数据组件申请需要的云产品资源、为每个组件对接监控和日志等。

4、统一元数据管理

无论从客户需求出发，还是从云原生数据架构的需求来看，需要具备在基于云计算分布式存储之上的独立统一元数据管理体系，统一管理结构化与非结构化的多种数据格式，以及对这些数据的访问权限控制，来支持多种计算引擎及上层需求。

这里云原生技术和湖仓一体技术是相互融合的关系，共同构建云原生湖仓一体数据平台。

5、智能化运维

Serverless 极大了降低了大数据平台的运维难度，但平台智能化的边界还可以向外延伸，结合丰富的历史统计信息、作业运行信息、查询信息等，利用 AI 算法可以更好得做到提前预测流量高峰，智能作业调优和诊断，以及智能 Agent 等。

6、可靠性与容灾

利用云资源做好数据和服务的备份和冗余，做到同城和异地容灾。

（二）云原生大数据平台的参考架构

各企业根据自己的场景，会有不同的架构，下图是结合了一种湖仓一体与云原生大数据平台的一个可能的架构设计，仅供参考：

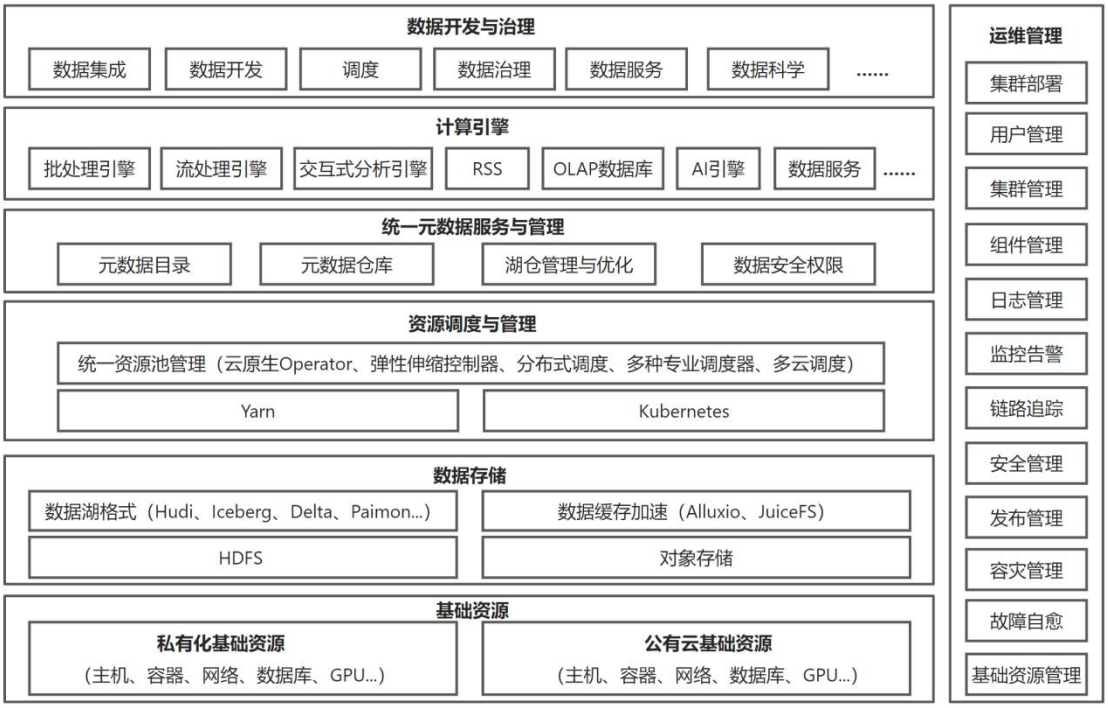


图 22：云原生湖仓一体数据平台参考架构

- **基础资源层：**部署大数据平台时依赖的基础资源，主要包含主机、容器、网络、数据库、GPU 等基础资源。这里分为公有云、私有云和混合云等情况。公有云部署可以采用云厂商售卖的半托管或全托管的 Kubernetes 服务，私有云部署可以使用自己部署的开源 Kubernetes 服务，而混合云客户可能同时拥有公有云和私有云的资源。

- **数据存储层：**主要包括 HDFS 存储与对象存储，以及基于之上构建的数据湖格式（Hudi、Iceberg、Delta Lake、Paimon 等），以及可能需要的数据缓存加速层（Alluxio、JuiceFS 等）。

- **资源调度与管理层：**对大数据组件进行统一调度，这一层可利用 Kubernetes 的抽象能力，将所有的基础资源抽象成统一的资源池，有些场景下依然需要依赖 Yarn，在此基础上为上层提供统一的资

源池管理，包括云原生 Operator、弹性伸缩控制器、分布式调度、多种专业调度器、多云调度等。

- **统一元数据服务与管理层：**存储引擎通常通过统一元数据服务，为上层的计算引擎提供统一的数据访问接口。元数据目录一般是 Hive Metastore 或者兼容 Hive Metastore 的元数据服务，对于更大量，丰富的分析型元数据需求，可以考虑构建元数据仓库，通过湖仓管理服务来解决湖仓一体的自动化治理难题，通过统一数据安全权限模块来体系化地提供数据安全权限服务。
- **计算引擎层：**包括大数据批处理引擎（Spark 等），流处理引擎（Flink 等），交互式分析引擎（Trino、Impala 等），远程混洗服务 Remote Shuffle Service， OLAP 数据库（Doris、StarRocks 等），AI 引擎（Pytorch、TensorFlow 等）以及数据服务引擎（Elastic Search、HBase 等），这些引擎都可以使用云原生 Operator 的方式进行部署和管理。
- **数据开发与治理层：**为用户在大数据引擎基础上提供的各种上层服务，主要包括数据集成，数据开发，调度，数据治理，数据服务，数据科学等服务。
- **运维管理平台：**负责整个大数据平台的部署和运维，包括集群部署，用户管理，集群管理，组件管理（部署、升级、下线），可观测性设施构建(Log/Metrics/Tracing)，安全管理，发布管理，容灾管理，基础资源管理等。

五、大数据云原生的未来发展和战略建议

（一）技术发展方向

当前大数据云原生技术领域涌现出一系列发展趋势，简述如下：

1. **多云和混合云战略：**企业越来越倾向于采用多云和混合云战略，将工作负载分布在不同的云服务提供商和本地数据中心之间，以避免被单供应商锁定，提高弹性和可用性。
2. **自动化和智能化运维：**云原生技术将更深入地实现自动化和智能化运维。通过整合自动化流程、自愈能力以及基于 AI 的运维策略，大数据云原生技术将帮助企业更有效地管理和维护其大规模的数据基础设施。
3. **数据治理架构变革：**随着数据规模的增加，以及多云环境的普及，数据的迁移成本，可观测性和有效的数据治理变得至关重要，Data Fabric 等数据治理架构和方法论正逐步被行业关注，企业需要更强大的工具来处理、监管和保护其数据资产。
4. **AI 与大数据的融合：**人工智能和大数据技术的融合将推动更高级的数据分析和智能决策。

（二）针对行业的建议

随着大数据和云原生技术的快速发展，国内在这一领域的探索和实践仍处于早期阶段。为了促进大数据云原生的健康发展，以下是针对整个行业的一些建议：

1. **建立更广泛的探讨：**鼓励信通院这样的权威机构牵头，组织行

业内的专家学者、企业领袖和技术先锋，共同进行更广泛和深入的探讨。

2. **产出专业研究报告和技术白皮书：**定期发布关于大数据云原生技术和产业的研究报告，以及相关技术的白皮书。这些文档应涵盖最新的行业趋势、技术进展、最佳实践和案例研究，为行业提供指导和参考。
3. **关注国内特殊需求：**针对国内市场的特殊需求，如私有化部署场景，考虑更多私有云、混合云的方向，提供针对性的解决方案和实践建议。
4. **制定统一的测评标准：**制定一套标准化、公正的测评体系，用于评估不同大数据云原生解决方案的性能、可靠性、安全性等关键指标。这将有助于提高市场的透明度，促进健康竞争。

（三）针对企业和用户的建议

对于企业和用户而言，正确地选择和实施云原生技术是实现数字化转型和提升竞争力的关键。以下是一些具体的建议：

1. **实际评估业务需求：**在跳入云原生的大潮之前，先停下来问一问：“我们真的需要这个吗？”评估你的业务场景是否真的会从云原生技术中受益，比如是否需要更便捷的部署运维、更强的扩展性或更高的资源利用率。
2. **选择合适的方案：**不同的需求会有不同的解决方案，是先不动，是完全升级，还是渐进升级，重要的是找到“对症下药”的解决方案。

3. **小步快跑，逐步迁移**：不必急于一步到位。可以先从一个不太复杂的项目开始尝试，逐渐感受和学习云原生的运作方式，再逐步扩展到其他项目。
4. **分享实战经验**：如果你在云原生转型中取得了成功，不妨写个案例分享出来，或者在行业会议上讲讲你的故事。这不仅能提升你的品牌影响力，也能帮助同行少走弯路。
5. **关注实际的安全和合规问题**：安全不是摆设，合规不是噱头。确保你的云原生方案一开始就能满足行业安全标准，有时候后面改会有较高的成本。
6. **持续监控，关注成本和效益**：定期检查你的云原生环境是否真的提高了效率，降低了成本。有时候，一些看似高效的技术实际上可能是金钱和时间的黑洞。

六、参考文献

在编撰本报告的过程中，我们深入研究了众多国内外大数据云原生领域的资料。在此基础上进行了综合性的分析和融合，加入了自己的理解。感谢所有原作者和专家的贡献，为我们提供了宝贵的信息和启发。最后感谢大数据云原生资深专家，开源 CloudEon 作者星河与 KubeData 作者刘彬，利用业余时间帮助审稿。由于时间仓促，水平所限，错误和不足之处在所难免，欢迎各位读者批评指正，让我们一起努力，共同推动大数据云原生的技术普惠。

[1] 信通院 - 湖仓一体技术与产业研究报告
[2] 信通院 - 云原生湖仓一体白皮书（2022 年）
[3] CNCF - 云原生技术官方定义：
https://github.com/cncf/toc/blob/main/DEFINITION.md
[4] CNCF - 云原生 Landscape：https://landscape.cncf.io/
[5] 大数据云原生的探索与实践：
https://cloud.tencent.com/developer/article/2222797
[6] 大数据云原生的现状与趋势：https://zhuanlan.zhihu.com/p/374842573
[7] 云原生时代的大数据技术演进：https://developer.aliyun.com/article/1309942
[8] 云原生大数据平台的构建思路：https://zhuanlan.zhihu.com/p/616335169
[9] 云原生大数据架构实践与思考：https://developer.aliyun.com/article/1309942
[10] 大数据系统云原生渐进式演进最佳实践:
https://mp.weixin.qq.com/s/Ely4Nme02Ks8SeyqXp9sgQ
[11] 湖仓一体 2.0：数据分析的终局之选：http://www.oushu.com/post/134
[12] Pepperdata - 2021 Big Data on Kubernetes Report：
https://www.pepperdata.com/wp-content/uploads/dlm_uploads/2021/12/2022-Big-Data-on-Kubernetes-Report.pdf
[13] 传统大数据平台如何进行云原生化改造：
https://www.infoq.cn/article/bzikh26uftd4x41jgj7y
[14] Starburst - Presto 在 2019 年 8 月宣布支持 K8S：
https://www.starburst.io/blog/presto-on-kubernetes/
[15] Apache Flink - Flink v1.10.0 发布 Native Kubernetes beta 版：
https://flink.apache.org/news/2020/02/11/release-1.10.0.html
[16] Datanami - Hive 探索 MR3 运行在 Kubernetes 上：
https://www.datanami.com/2020/02/18/mr3-unleashes-hive-on-kubernetes/
[17] Kubernetes Blog - Kubernetes Airflow Operator：
https://kubernetes.io/blog/2018/06/28/airflow-on-kubernetes-part-1-a-different-kind-of-operator/
[18] KubeData: 大数据离在线混部场景资源调度的演进与选型
https://mp.weixin.qq.com/s/ISQ-i_iXM7t4FtDxoUu_aQ
[19] Volcano https://github.com/volcano-sh/volcano
[20] yunikorn https://github.com/apache/yunikorn-core
[21] Koordinator https://koordinator.sh/
[22] CloudEon https://github.com/dromara/CloudEon
[23] 数据库应该放入 K8S 里吗？
https://mp.weixin.qq.com/s/4a8Qy4O80xqsnytC4l9lRg
[24] 没错，数据库确实应该放入 K8s 里！
https://mp.weixin.qq.com/s/IDsF_f7ZnB19jEu8ZtO-Nw
[25] ChatGPT：https://chat.openai.com/