北京金融科技产业联盟
BEIJING FINTECH INDUSTRY ALLIANCE

海量数据处理技术金融应用研究报告

北京金融科技产业联盟
2024 年 1 月

版权声明

编制委员会

编委会成员：

何　军　聂丽琴　胡利明　周刚慧

编写组成员：

艾轶博　鲍　玲　曹　骏　陈　明　戴大海　郭龙飞　火雪挺
江　灏　姜　江　靳悦荣　李晨宇　罗　俊　杨文龙　刘亚龙
潘熙文　佘万君　唐天辰　王　锋　王良杰　王　能　许耀栋
徐晓敏　杨景瑞　袁　一　张　昆　张敬之　张　毅　周　飞
周　允

编　审：

黄本涛　郭　栋　刘宝龙

牵头单位：

深圳市腾讯计算机系统有限公司

兴业银行股份有限公司

参编单位：

中国工商银行股份有限公司

中国银行股份有限公司

浙商银行股份有限公司

飞腾信息技术有限公司

深圳市连用科技有限公司

北京科技大学

目 录

一、发展概况 2

（一） 法律法规和政策环境 2
（二） 技术发展阶段及特征 5
（三） 技术框架与形态 9

二、应用情况 15

（一） 平台建设应用情况 15
（二） 技术应用情况 20

三、主要挑战 28

（一） 数据存储的挑战 28
（二） 数据计算的挑战 29
（三） 云化计算的挑战 31
（四） 融合计算的挑战 32
（五） 研发运营一体化的挑战 33

四、关键技术与建设思路 36

（一） 云数一体化 36
（二） 存算分离化 44
（三） 数据湖仓化 50
（四） 计算融合化 59
（五） 研发运营一体化 68

五、发展趋势和展望 78

（一） 生成式人工智能驱动数据技术方面 79
（二） 实时数据湖仓方面 81
（三） 数据网格方面 90
（四） 数据编织方面 93

六、实践案例 95

摘要:

海量数据处理是金融业大数据技术领域的关键难点，对金融业海量数据进行高效的存储、计算、分析和运营，将有效帮助金融机构深度挖掘数据的潜在业务价值，实现降本增效。现阶段，金融业在海量数据处理方面呈现出“五化”的技术趋势，即云数一体化、存算分离化、湖仓一体化、计算融合化与研发运营一体化。其中一些新的关键技术已在部分金融机构进行了较深入的实践应用，取得了可观的成果。但是，作为一项复杂的课题，海量数据处理还面临着技术、产品、应用等多方面的挑战和痛点，且这些难题当前尚未进行全面的研究和分析。因此，本报告对海量数据处理的技术、应用、建设等方面进行系统的分析，从行业发展、应用情况、落地痛点以及关键技术等多个维度展开研究，结合我国金融业多个典型案例，论证这些关键技术和实施路径的有效性和可行性，并对一些诸如人工智能、数据湖仓、数据网格等前沿数据技术应用进行初步分析，探讨金融业未来的数据技术发展趋势。

兴业数字金融服务（上海）股份有限公司为本报告的编制提供了支持。中信建投证券股份有限公司、上海汽车集团财务有限责任公司为报告编制提供了行业案例。

一、发展概况

（一）法律法规和政策环境

在金融业早期阶段，数据处理主要集中在银行和证券公司的业务数据处理，以及金融管理部门的监管工作中。相关法律法规和政策主要包括 1983 年 12 月 8 日第六届全国人民代表大会常务委员会第三次会议通过的《中华人民共和国统计法》，以及 1994 年 2 月 18 日中华人民共和国国务院令第 147 号发布的《中华人民共和国计算机信息系统安全保护条例》等。这些法规为金融数据处理提供了基础的法律保障，确保了数据的准确性和安全性。

在信息化阶段，随着信息技术的不断发展，金融业数据处理技术逐渐成熟。在这一阶段，金融业开始将数据处理技术应用于更多的领域，如互联网金融、金融风险控制等。相关法律法规和政策主要包括十二届全国人大常委会通过的《中华人民共和国网络安全法》，以及中国人民银行等十部委发布的《关于促进互联网金融健康发展的指导意见》（银发〔2015〕221 号）等。这些法规促进了金融业的信息化进程，为金融业的快速发展提供了有力的支持。

在数字化转型阶段，金融业开始迈向全面数字化。在这一阶段，金融业数据处理技术得到了更广泛的应用。相关法律法规和政策主要包括《关于推进金融科技创新发展的指导意见》（银发〔2019〕202 号）、国务院印发的《关于促进大数据发展的行动

纲要》（国发〔2015〕50号）等。这些法规推动了金融业的数字化转型，使金融业能够更好地适应现代经济的发展需求。

智能化发展阶段，是金融业数据处理技术发展的最新阶段。在这一阶段，金融业开始采用人工智能等先进技术进行数据处理。相关法律法规和政策主要包括2021年6月10日第十三届全国人民代表大会常务委员会第二十九次会议通过的《中华人民共和国数据安全法》、2021年8月20日第十三届全国人民代表大会常务委员会第三十次会议通过的《中华人民共和国个人信息保护法》、中央深改委发布的《关于促进人工智能和实体经济深度融合的指导意见》（工信部联科〔2019〕222号）、科技部等六部门发布的《关于加快场景创新以人工智能高水平应用促进经济高质量发展的指导意见》（国科发规〔2022〕199号）等。进一步法规明确了数据使用的安全合规和保护要求，为金融业数据处理技术的发展提供了良好的法律环境，促进数据处理技术的健康发展。相关政策为金融业的智能化发展提供了有力的支持，推动了金融业的创新和发展。

此外，全国金融标准化技术委员会近年陆续发布了《个人金融信息保护技术规范》（JR/T0171—2020）、《金融数据安全数据安全分级指南》（JR/T0197—2020）、《金融数据安全数据生命周期安全规范》（JR/T0223—2021）、《金融大数据术语》（JR/T0236—2021）、《金融大数据平台总体技术要求》

（JR/T0237—2021）等金融行业标准，为金融数据的处理提供了标准指引。

除了国内的法律法规和政策文件，国际上的法规和政策也对金融业数据处理技术的发展产生了重要影响。例如，2018 年 5 月 25 日欧洲联盟出台的《通用数据保护条例（GDPR）》和 2022 年 6 月 3 日美国参议院和众议院发布的《美国数据隐私和保护法》等，这些法规对金融业的数据处理提出了更高的要求，促使金融业不断提高数据处理的标准和质量。

过去一年，全球金融数据处理市场从疫情中逐步恢复，重新进入平稳增长态势，也呈现出一些新的特点：一是从区域发展来看，北美地区仍保持发展优势，东南亚及拉美地区的发展速度最快；二是从业务领域来看，数字货币、绿色普惠、数据安全等是全球各国共同关注的热点，金融基础设施的数字化升级也要求金融科技监管的国际合作水平不断提升；三是从市场主体来看，大型互联网科技企业持续强化金融数据处理市场布局，传统金融机构不断加大数字化转型投入，重回快速增长轨道。

在政策、市场和技术等多种因素影响下，国内外金融数据处理技术发展环境和产业生态都在发生着深刻变化。中国金融数据处理市场在审慎稳妥的监管环境下，市场格局也正在发生改变，传统金融机构在金融科技战略定位上正在从“科技赋能”逐步向“科技引领”转型。大型互联网平台公司金融数据处理业务在监管政策环境下，更加注重科技服务与类金融业务的隔离，类金

融业务加快获取金融牌照步伐，并不断强化自身科技属性，推进核心技术持续演进，推进金融数据处理关键技术与热点应用的规模和范围不断扩展。

（二）技术发展阶段及特征

随着技术的发展，数据的处理从最开始的纸质票据和邮件寄送，到后来的传统数据库、小型机与大型机，到如今的中大型分布式数据存储与计算集群；从依靠掌柜和经理人的经验，到如今分析师和数据科学家们基于数据、算法与算力开展协同工作，实现在风控、反洗钱，反欺诈、反社工，以及信贷、借款、用户画像、网络安全等各个场景的数据价值。

1.从传统数据库到大数据体系的变革

随着数据在金融行业中的深度应用，数据规模的不断扩大，数据类型也不再局限于关系型数据。传统数据库开始在数据处理方面力不从心，业务使用的复杂性增高、数据管理的复杂性变大、海量数据处理的时效性差、成本高。于是，为了应对上述挑战，大规模并行处理 MPP（Massively Parallel Processing，MPP）数据处理技术开始被使用，以解决数据规模带来的复杂性问题。但是数据规模持续增长、数据表达维度增多、数据类型进一步多元化等问题所带来的复杂性挑战远远超过了预期，数据处理成本高昂和数据类型支持有限变成了新的困难。于是，行业内开始采用 Hadoop 及其衍生技术作为经典大数据方案来应对新的数据处理挑战，并取得了很好的效果。

2.从处理海量文本到高价值、多维度、多类型特征的转变

随着数据的价值不断被证明，数据工具的利用也从数据科学和数据分析等专业的技术团队逐步延展到业务团队，业务分析与挖掘的需求也更加旺盛。随着需求所对应的数据类型增加，原本仅面向海量文本及结构化类型的数据特性渐渐无法满足业务需求；最终，在保持海量数据处理能力的前提下，逐步向满足高价值、多维度和多类型的数据特征快速演进。

Hadoop 体系诞生自互联，是沙中淘金的过程。随着金融业数字化转型的发展，在线业务通常采用 SDK 等方式进行埋点，数据清洗的无效计算量大大降低；通过数据压缩等方式，在性能影响微乎其微的前提下减少了 70%以上的存储空间浪费，但互联网用户仍旧是“沙中淘金”的思路。可转换到行业领域，尤其是金融业，原本的数据纯度就较高，是“金中炼金”的过程，处理过程中更多解决的是单节点无法完成计算的问题，亦或是处理速度不高的挑战；数据之“大”不再是单纯的存储规模，更是计算参与的维度之“多”。

3.存算分离需求的萌芽

数据规模与价值挖掘所需的资源之间，随着时间的推移表现出不同的关系。海量数据处理平台建设之初，所有存储的数据都会参与计算。随着数据价值的变化，参与计算的数据会逐步稳定在一定的比例，更多的数据因为合规或其他需要存储在服务器上但并不会持续参与计算。诚然，这个比例随着场景和策略的不同

而有所不同，例如对于离线数仓的场景来说，3 年以上的存储周期，参与计算的数据占总数据存储量的比例大致约为 23%左右。随着关联度和热度的降低，这部分数据的计算参与度也会随之降低。而对于行为特征类的数据，热度降低效应则会更加明显。因此，会出现存储和计算所需资源不匹配的情况。

传统的海量数据处理方案也尝试过解决存算分离的挑战，通过将提供存储能力和计算能力的相关组件角色分别部署在不同服务器节点，获得初步的存储和计算分离能力。但这样的方案引入了集群灵活性不足、运维要求高、业务应用容易造成资源耗尽等各类衍生问题与风险，并不适合作为生产环境的最佳实践。

4.易用性优化推动使用难度进一步降低

传统数据库向海量数据处理体系迁移的过程中，遇到的最大挑战便是初代数据处理体系的技术方案中，需要高级编程语言而非 SQL 语言来操作，这对方案的普适性推广造成了障碍。当 SQL 语言被全方面地融入海量数据处理体系中后，成本更低，使用更灵活和易用的技术平台才被广泛推入生产环境使用。

如今，海量数据处理平台已深度融入金融业的数据处理过程中，并从分担传统数据库 OLAP(On-Line Analytical Processing，OLAP)压力的旁路辅助角色，升级为数据中心中基础设施的核心，作为金融数字化的关键，处理近乎全量数据。而在数据开发与治理的交互方面，在满足高级编程语言支持的同时，尽可能实现支持类 SQL 兼容语法，以满足从业者快速上手和新旧技术栈快速迁

移的需求，这进一步降低了使用海量数据处理技术的难度，提高易用性，最大程度地帮助从业者发现并利用数据价值。

5.行级别的海量数据近实时更新能力需求

初代海量数据处理体系的技术方案中，为了满足大规模数据规模和读写性能需求，在底层实现中采用了“追加写”的方式，即：无论是数据新增、修改还是删除，在底层技术实现上均表现为写入一条新的数据，在后续构建离线数仓时，再进行有效的数据整理与合并，这样的方案初步解决了传统方案无法处理海量数据的挑战。

随着数据的应用场景越来越丰富、数据价值越来越重要，对数据的时效性要求也越来越高。曾经基于全量数据定时构建离线数据仓库的方式不但资源消耗巨大，在满足时效性方面也越来越受到挑战。因此需要更高效的数据组织方式，将早期方案中粗犷的数据使用与资源利用模式进行深度优化，以应对挑战。

数据湖和数据仓库的融合将构筑数据湖仓化，带来的行级别更新能力支持是很好的实践路径。

通过有效的数据组织格式，基于行级别更新能力的支持，使得之前需要全量数据参与才能实现的数据更新时效性提升到了近实时，并极大减少了资源消耗，提升了资源利用效率。

数据湖和数据仓库融合形成的数据湖仓一体架构，消除了数据湖和数据仓库之间的数据壁垒，实现了数据的自由流动，降低

了数据冗余，同时也实现了数据湖和数据仓库之间的优势互补。数据不必再进行湖仓之间的传递，极大优化了数据处理的时间。

海量数据规模条件下的近实时数据更新能力，将为业务提供更高效的数据处理支持，更好地实现业务价值。

（三）技术框架与形态

在不同的领域和行业中，对于海量数据的定义有所不同。**一般来说，“海量数据”（Massive Data）是指数据量大到用传统的数据管理和处理技术难以有效存储、管理和分析的数据集合。而海量数据处理技术，并非特指某一项技术，而是为了满足业务和行业实际需求的综合性解决方案技术栈，帮助金融机构充分利用数据，更加轻松地挖掘分析数据价值。**

1.海量数据处理技术基本形态

从外部形态上，海量数据处理技术需具备类 SQL 交互语言支持、Python 语言支持、常用如 Flink、Spark 等计算引擎支持，保持标准开放性，主要支持从 TB 至百 PB 级别的数据处理能力，延展至 EB 级数据能力规模，以应对当下和未来的持续挑战，支持存算分离，以实现按需配置，最终实现性能、需求、成本、易用性、灵活性的平衡等。如图 1 是一个典型的海量数据处理技术架构:

图 1 典型的海量数据处理架构

2.分布式存储框架

海量数据的存储通常基于分布式文件存储或对象存储，支持水平扩容，支持多种存储数据类型，提供结构化、半结构化、非结构化数据的存储解决方案。目前常用的存储框架，主要以文件存储、列式存储、对象存储三大类为主，属于图 1 的“分布式存储管理”模块，基本覆盖包括金融业在内的主要存储场景，这三者存储类型同属于大数据技术栈的底层存储层，但满足的是不同场景的存储需求，是金融业海量数据处理环节中的第一步。

HDFS（Hadoop Distributed File System，HDFS）是面向 PB 级数据存储的分布式文件系统，可以存储任意类型与格式的数据文件，包括结构化的数据以及非结构化的数据。HDFS 将导入的大数据文件切割成小数据块，均匀分布到服务器集群中的各个节点，

并且每个数据块多副本冗余存储，保证了数据的可靠性。HDFS 还提供专有的接口 API，用以存储与获取文件内容。

Ozone是大数据场景中融合文件系统和对象存储的较佳解决方案，能有效解决用户在使用过程中各类存储需求，并延续Hadoop开源存储项目的存储成本优势。生态方面支持Hadoop文件系统、对象存储/S3、本地路径挂载和K8S CSI等多种访问方式。Ozone与Hadoop生态融合，如Apache Hive、Apache Spark等无缝对接。Ozone支持Hadoop Compatible FileSystem API (aka OzoneFS)。通过OzoneFS， Hive，Spark等应用不需要做修改，就可以运行在Ozone 上。除此之外，Ozone还同时支持数据本地化，使得计算能够尽可能地靠近数据。

HBase 是一个构建在 HDFS 上的分布式存储系统，主要用于海量结构化数据存储。从逻辑上讲，HBase 将数据按照表、行和列进行存储。与 HDFS 一样，HBase 目标主要依靠横向扩展，通过不断增加廉价的商用服务器，来增加计算和存储能力。一方面，HBase 能够支持灵活的列字段定义；另一方面，HBase 利用LSM(Log-Structured Merge-Tree，LSM)数据结构模型，将数据的随机访问转换成对磁盘的顺序读写，从而实现高性能的数据随机访问。HDFS节点主要负责HBase底层存储，HDFS保证了HBase

的高可靠性。HDFS 为 RegionServer 和 Master 节点提供分布式存储服务，同时保证数据的可靠性。HBase 的架构如图 2 所示:

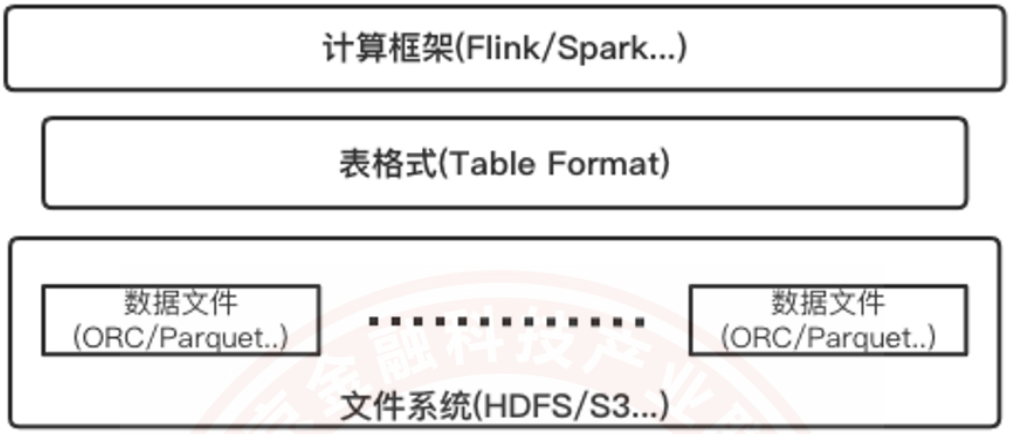


图 2 HBase 分布式存储架构

3.数据组织方式与分析技术框架核心

Iceberg是一个面向海量数据分析场景的开放表格式(Table Format)，有时也被认为是新一代的数据湖仓组件。定义中所说的表格式(Table Format)，可以理解为元数据以及数据文件的一种组织方式，处于计算框架(Flink，Spark...)之下，数据文件之上。

表格式(Table Format)属于数据库系统在实现层面上的一个抽象概念，一般表格式会定义出一些表元数据信息以及 API 接口，比如表中包含哪些字段，表下面文件的组织形式、表索引信息、统计信息以及上层查询引擎读取、写入表中文件的接口。

4.数据编排与缓存加速核心

Alluxio被认为是一种数据编排技术。它为数据驱动型应用和存储系统构建了桥梁,将数据从存储层移动到距离数据驱动型应用更近的位置从而能够更容易被访问。在海量数据处理生态系统中，Alluxio位于数据驱动框架或应用（如Apache Spark）和各种持久化存储系统（如HDFS）之间。Alluxio统一了存储在这些不同系统中的数据,为其上层数据驱动型应用提供统一的客户端API和全局命名空间。

5.消息队列

消息队列支持亿级的消息接收、中转和推送服务，可弹性扩展，无并发限制，高性能具备低延迟、高并发、高可用、高可靠等特性，可支撑亿级数据洪峰的分布式消息中间件，无缝迁移，更安全、更可靠、更易运维。

6.分布式计算框架与分析引擎

Hive把存储在HDFS之上的结构化数据抽象成关系型数据表，并提供SQL接口对数据表做查询操作。因此，用户能够以传统关系型数据库的方式来查询大数据存储系统，可以通过Hive来实现SQL查询分析。

Flink 提供高吞吐量、低延迟的流数据引擎以及对“事件-时间”处理和状态管理的支持。Flink应用程序在发生机器故障时具有容错能力,并且支持exactly-once语义。程序可以用Java、Scala、Python和SQL等语言编写，并自动编译和优化集群或云

环境中运行的数据流程序。此外，Flink 的运行时本身也支持迭代算法的执行。

Tez 是一种支持 DAG 作业的开源计算框架，它可以将多个有依赖的作业转换为一个作业从而大幅提升 DAG 作业的性能，相对于原始的 MapReduce 框架，Tez 可以一次 Map 读取，多次 reduce 操作而中间不用进行 IO 操作，从而降低频繁的文件 IO 和网络 IO，相对 MapReduce，使用 TEZ 做计算引擎性能能提高很多。

Spark 是一种海量数据并行计算框架，充分利用集群的内存资源来分布数据集，大幅提高计算性能。Spark 包含丰富的计算生态，包括 SparkSQL、MLlib 等。Spark 支持丰富的编程语言如：Scala、Python、R、Java 等等。

Presto 是一个分布式 SQL 查询引擎，适用于交互式分析查询，数据量支持 GB 到 PB 字节。Presto 的设计和编写主要是为了解决 PB 规模的海量数据交互式分析和处理问题。同时，Presto 支持多种数据源，比如 Accumulo，HDFS，Redis，PostgreSQL，MySQL 等，支持多数据源 JOIN 查询。

二、应用情况

（一）平台建设应用情况

1.技术平台上云情况

金融业海量数据处理平台上云已经成为一个不可逆转的趋势。这一趋势的出现主要是由于云计算技术的不断发展和进步，提供了更高的计算能力、更大的存储能力和更好的灵活性。因此，金融机构正在逐渐将他们的数据、应用和计算资源迁移到云计算平台上，以实现更高效、灵活和低成本的数据管理和处理。金融业海量数据处理平台因其涵盖交易、客户信息等高敏感数据，一般采用私有云部署，部署模式的发展历程可以分为以下几个阶段:

第一阶段是物理机部署，使用传统的硬件和软件资源构建管理数据处理平台。

第二阶段是虚拟化部署，通过虚拟化技术将服务器、存储和网络等资源进行虚拟化，以提高资源利用率和管理效率。

第三阶段是云化部署，将数据、应用和计算资源迁移到私有云平台上，以实现更高效、灵活和低成本的数据管理和处理。

第四阶段是多云部署，将数据、应用和计算资源分散部署在多个不同的云平台上，以实现更好的容灾、备份和安全性。

据不完全统计，目前部分中等规模以上的金融机构已进入第三阶段云化部署。对于云化部署的范围，金融行业内也有多种不同观点，主要如表 1 所示:

表 1 云化部署的范围

序号	名称	观点内容
1	浅上云	仅将非核心或外围系统迁移到私有云，核心系统仍采用物理机部署
2	核心上云	将核心系统都迁移到私有云
3	以云为主	保留部分传统物理机部署，但将大部分业务迁移至私有云平台，并借助云平台提供的技术创新推动业务发展
4	深上云	将所有业务迁移到私有云平台，借助云平台提供的先进技术推动业务深度创新

2.技术平台规模情况

在金融业中，海量数据涵盖了交易、用户行为、市场、客户信息、风险评估、反欺诈检测等多个方面，具有极高的复杂性、多样性和处理速度要求。在金融行业，海量数据通常通过以下指标来定义，如表 2 所示:

表 2 海量数据通定义指标

序号	指标	指标解释
1	数据规模	海量数据具有非常大的数据量，通常以 TB 甚至 PB 为单位
2	数据速度	海量数据的产生和接收速度极快，需要实时处理和分析

3	数据多样性	海量数据包括各种类型的数据，如结构化数据、非结构化数据、实时数据等
4	数据价值	尽管海量数据规模庞大，但其中包含的有价值信息可能非常有限，需要进行深入的数据挖掘和分析。通过对海量数据进行深入挖掘和分析，可以为金融企业提供更好的业务决策支持

为了有效管理和分析这些海量数据，需要采用新的数据处理和分析技术，以提高数据处理和分析效率，为金融业务提供更准确、更快速的数据支持，从而提高金融业务的竞争力和效率。因此，金融业海量数据处理平台一般采用多样化的技术栈构建。

近年来，随着互联网金融及第三方支付业务的快速发展，金融行业各机构的业务量也出现了大幅度的增长，产生的数据量也越来越大，底层平台规模也随之越来越庞大。如何评价一个组织的海量数据规模有很多维度，但基本上以数据总量和机器节点数量两个主要维度来判断一个组织的海量数据规模。

在平台整体节点规模方面，体量较大的金融机构，典型如国有大行（工商银行、中国银行、建设银行等）已超过 8000 台，并计划在 2023 年扩容超过 10000 台；体量较小的也已超过 2000 台。在单集群节点规模方面，Hadoop 集群已有超过 2000 台，MPPDB 集群最大已超过 500 台。在数据总量规模方面，体量较大的海量数据平台已超过单副本 80PB。

这些数据反映了金融行业海量数据处理平台的规模和实力，以及金融行业在数据处理方面的挑战和发展趋势。

3.研发运营一体化应用情况

DataOps 的概念最早在 2014 年由国外学者提出，随后业界逐步对其内涵进行补充。在 2018 年 DataOps 正式被纳入 Gartner 的数据管理技术成熟度曲线当中，由此进入了国际的视野当中。2022 年中国信通院正式牵头成立了 DataOps 能力标准工作组，以此为基础推动我国大数据产业的多元化发展，助力企业完成数智化升级。

不同组织对于 DataOps 的定义如表 3 所示:

表 3 DataOps 的定义

机构	定义
Gartner	DataOps 是一种协作性的数据管理实践，专注于改善整个组织的数据管理者和消费者之间的沟通、整合和数据流的自动化
IBM	IBM 将 DataOps 定义为 DataOps 是人员、流程和技术的有机结合，用于快速向数据公民提供可信的高质量数据

维基百科	DataOps 是一套实践、流程和技术，它将综合的、面向流程的数据观点与敏捷软件工程中的自动化和方法相结合，以提高质量、速度和协作，促进数据分析领域的持续改进文化

研发运营一体化是国内对 DataOps 理念的一种理解，是一种新兴的数据处理方法论和管理能力。这种技术旨在提高数据分析、数据工程和数据科学团队的生产效率和质量。它借鉴了数据研发、数据运营等方面在实践中的优劣点，强调数据工程中的自动化、协作、持续集成和持续交付，以便更快地从数据中获取价值，与 DataOps 的理念具有很多相似处。研发运营一体化的核心理念是将数据处理过程视为数据产品的生产线，通过改进数据流程、工具和团队协作，实现数据产品的快速、可靠和高质量交付。

在金融行业，研发运营一体化的作用具有许多意义，比如通过研发运营一体化，可以提高数据质量。金融行业对数据的准确性和完整性要求很高，因为数据质量直接影响到风险管理、投资决策和客户服务。研发运营一体化通过自动化数据清洗、验证和转换过程，确保数据质量得到有效控制。此外，还可以加速数据分析和决策，因为金融行业需要快速响应市场变化，做出及时的决策，通过自动化数据处理和分析过程，缩短了数据准备时间，使数据科学家和分析师能够更快地获得洞察力和建议。最后，研发运营一体化可以促进跨部门协作，由于金融行业的数据分析涉及多个部门，如风险管理、营销和客户服务。研发运营一体化实

现之后，可鼓励跨部门协作，共享数据和知识，实现业务目标的一致性。

总结来看，研发运营一体化在金融行业的作用和发展主要体现在提高数据质量、加速数据分析和决策、降低数据安全风险、促进跨部门协作和支持创新和实验等方面，金融机构可以更好地利用数据，实现业务价值的最大化。

（二）技术应用情况

1.存储技术应用情况

金融业目前主流的数据存储技术中，HDFS、对象存储是常见的存储选型，它们各自有不同的应用情况:

HDFS 用于存储大规模数据的分布式文件系统。它将数据划分为较小的块并分布在多台机器上，提供高可用性、容错性和可扩展性。HDFS 适用于批处理工作负载，如 MapReduce 任务，但对于随机读写性能较差。

对象存储:　对象存储是一种数据存储模型，将数据存储为对象，每个对象包含数据本身以及元数据（例如文件名、时间戳等）。对象存储不使用传统的文件系统层次结构，而是通过统一的 API 进行访问。这种存储方式不仅适合存储传统结构化数据，也适用存储大规模的半结构化、非结构化数据，如图像、音频、视频等。因此，随着大数据上云成为新的技术趋势，业界普遍将云上对象存储作为新一代的数据湖存储。流行的对象存储解决方案包括 Amazon S3、Azure Blob Storage 等。

据统计，金融行业中，当前以 HDFS 存储为核心的 Hadoop 技术广泛应用于构建企业的统一数据湖。随着大数据上云成为新的技术趋势，业界普遍趋向于将云上文件存储、对象存储等作为新一代的数据湖存储。

2.计算技术应用情况

金融业对计算技术的要求越来越高，而且由于金融的计算处理非常复杂，所使用的计算技术也较为繁杂，目前使用的主要计算技术包括：批计算引擎 Hive/Spark、流处理引擎 Flink、交互式分析引擎 Presto。

Apache Spark 是一个通用的大数据处理框架，支持多种计算模式，其中批处理模式是其最常见的用法之一。在 Spark 批处理中，数据被划分为小块，称为 RDD（Resilient Distributed Datasets，RDD），并且可以通过一系列的转换和操作进行处理。Spark 批处理适用于对离线数据进行分析和处理，例如数据清洗、ETL（Extract-Transform-Load，ETL）、批量计算等。它的优点包括快速数据处理、容错性和丰富的 API。

Apache Flink 是一个开源的流式处理框架，专注于支持实时流数据的处理和计算。与 Spark 的批处理模式不同，Flink 专注于实时数据的流式处理。Flink 支持事件时间处理、窗口操作、状态管理等功能，使得用户可以实时处理和分析数据流。Flink 适用于需要实时响应的应用，如实时监控、实时分析、事件驱动型应用等。

Presto 是一款开源的分布式 SQL 查询引擎，专注于实时分析和大规模数据查询。其独特之处在于高速的查询能力和弹性的分布式架构，适用于需要快速响应和复杂分析的场景。其出色的查询性能、灵活的数据源连接方式以及高度可扩展的架构，使其成为处理各种数据分析挑战的首选工具。Presto 适用于数据探索、BI 报表等领域，能够处理大量数据并支持复杂查询操作，是处理实时数据分析的强大工具。

不同的计算技术适用不同的业务场景。在金融机构中，选择适当的计算模式取决于业务需求、数据特点和性能要求。金融行业普遍使用 Hive/Spark 引擎用于海量数据的批量加工，比如标签加工、指标加工,Flink 引擎广泛应用于金融行业的实时营销、实时风控等场景中，Presto 应用于面向数据分析师的数据探索业务，以及 BI 报表对应。

为了发挥计算技术的最大优势，在构建数据处理平台时还会采用计算与存储一体的数据库技术，在专项领域达到更优的性价比，如: HBase、时序数据库、图数据库、ClickHouse、MPPDB 等。

HBase是一个适合存储海量非结构化数据或半结构化数据的、具备高可靠性、高性能、可灵活扩展伸缩的、支持实时数据读写的分布式数据库。HBase 适合需要高吞吐量、低延迟、大规模数据存储和实时查询的业务场景。其分布式、可伸缩的架构适用于大数据应用，如实时分析、日志处理、社交网络、物联网数据管

理等。同时，HBase 的数据模型类似于分布式哈希表，适合需要快速存取、随机读写的场景。

时序数据库具备高效的时序数据存储和查询能力，适用于需要处理大规模、高频率时间序列数据的场景。支持复杂查询、聚合分析以及数据压缩，能够满足设备监控、智能城市、工业生产等领域的需求。时序数据库的优势在于可扩展性、低延迟读写以及对大量设备生成的实时数据的强大支持，在金融场景中可以用于处理大量的实时数据，如股票价格、汇率、交易信息等。时序数据库还非常适合金融的风险管理，实时交易监控等领域，在金融行业越来越受到重视。

图数据库是用于图谱数据的存储和分析的数据库。客观世界的事物可以抽象成事物实体和事物之间的联系，将实体抽象成点，将实体之间的关系抽象成边，则客观的世界可以抽象成由点和边组成的图谱。用图谱的表示方式很多时候能更客观直接地描述事物的规律，针对图谱数据的分析也能更高效地发现事物直接的规律。图数据库就是专门针对这类抽象为点和边的图谱数据的分析组件。

日志存储和搜索引擎以 Elasticsearch 为例，它是一个基于 Apache Lucene 的分布式搜索和分析引擎，它主要用于搜索和数据分析场景。它允许用户构建实时搜索引擎、日志分析、指标监控和推荐系统等各种应用。日志存储和搜索引擎工具在金融业使用非常广泛，比如在金融的实时搜索和分析场景中因为涉及大量

文本数据，如交易记录、报告、研究论文等，Elasticsearch 的实时搜索和分析功能可以帮助金融企业快速查找相关信息，提高工作效率。另外金融的风险控制、合规、反洗钱（AML）和客户识别，甚至金融数据挖掘和分析等各个场景下，都需要日志存储和搜索引擎这一类的工具，对金融业具有重要意义。

ClickHouse 是面向联机分析（OLAP）处理的列式数据库，适合大规模并行分析、交互式分析应用场景。支持通过分布负载到多个数据库服务器主机，实现存储和处理大规模数据，具有完全的伸缩性、高可用、高性能、资源共享等特征。支持基于 SQL 的查询、统计、分析，且性能好，特别是基于大宽表的聚合分析查询、分析性能非常优异。ClickHouse 完全使用 SQL 作为查询语言，提供了标准协议的 SQL 查询接口，具备基于 SQL 的数据查询、统计、分析等能力。使得现有的第三方分析可视化系统可以轻松与它集成对接。

MPPDB 是一种面向数据分析的分布式并行运算数据库，通过将查询和计算分布在多个计算节点上来实现高性能分析。MPPDB 数据库针对复杂数据查询和大规模数据分析进行了优化，具有并行性、高性能和高吞吐量等优势，适合实时和交互式分析。

各类计算与存储一体的数据库技术都在金融业不同的场景中发挥着重要作用，如：HBase 广泛应用于金融机构的明细高并发查询场景中（如交易流水查询、账单查询），MPP 则应用于构建数据仓库支持大规模数据集的存储和查询场景；部分金融机构

已将时序数据库应用于量化交易、智能化运维等场景中，将图数据库应用于知识图谱、反欺诈等业务场景中，将 ClickHouse 应用于贷款、营销等场景的实时 SQL 统计分析，以及构建标签管理系统。而日志存储和搜索引擎则长期都是金融业处理海量日志以及实时搜索的必要工具。

3.数据湖仓技术应用情况

数据湖仓技术是指将数据湖（Data Lake）和数据仓库（Data Warehouse）两种数据存储和管理模式结合起来的方法，旨在实现更灵活、高效和综合的数据处理和分析。这种方法可以让企业在一个统一的平台上同时享受数据湖和数据仓库的优势。目前主流的数据湖仓技术为湖仓一体技术，该技术在数据湖构建和管理方面有着不同方法和策略，近几年深受金融行业的青睐。

Iceberg 和 Hudi（Hadoop Upserts Deletes and Incrementals）是两种流行的数据湖仓 Lakehouse 组件，常被用来构建湖仓一体技术架构，它们提供了类似事务性操作、数据版本控制、分区管理等功能。通过在数据湖中引入 Hudi 或 Iceberg 等技术来实现增量数据存储、数据更新和删除操作的支持，实现了数据湖和数据仓库的融合。数据湖不仅承载原始数据，还负责承担数据仓库的职责，包括数据的清洗、转换、加工和分析。这种集成的方法不仅简化了数据架构，还提高了数据的可访问性和灵活性，使得金融机构能够更好地应对实时分析和复杂业务需求，从而实现更智能的决策制定和业务创新。

随着湖仓一体技术的兴起，金融机构也逐步将传统的数据湖和数据仓库升级到湖仓一体。目前头部的金融机构已经开始探索湖仓一体技术，并在生产业务中落地部分业务。随着未来时间的推移，湖仓一体将逐步成为金融行业的主流形态。

4.数据架构应用情况

当前金融行业已经从传统的数据仓库体系、大数据体系升级演进到数据中台体系。这一演进是为了更好地整合企业内部的数据资源，提高数据的使用效率，以及满足金融行业日益增长的数据需求。

在这种背景下，金融行业的数据中台体系下的数据架构往往做了数据分层。典型的数据分层如下:

（1） 贴源层: 直接对接各个数据源，如交易系统、核心业务系统、外部数据接口等。通常保持数据的原始状态不做过多处理。

（2） 明细层: 存储经过初步清洗和转换的详细事务数据，为后续的数据分析和报表提供了基础数据。保留了数据的细节，允许对数据进行深入的分析。

（3） 汇总层: 基于明细层的数据，进行进一步的加工和汇总，生成对应的汇总数据、聚合数据或预计算的度量数据，以支撑快速查询和报告。

（4） 应用层：为终端用户或者应用系统提供数据访问服务。这一层通常包括为特定业务或功能定制的数据集，例如数据看板、报表、数据产品或 API 接口。

数据架构的技术落地上，数据贴源层普遍基于数据湖技术构建，如 Hadoop、Delta Lake 或 AWS S3 等。数据湖为后续的数据分析、机器学习和报表提供了一个灵活的、原始的数据基础。明细层和汇总层部分金融机构会基于 MPPDB 构建，对于领先的金融机构而言，则通常使用大数据技术构建。最上层的应用层一般基于高并发的 OLAP 数据库和各类大数据技术组件比如 HBase、ElasticSearh、Redis 等构建。

领先的金融机构已经实现完全基于开放大数据技术栈来构建整个数据架构，从而确保数据处理的高效、灵活和可扩展性。

三、主要挑战

（一）数据存储的挑战

近年来，金融业的数据呈现爆发增长的趋势，金融机构的数据存储面临多种问题和挑战，这些数据存储的痛点包括以下几个方面:

一是数据内容爆炸问题，金融机构每天都产生大量交易、市场和客户数据，导致数据量急剧增长，挑战存储基础设施的容量和扩展性。此外数据多样性也存在较大挑战，数据来自不同业务源、格式和结构，如交易记录、文本、图像等，需要适应多样化的数据存储和处理需求。

二是数据利用率和弹性伸缩问题，随着数据规模增长，传统的大数据处理系统由于存储和计算资源往往是紧密耦合的，这可能导致资源利用率低下，存储的弹性拓展性能也表现不佳，在这种情况下，如果可以把存储和计算分离出来将有效地解决这些问题。

三是性能问题，数据规模一旦增长，数据存储性能非常容易遭受瓶颈问题，目前传统大数据平台架构，任意节点在数据膨胀后极可能会出现 I/O 瓶颈，影响系统性能。

四是数据安全与高可用。金融数据涉及敏感信息，需要严格的安全措施，以保护客户隐私和符合监管要求，增加了存储的复

杂性，此外数据存储需要高度的系统可用性和容错性，存储系统必须具备故障恢复机制，以防止数据丢失或业务中断。

当然，数据存储面临的挑战并不止这些。最近几年，金融业为了应对越来越复杂的存储难题，许多机构尝试采用了存算分离的数据架构，用以应对海量数据的存储痛点并取得了非常好的效果。存算分离技术是一种在大数据处理和存储领域的架构设计理念，它将数据的存储和计算过程分离开来，使得存储系统和计算系统可以独立进行扩展、优化和调度。这种技术能够提高资源利用率，提升系统性能，降低成本，保障数据安全，并提高系统可扩展性。金融业不同的组织通过实践证明，存算分离具有提高资源利用率、系统性能、扩展性，降低数据冗余，保障数据安全，以及简化任务调度等诸多优势，可以帮助金融业更好地应对大数据时代的挑战。

（二）数据计算的挑战

在金融行业，数据计算是关键的业务部分，金融行业始终在追求速度更快、性能更高的数据计算能力，以满足较为严苛的数据分析、时效性等方面的要求。随着金融的业务要求越来越高，数据计算也面临着许多挑战：

一是海量数据计算压力，随着数据规模的膨胀，给数据计算也带来巨大的负担，传统数据仓库或原有架构在遭遇海量数据时会遇到明显的性能瓶颈和高成本问题，因此对于 PB 级的大规模数据存储，需要一套新的架构来应对数据计算的压力，并且这套

架构还需要支持分布式计算和存储，这样用户可以在一个统一的平台上高效地存储、查询、分析和处理各种类型的数据。

二是实时数据计算能力，不同于传统数据计算，金融业对实时数据的计算能力要求较高，而且相比于经典数据仓库和普通的架构，金融实时计算还要求接收和处理各种类型和结构的数据流，并进一步实现实时数据分析和可视化，以满足金融业快速响应业务需求和实现实时决策的需求。目前越来越多的金融机构选择数据湖仓化来应对实时数据计算的挑战，湖仓一体技术不仅能支持实时数据处理，而且具备分布式拓展和提供统一平台管理数据计算引擎等多种优点，大幅度提升数据计算的效率。

三是数据分析和挖掘能力，当前金融行业已不再满足停留在通用的数据计算能力上，还需要实现更高层次的数据分析和挖掘能力上，对于复杂的数据分析和挖掘一直都是金融行业孜孜不倦的追求。然后数据仓库以及传统大数据平台架构并不能很好满足这类需求，存在功能缺失、技术栈复杂、性能瓶颈等各类问题。目前，许多金融业机构采用湖仓化的架构，已提供了丰富的数据分析和挖掘功能，如 SQL 查询、机器学习和图分析等，使金融机构能够从大量数据中挖掘有价值的信息，为业务决策提供支持。

四是弹性计算能力的需求，采用弹性计算能力可以大大提高数据计算的可靠性和拓展性，极大增强数据计算的性能，目前使用数据湖仓、容器化或分布式计算等架构已成为金融行业重点探索和实践的架构。通过这些灵活的弹性架构，可以根据业务需求

动态调整计算资源，实现资源的弹性伸缩，既解决了资源利用和运维成本方面的问题，又提高了资源利用率，并降低了金融业的运维成本。

（三）云化计算的挑战

在金融行业，采用云原生部署的计算引擎模式逐渐成为青睐的对象，云原生计算的模式为金融行业带来了敏捷性、弹性、成本效益、安全性、数据分析能力和丰富的生态系统等多种优势，为金融行业的海量数据处理能力大大新增了丰富且优秀的能力，因此最近几年许多金融机构纷纷拥抱云原生计算并逐步开展了实践和业务落地。当然，尽管云化计算后会带来许多优势，但也并不意味着云化计算这项技术没有风险，事实上云化改造的过程中也存在着一些挑战。

一是迁移成本，对于已有的传统应用，迁移到云原生计算平台可能需要进行重构，以适应微服务架构和容器化部署，这可能会导致较高的迁移成本和风险。这一点目前有经验的大型云厂商都向客户提供自己的迁移工具，配合专业的人力服务，尽可能地降低成本。

二是管理运维方面，云原生计算环境通常比较复杂，涉及多个微服务、容器和网络组件，投入更多的精力进行运维和管理。从现实情况来看，目前许多金融机构都会选择和一些有技术积累的大型厂商合作，借助工具和平台有效地补足管理运维方面的能

力，使得用户自身不需要关心云原生环境，只需要将大部分注意力聚焦在业务和数据上即可。

三是技术复杂性上，云原生计算涉及许多新技术和概念，如容器、微服务、Kubernetes 等。金融行业在采用云原生计算时，需要投入时间和精力学习这些技术，可能会面临一定的学习曲线。目前一些国内主流的大型云厂商都会提供相对应的学习路径和配套设施环境，以帮助金融机构缩短学习时间，尽快实现云化部署。

因此，虽然云化计算已成为很明显的趋势，但从普遍情况来看金融机构自身的云化转型是有一些风险的，往往需要借助一些外部组织的力量，以平稳可靠地实现过渡与改造。

（四）融合计算的挑战

近几年，融合计算在金融行业的应用具有巨大的潜力，其独特而优秀的能力受到了越来越多金融机构的关注，但这项前沿技术也面临一些问题和挑战：

一是异构数据源处理，金融数据通常来自多个异构数据源，包括交易记录、市场数据、客户信息等，这些数据源可能有不同的格式、结构和存储方式，因此在做融合计算的时候需要自适应这些复杂的数据结构，以便进行跨源分析和计算，有时候还会涉及数据转换、标准化等问题。

二是计算自适应，不同数据可能需要不同的计算模型和算法，跨源自适应引擎需要根据数据源的特点和需求，选择合适的计算

引擎（如 Spark、Hive、Presto 或其他组件），有时候针对计算情况还需要选择适合跨源计算的算法，并进行算法优化，以提高计算效率和准确性。

三是计算效率和监控，融合计算可能涉及多个数据源的计算，需要有效管理计算任务和资源，以确保计算的效率和速度，这对融合计算平台提出了很高的要求，此外融合计算因为涉及多个数据源和计算组件，需要有效的监控和调试方法，以确保计算过程的稳定性和可靠性。

四是技术整合，融合计算需要整合多个技术和工具，如数据集成、计算引擎、安全机制等，需要确保这些技术能够协同工作，不仅如此，技术的整合也需要专业的团队，金融机构需要这些人才熟悉融合计算的技术和最佳实践，以便有效地管理和操作。

然而，尽管有着一些困难，但越来越多的金融机构开始采用融合计算的技术和理念帮助组织解决海量数据计算过程中的一些挑战，并取得了很好的效果。

（五）研发运营一体化的挑战

1.数据研发运营一体化（DataOps）金融应用的优势。

数据研发运营一体化（DataOps），通过对数据相关人员、工具和流程的重新组织，打破协作壁垒，构建集开发、治理、运营于一体的自动化数据流水线，不断提高数据产品交付效率与质量，实现高质量数字化发展。采用数据研发运营一体化的优势非常明显，对于金融行业来说至少带来以下三点改善:

一是采用 DataOps 可以提供敏捷数据产品开发流程，通过敏捷迭代，快速响应需求变化，并实现自助服务，各部门可以主动利用数据资产。

二是构建企业内的开发治理一体化流水线，通过数据研发运营一体化流程，把数据质量管控前置到流程中，并在数据治理过程融入开发流程，这样就有针对性地打造了自动化测试流程，及时处理问题。

三是建立精细化的数据运营体系，在 DataOps 中可以实现全数据链路的度量与反馈，基于流程和平台精细化地进行数据的运营，减少人力成本，并降低运营成本，尤其是海量数据处理的时候这种运营体系的优越性更为明显。

2.数据研发运营一体化（DataOps）金融应用的问题。

尽管数据研发运营一体化可以为金融业带来许多的便捷和优势，但在实际调研过程中发现，许多金融机构在实践这项技术的过程中遭遇到诸多问题，导致许多金融从业者并没有真正享受到落地数据研发运营一体化带来的金融科技进步，问题如下:

一是技术和工具的缺失，目前较多金融机构还未完全或仅部分意识到数据研发运营一体化工具的好处，因此并没有真正开展落地的工作。此外当前市场上出现了不少数据研发运营一体化的工具，但是许多工具并不符合金融行业的技术特点，因此要真正提高数据运营效率，需要选择合适的工具，这是众多金融机构需要面临的首要问题。

二是组织变革和调整，实施 DataOps 通常都需要金融机构进行组织架构调整，各个部门之间必须重新组建数据流程，这样才能打破数据孤岛，实现数据的整合和共享。更深层的方面，金融业需要建立数据驱动的文化，推动组织变革，鼓励员工积极参与数据运营过程。

三是数据思维的转型，许多金融机构依然存在研发治理思维的固化，研发角色和治理角色比较割裂，导致实际过程中依然存在分工和不协同的问题，这样就无法有效发挥数据研发运营一体化工具的作用。

四是人才缺口，金融行业对数据运营人才的需求远大于供应，尤其是具备金融业务知识和数据技能的复合型人才。当前许多金融机构在实施 DataOps 过程中，往往优先和相应的 IT 供应商企业合作，采购工具的同时也寻求技术人力支持，度过适应期，然后一并投入资源进行金融机构自身的人才培养和引进。

总的来说，当前金融机构普遍需要解决的是选择合适、全面的数据研发运营一体化工具，以尽快建立流程机制，再逐步推动组织、思维以及人才的变革与进步。

四、关键技术与建设思路

（一）云数一体化

1. 建设思路

随着大数据使用场景的不断丰富，数据使用强度逐步加深，数据的重要性也越来越凸显。其中，数据价值的挖掘离不开两个最基础的能力——存储和计算，也离不开最基本的承载——服务器。

一般业务系统的建设，大多采用“前端—服务端—数据库”的经典构筑模式，且每个业务系统相对独立，系统之间的数据或权限互联互通。业务系统的数据交互，一般采用数据库表同步或跨系统 API 接口调用这两种方式。而海量数据处理平台的构筑模式，第一步是建立一个技术平台；第二步，是将来自业务系统或其他订阅数据源的数据，实时同步或定时同步到海量数据处理平台；第三步，经过符合业务要求的数据治理和开发等形成各种丰富的数据资产；第四步，数据资产采用定期报表、实时报表、数据同步、数据订阅、直接为业务系统服务等各种方式提供最终价值。

（1）云的价值和意义

软件系统的生命周期与硬件的生命周期并不是完全同步，有时软件系统的生命周期可能仅有几个月到几年的时间，有时软件系统的生命周期却可能有 20 年、30 年或更漫长的时间，甚至超

过编写它编程语言的主流生命周期。而硬件设施的生命周期一般是 3 至 5 年，虽然有很多时候也有被使用超过 7 年的硬件，但实际的物理可靠性早已无法满足商业生产的要求。

同样，随着时间和业务需求的变化，软件系统的设计、开发、部署、资源调度的周期越来越快，不再是之前以年计算的漫长流程，而是更加敏捷的 DevOps 开发部署过程。传统的立项、采购、测试、上线、周期结束等漫长过程无法很好地适应快速的需求发展变化等挑战。

为了应对上述挑战，采用云的方式管理团队的基础设施变成了一种好的选择。基于云技术标准化一层虚拟基础设施，既可以随着资源需要和项目过程的变化灵活调整，也可以屏蔽底层硬件生命周期的影响，使得上层的业务系统只需要关注业务本身，而无需再将精力浪费在硬件异常的处理。

（2）早期设计的海量数据处理技术为何不上云

无论哪种海量数据处理技术栈的实现，均具备数据副本冗余能力，即：任意非关键节点损坏，不影响集群的整体工作情况，数据具备高可靠性，数据出了问题可以在极短时间内修复，而且整个过程对上层服务无感知。换句话说，无论是哪种技术栈，均可原生具备物理硬件损坏或更新后的兼容与冗余能力，无需使用云技术再进行一次封装。

其次，基于上文提到的数据价值挖掘过程，海量数据处理的业务均在平台自身内部实现资源调度和故障迁移，计算任务的上

线和下线均可在内部完成管理和冗余，不需要使用云技术再进行一次封装和管理。

再次，早期硬件资源能力不足，采用千兆网络实现节点间互联，数据交换带宽严重不足，需要存算一体设计，并采用计算本地化方式节约数据交换带宽，即：利用数据冗余副本，将计算任务调度到参与计算的数据副本节点，直接从本地获得数据而非网络，以此节约网络带宽。

随着万兆网络的逐步普及，且数据规模的增长和数据价值的显现，存储资源和计算资源不再强匹配，采用一体化设计将会出现资源配置不均衡或浪费等情况。因此，采用不同的服务器，按照需求分别部署计算角色和存储角色，从而实现初步的存算分离，成为一种通用做法。

最后，还有一个不能被忽视的问题，早期的云平台一般能提供给海量数据处理平台的有效关键能力就是服务器，也就是将虚拟出的节点当作一台服务器交给平台，而平台把它当作一台物理机使用。但数据平台本身具有数据冗余能力，这就导致了数据的双重膨胀，即原本一份数据采用三副本策略，在上云之后，云的冗余机制可能又在该副本的基础上再加三副本，形成了九份数据副本的情况，一下子资源被冗余出了九倍，最终形成了严重的资源浪费情况。

（3）海量数据处理平台的新挑战

云数一体化助力构建了数据即服务（Data as a Service）的理念，海量数据处理平台迎来了更好的整体解决方案。首先，随着物理硬件的发展，使得万兆网络或更高条件的网络条件变得普遍且成本不再高昂。随着数据使用的不断深入而导致的存储资源和计算资源不均衡情况，也可以通过存算分离的方式进行解决，而不再有物理硬件方面的阻碍。

其次，随着云和大数据技术的共同发展，存储的多重冗余不再是一个问题。随着海量数据处理引擎的发展，云提供的存储可以被海量数据处理技术直接使用，而不必先同步到数据平台内部才可被后续处理。

其次，随着业务需求的不断发展，海量数据处理技术的不断发展，出现了更加丰富的计算引擎。较早的 MapReduce 引擎被继任者 TEZ 逐步替代，Spark 引擎的加入丰富了计算引擎的生态，Flink 引擎的加入则进一步拓展了海量数据处理的能力范畴。但经典的大数据架构产品设计上很难完成版本间兼容，例如 Spark2.X 与 Spark3.X 的同时支持，此外新增一个全新的处理引擎从产品架构上看也不是容易的。而业务的发展是逐步滚动向前的，新的业务有需求，老的业务期望能保持成熟版本，这就为经典大数据一体化设计产品提出了新的挑战。多云多集群环境或是解决这个问题的有效途径之一。

再次，大数据的资源调度弹性，是在大数据计算集群内部的弹性，但与其他业务资源之间存在壁垒，从整体业务使用特征看，

一般来说业务系统白天繁忙晚上空闲，而海量数据处理系统恰恰相反，一般来说是晚上繁忙白天空闲。随着整个平台的使用，业务系统和海量数据处理平台之间的峰谷也会越来越明显，因此需要一种基于云的全局弹性调度能力。

最终，采用一切皆服务理念完成云数一体化构建，大数据作为云的一种服务实现深度融合，而非早期的集中部署。最终实现云原生大数据能力，实现云数的一体化建设，从而获得以下能力：

一是，全局弹性调度能力。消除平台间的资源壁垒，进一步提高计算资源的利用效率。

二是，大数据引擎的多版本兼容。实现不同历史时期的数据应用可以同时在一个平台实现调度和运行，不会导致因一次升级从而全部数据应用推倒重来的窘困局面。

三是，深度融合降低无效存储冗余。从之前各自独立构建部署，数据在不同平台间必须进行同步，逐步发展为基于数据湖仓化实现数据价值的最大化的同时，降低无效存储冗余度，提高数据一致性。

四是，基于云屏蔽基础设施变更带来的挑战。大数据技术栈虽然具备数据冗余和架构冗余能力，但当物理硬件故障或损坏后，完成修复的过程仍旧是需要大数据近乎全部的技术知识体系才能安全地完成操作。通过云技术，实现对基础设施层故障的屏蔽，大数据的用户可以将最大的精力关注在数据价值挖掘和利用方面等更有价值的地方，而非纠结于大数据平台运维。

（4）实现云和数的一体化

海量数据处理技术的基础设施通常以复杂著称，利用云原生技术，提供敏捷易用低成本的云原生大数据服务，是当前海量数据技术发展趋势。

云原生大数据底座通过构建虚拟集群，容器化中间件、大数据组件及工具平台，如 Spark、Flink、Presto 等，屏蔽底层各种资源的细节，为大数据抽象通用接口，对上实现数据应用的无缝对接，对下实行云化资源的高效调度，大数据产品对接底座即可在各种云化环境提供服务，灵活便捷、安全高效且具备较低成本。典型的云原生大数据架构如图 3 所示：

图 3 典型的云原生大数据底座架构

云原生大数据底座为了进一步提升资源利用率，专为海量数据处理场景设计高性能、高扩展、高安全的统一资源调度器，支

持在线和离线的全场景安全混部，具备管理千万级资源能力，在保证在线服务质量的同时，尽可能提升资源利用率。

基于内存的分布式文件系统缓存，可以同时管理多个底层文件系统，并将不同的文件系统统一在同一个名称空间下，让上层客户端可以透明地访问统一名称空间内的不同路径对应的存储系统内的数据。它统一了数据访问并且连接了计算框架和底层的存储系统，解决了云原生大数据架构网络 IO 以及数据本地化不足的问题，应用程序只需要连接到这个缓存层就可以访问存储在任何底层存储系统中的数据，并能够带来显著的性能提升。

2. 业务应用价值

海量数据处理技术与云平台的互相融合，既拓展了云的相关能力，也更好地将数据处理融入平台之中以帮助用户实现更好地利用数据，其业务应用价值主要体现在以下几方面:

（1）数据服务化。通过云原生及多租户技术，实现跨数据中心、跨集群、跨引擎的统一数据运营及运维；避免由于技术栈升级，业务需求变化等，需要将全部平台推倒重新建设的情况。最终实现不同引擎之间、不同集群之间、不同租户之间，通过云的技术与海量数据处理技术的深入融合，实现数据及其技术组件作为云上的一种服务，为业务提供持续动力，实现数据价值挖掘的持续性、易用性和便利性。

（2）更大且更灵活的规模。早期的数据处理技术栈，当集群规模达到 200 节点时，虽然可以很好地完成部署和使用，但由

于不同租户间的资源竞争、早期技术栈的不完善等因素，需要配置专业的运维人员参与维护；而当达到800节点时，一般至少需要配置3人的专业运维人员参与维护;当达到1200节点以上时，有规模带来的各种问题往往变得不可预估。而采用云原生大数据的方式，可以很好地解决大集群性能优化、弹性伸缩、集群稳定性等问题，实现从小规模到大规模以至于超大规模集群的部署、使用和日常运维。

（3）降本增效。如前文所述，一般的数据处理平台虽然具备对内部计算任务的资源协调与调度，但无法实现全局的资源协调与调度。当云技术和海量数据处理技术深度融合，海量数据计算队列之间、数据与业务系统之间的资源壁垒将被打破，同时可以基于多租户实现资源的隔离、避免互相干扰。最终，通过虚拟集群技术，复用混部资源，实现公平共享的弹性调度，降低成本，提升资源利用效率。

3. 技术示例

云数一体化的技术构建，已经不再是直接将大数据部署在云平台所提供的虚拟机上的方式，而是采用云原生架构的一种深度技术融合。整个技术融合分为以下三大部分:

第一部分，是底层的云基础设施，提供必要的存储和计算能力；存储层，一方面作为数据源，另一方面也是海量数据存储的支撑底座。容器平台，作为整体技术架构的核心资源调度支撑，是各类角色和功能的能力实体。

第二部分，是数据处理技术的云原生架构，提供云原生大数据运行所必需的相关能力，例如数据编排、大数据运行时、混合部署检测、隔离能力与统一任务调度等，包括各类计算引擎能力的云原生引擎支持，最终被数据开发所需的各类开发工具集和可视化工具集进行产品封装。本层提供海量数据处理所需的所有核心能力，包括各类计算引擎和框架，工作流调度等等。

第三部分，是最上层的大数据相关应用，如：BI、AI、自助分析、数据门户等等，实现数据价值的最终可见和高效价值体现。

（二）存算分离化

1. 建设思路

如今，以数据对抗“不确定性”的需求进一步增多，技术也随着需求的变化不断发展，带来了计算能力的提升，也会导致原先的存算一体化资源配置出现比例失调的现象。以海量数据处理领域的离线计算来说，从最初的 Hive 发展到 Spark，而 Spark 从 Spark1.x 到当前的 Spark3.x，相比于最初的框架的能力，整体性能上有数量级的提升。基于数据的业务也逐步丰富，随着技术、业务、数据的高速发展与需求涌现，海量数据处理平台也面临着存储资源和计算资源需求的不均衡，不同历史时期建设标准的不一致，不同业务之间使用的技术栈有所差异的挑战。

为了应对上述三个方面的挑战，海量数据处理的技术架构逐渐从存储计算一体化的构建方式，演进为“存储标准统一、多计算引擎统一调度”的存算分离构建方式。存储和计算不再一对一

强贴合，而是存储支持 HDFS 标准接口和对象存储标准接口，聚焦存储引擎本身的任务，以完成来自业务系统和历史平台等各类不同的数据类型和数据源支持。计算则采用统一的云原生调度引擎，实现多类型、多版本的计算引擎调度，以满足不同建设周期，不同业务需求的使用。同时采用一致的数据开发与治理平台，形成对数据的有效管理，避免出现数据碎片化与数据沼泽。

最终，从业务需求实际出发，以“互相之间的标准兼容”为衔接，以硬件能力为基础，实现存储能力、计算能力、数据开发与治理能力的各自最大兼容与系统性总体统一的存算分离架构，从而满足不同建设周期、不同业务需求对大数据平台的实质要求，避免重复建设、循环建设，使得数据可以持续地被挖掘与发挥其价值。存算分离的技术架构如图 4 所示：

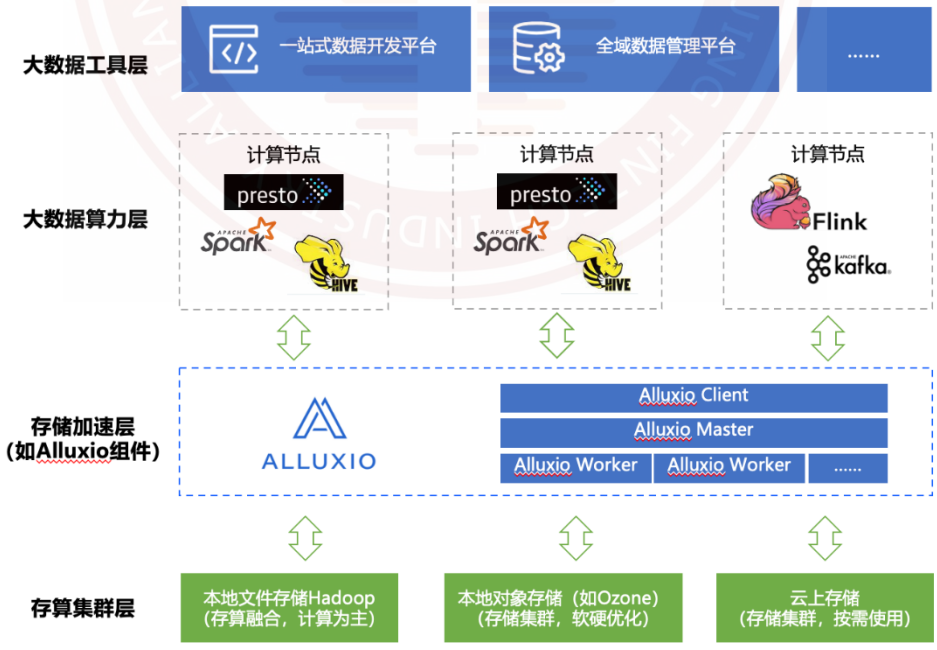


图 4 海量数据处理技术之存算分离架构

基于云数一体的技术能力基础，海量数据处理的技术逐步支持云原生部署，由此解决上一代技术在硬件资源与负载要求的错配、资源浪费、系统难升级、难扩容、增删节点需要成倍的资源处理和数据重分布等挑战，并获得存算分离架构演进所带来的敏捷、池化、弹性、低成本等优势能力。

存储层构筑多集群融合能力，即：同一平台中，既可以部署以 Hadoop 为主的存算一体集群，也可以部署以对象存储为主的存储计算分离集群，亦或是二者同时部署在同一个集群中被使用。

算力层与云原生底座深度融合，通过构建虚拟集群，容器化中间件、计算引擎及工具平台，如 Spark、Flink、Presto、Iceberg 等，屏蔽底层各种资源的细节，为大数据抽象通用接口，对上实现上层应用的无缝对接，对下实现云化资源的高效调度。

存储加速层，是基于内存的分布式文件系统缓存作为以内存为中心的虚拟分布式存储，可以统一数据访问方式，为计算与存储构建了桥梁，可以大幅提升数据计算的性能。

2. 业务应用价值

存储系统和计算系统面对的挑战不一样，存储系统通常使用标准的接口，如：HDFS、对象存储等，其复杂性主要是如何实现数据的有效组织，即：用户、权限、分布/目录、格式支持等；还有计算支持，即：读写性能挑战、大小文件挑战、分布式计算所需的大规模并发度的挑战。而计算系统，其主要面对的是多样化的数据处理，即：采用不同技术特性应对不同业务需求的标准，

同时由于技术的发展，同一类技术也会有较为重大的更新，但受限于业务我们无法直接采用替换升级的方式，所以需要早期版本和最新版本同时支持的情况。采用存储和计算的分离设计，分别满足业务需求和挑战，实现无论是横向不同业务需求之间的平衡，还是纵向新旧技术栈之间的平衡，最终实现数据平台以动态满足业务需求的方式，实现持续的数据价值。因此，存算分离的业务应用价值主要如下:

（1）持续迭代的海量数据处理平台。无论是横向不同业务需求，还是纵向新旧技术栈，均可实现灵活的支持，而不必每一次新技术的诞生就不得不淘汰老的平台，而又因为技术之间的差异和迁移成本等无法实现完全迁移，最终造成多个平台同时运行的窘境。通过存算分离的架构设计，可以完成类似插件化的技术实现，实现在一个大数据平台中的持续迭代，避免一体化设计时的重复建设问题。

（2）统一的存储，灵活地计算，一体化的数据开发和治理工具，快速满足业务需求的同时，避免碎片化。

（3）成本最优解。避免了重复建设、循环建设，以类似可插拔引擎的方式实现能力随需求的扩展，避免了仅技术因素而非业务优化带来的历史工程二次实现或兼容;避免由于技术栈不同而形成的一份数据多重冗余副本等问题。

（4）实现“存储标准统一、多计算引擎统一调度、一体化数据开发与治理”为特征的，具有生命力的、可持续迭代的海量数据处理平台，持续动态地满足业务需求。

3. 技术示例

存算分离架构与经典架构之间，在数据开发和治理平台的参与下，不会对业务层面使用平台的方式产生根本影响。但平台内部的技术架构为了能适应更好的部署形态和业务需求，变化较大。

经典大数据架构包括基础硬件层、运维管控层、存储层、计算层、工具链层；存算分离后，包括混合存储层、缓存加速与统一数据编排层、元数据层、弹性计算层、大数据工具层和统一管控层。

其中，统一管控实现全局资源管理、权限管理、用户管理、安全认证和整体的平台管理和运维监控。混合存储层即支持的所有存储标准能力；缓存加速与统一数据编排层，主要是为了对混合存储层实现兼容和缓存加速，因为大数据计算的分布式性和并发性，对底层存储的压力较大，但又并非所有的存储标准都是为了这样的场景设计实现，故而增加一层缓存和兼容层，以实现不同存储直接的融合；元数据层，实现不同类型数据源和数仓的统一元数据管理，以支持数据的高效、清晰地使用和价值挖掘；弹性计算层，实现不同计算引擎和相同引擎不同版本等灵活支持，以满足业务的各类新增需求与历史兼容性；数据工具层，一方面提供可视化的界面以降低用户对海量数据处理平台的使用难度，

清晰高效地完成数据治理和数据开发工作，另一方面也同时封装了底层架构变动对业务的相关影响，业务层面将尽可能少的变动即可适应新架构的平台。

采用存算分离架构获得优势的同时，也会面临存算之间数据距增大的挑战——计算节点无法从本地获得数据，也就是无法实现所谓“数据本地化”从而影响计算效率，因为这时的数据将全部来自网络传输，计算任务对所需数据的读取时间相较于传统方式将有所增加。通过分布式缓存和数据编排组件，与计算集群资源混合部署，实现对热点数据的缓存与数据本地化，计算需求首先从分布式缓存和数据编排中获取数据，缩小了计算和存储之间的距离，最终实现加速效果。因为实现了数据缓存的构建，其也将大大减少存储节点的访问压力，有效减少对 HDFS 关键主节点 NameNode 或对象存储请求压力等等，优化计算瓶颈。而分层存储的特性，支持内存、SSD、HDD 等不同的存储介质，使得低频冷数据可以以高性价比方式存储的同时，被高频使用的热点数据也以近乎内存的速度支持计算任务的高效使用。

综合来说，以存算分离架构中重要的分布式缓存与数据编排组件 Alluxio 为示例，其优点包括:

一是内存速度 I/O: Alluxio 能够用作分布式共享缓存服务，这样与 Alluxio 通信的计算应用程序可以透明地缓存频繁访问的数据(尤其是从远程位置)，以提供内存级 I/O 吞吐率。此外，

Alluxio 的层次化存储机制能够充分利用内存、固态硬盘或者磁盘，降低具有弹性扩张特性的数据驱动型应用的成本开销。

二是简化云存储和对象存储接入：与传统文件系统相比，云存储系统和对象存储系统使用不同的语义，这些语义对性能的影响也不同于传统文件系统。在云存储和对象存储系统上进行常见的文件系统操作（如列出目录和重命名）通常会导致显著的性能开销。当访问云存储中的数据时，应用程序没有节点级数据本地性或跨应用程序缓存。将 Alluxio 与云存储或对象存储一起部署可以缓解这些问题，因为这样将从 Alluxio 中检索读取数据，而不是从底层云存储或对象存储中检索读取。

三是简化数据管理：Alluxio 提供对多数据源的单点访问。除了连接不同类型的数据源之外，Alluxio 还允许用户同时连接同一存储系统的不同版本，如多个版本的 HDFS，并且无需复杂的系统配置和管理。

（三）数据湖仓化

1. 建设思路

数据湖通常被认为是一个成本较为低廉的存储库，也被理解为一种数据组织形态，承载各种各样来源的结构化或非结构化的原始数据或原始数据的镜像，它可以存储任何类型的数据，包括像图片、文档这样的非结构化数据。这样的数据存储结构，存储其中的数据不需要满足特定的 schema，数据湖也不会将特定的 schema 施加其上。相反的是，数据的拥有者通常会在读取数据的

时候解析 schema，当处理相应的数据时，将转换施加其上。相比传统的分治式数据组织方式，数据湖强调的是统一的数据组织方式，也就是所有的数据都在湖中进行加工、处理、分析和流动。

数据仓库，则是将原始数据源或数据湖中完成数据初步加工的结果进行进一步的处理。数据仓库主要存储的是基于关系型数据库组织起来的结构化数据。数据通过转换、整合以及清理，并导入到目标表中。在数据仓库中，数据存储的结构与其定义的 schema 是强匹配的。

由于业务场景和需求的复杂性，数据湖和数据仓库虽各有优势，但又有各自的局限性，为了满足业务实际需要，我们常常将二者联合使用，即：数据湖、数据仓库和其他专门的系统，采用数据持续集成的办法将架构拼接混合使用，而这将带来三个常见问题：

一是缺乏开放性。数据仓库将数据转换为专有格式，这增加了将数据或工作负载迁移到其他系统的成本。由于数据仓库主要提供 SQL 的访问模式，因此很难运行其他分析引擎，如机器学习系统。此外，使用 SQL 直接访问数据仓库中的数据非常昂贵和缓慢，这使得与其他技术的集成变得非常困难。

二是对机器学习或其他新生的技术栈的支持有限。尽管有很多关于 ML 和数据管理融合的研究，但没有一个领先的机器学习系统，如 TensorFlow、PyTorch 和 XGBoost，能够很好地在仓库之上工作。与提取少量数据的 BI 系统不同，ML 系统使用复杂的

非 SQL 代码处理大型数据集。对于这些用例，有些数据仓库供应商建议将数据导出到文件中，这进一步增加了系统的复杂性。

三是数据湖和数据仓库之间的强制权衡。超过 90%的数据存储在数据湖中，这是因为数据湖使用廉价存储，从开放直接访问文件到低成本的灵活性。为了克服数据湖缺乏性能和质量的问题，企业将数据湖中的一小部分数据 ETL 到下游数据仓库，用于最重要的决策支持和 BI 应用。这种双系统架构需要对数据湖和数据仓库之间的 ETL 数据进行持续的工程处理。每个 ETL 步骤都有可能导致失败或引入 bug，从而降低数据质量。同时保持数据湖和数据仓库的数据一致性是非常困难和昂贵的。除了需要为持续的 ETL 支付费用外，用户还要为复制到仓库的数据支付两倍的存储成本。

而数据湖仓化 Lakehouse 技术可以有效解决以上三个问题，具体如下:

一是 Lakehouse 技术使得海量数据处理技术所构成的平台同时具备数据仓库和数据湖的优势，数据分析师和数据科学家可以在同一个数据存储中对数据进行操作，同时也能为数据治理带来更多的便利。

二是 Lakehouse 技术通过将数仓构建在数据湖上，使得存储变得更为廉价和弹性，并能够有效地提升数据质量，减少数据冗余。正因为将数据仓库构建在数据湖之上，数据湖采用的 Parquet、ORC 等开放兼容的存储格式将作为最基础的底层数据

存储格式，而非绑定某种特定的引擎。这样不同的存储引擎和计算引擎，不同的编程语言或数据操作语言都可以在湖仓上进行操作和数据处理。同时，数据湖所支持的存算分离架构、低成本硬件和集群化部署等优势，均会在数据湖仓上得到体现。

三是 Lakehouse 技术通过将数仓构建在数据湖上，使得原本仅可以支持结构化数据的数据仓库，在数据湖仓化后，其结构可以支持更多不同类型的数据，包括文件、视频、音频和系统日志等等。数据湖仓可以根据应用的需求为绝大多数的数据施加 schema，使其尽可能标准化，避免数据沼泽的产生。湖仓中所保存的数据经过了清理和整合，可以被直接使用以加速分析过程。同时相比于数仓，它能够保存更多的数据，数据的时效性也会更高，能显著提升报表的质量。

2. 业务应用价值

数据湖仓化不但具备数仓超高分析性能，同时也融入了数据湖的灵活性，存储和管理任意规模、类型的数据。湖仓中的数据可自由流动，通过统一元数据管理可以实现对湖仓资产进行统一管理，同时湖仓方案也具备湖仓数据联邦分析、统一开发平台以及统一管控等湖仓融合特性。

通过构建数据湖仓 Lakehouse，不同角色的用户可以基于统一的数据平台简化整体的数据存储、计算、管理的流程和需求。数据湖仓也可以拆除湖与仓库之间的壁垒，无论是全局视角，还是利用存算分离技术与数据网格的架构方式，实现在满足不同使

用者需求的同时，实现统一标准支持和必要的资源复用，从而实现数据利用的最大价值。

数据湖仓化主要的业务应用价值如下:

一是更少的数据管理复杂度。使用数据湖仓 Lakehouse 的架构方式，任何已经被纳入平台的数据源都可以被直接访问或合并数据等方式以供数据价值使用，可以是直接从数据湖中的源数据，也可以是被治理后的数仓化数据; 而不是从原始数据中逐步提取数据，并再经过数据处理过程后，流转到数据仓库进行最后的数据开发工作。

二是降低数据治理难度并优化数据治理过程。使用数据湖仓 Lakehouse 的架构方式，可以通过全面整合资源和数据源管理来简化并改进数据治理过程，并基于各类数据标准，以标准化的开放模式构建，可以更好地控制安全性、指标、基于角色的访问和其他关键管理元素等。

三是提高成本的利用效益。数据湖仓 Lakehouse 的构建方式，其基础设施采用云数一体、存算分离等特性，当存储和计算资源需求不匹配的时候，可以轻松添加存储资源或计算资源，并通过数据缓存加速层优化存储资源与计算资源极度不均衡时带来的性能挑战。同时，实现不同计算需求之间的互相隔离，若有需要被启用，则无须担心不同用户之间由于计算资源竞争带来的衍生问题。最终，实现低成本数据存储，经济高效地扩展。

3. 技术示例

数据湖仓 Lakehouse 的整体架构，并没有脱离海量数据处理技术的整体框架，而是实现了有机整合，并发挥最大限度的效能。数据仍是从各类数据源被接入，如：传统的数据库，消息队列或实时数据流，亦或者各类 IoT 设备、日志数据、爬虫数据、订阅的文件等等；但可以同时发挥数据湖和数据仓库的优势，即：将适合的数据类型存储到适合的存储方式和格式中，并形成初步的元数据构建和初步的数据治理，以类似数据仓库的数据组织方式管理数据湖中的多类型数据。

基于表格式引擎和缓存加速引擎，并通过针对元数据信息和各类任务的综合性管控，实现对数据的有效组织。如：入湖时，通过元数据自动发现、一键入湖、增量迁移等实现入湖管理，生命周期内，通过库表元数据管理、小文件合并、数据分区管理等实现数据的湖内管理，各类必要的数据流转和数据治理过程，通过任务管理、作业脚本管理、监控日志管理等实现对各类任务的有效管控和自运行。通过表格式引擎进行有效的数据组织，采用缓存加速引擎优化不同存储技术的性能与接口差异，最终实现对数据的有效组织。

经过初步组织后的数据，通过各类计算引擎，结合业务所需的任务流程和数据关联与流转，最终发挥数据的价值。

湖仓架构同时具备海量存储系统和传统 DBMS 的优点。一方面，湖仓架构提供低成本的弹性存储，支持存储各类数据格式；另一方面，湖仓架构也提供 ACID（Atomicity 原子性，

Consistency 一致性，Isolation 隔离性，Durability 持久性）事务性支持、数据版本、审计、元数据管理、缓存、查询优化等。

举例来说，数据湖仓组件 Iceberg 在设计之初的目标就是提供一个开放通用表格式(TableFormat)来实现湖仓一体的方案。因此，它不和特定的数据存储、计算引擎绑定。目前大数据领域的常见数据存储(HDFS、S3...)，计算引擎(Flink、Spark...)都可以接入 Iceberg。在生产环境中，可以根据实际情况选择不同的组件使用。甚至可以不通过计算引擎，直接读取存在文件系统上的数据。

基于 Iceberg 可以实现海量数据处理的流批一体。上游组件将数据写入完成后，下游组件及时可读、可查询，可以满足实时场景。并且 Iceberg 同时提供了流/批读接口、流/批写接口，可以在同一个流程里，同时处理流数据和批数据，大大简化了 ETL 链路。

基于 Iceberg 可以实现通过 SQL 的方式进行表级别模式演进，即：数据表演化(TableEvolution)，进行这些操作的时候，代价极低，不存在读出数据重新写入或者迁移数据这种费时费力的操作。比如在常用的 Hive 中，如果我们需要把一个按天分区的表，改成按小时分区。此时，不能在原表之上直接修改，只能新建一个按小时分区的表，然后再把数据 Insert 到新的小时分区表。而且，即使我们通过 Rename 的命令把新表的名字改为原

表，使用原表的上层应用，也可能由于分区字段修改，导致需要修改 SQL，这样花费的经历是非常繁琐的。

此外，数据湖仓组件 Iceberg 还有如下主要优点:

一是分区演化(PartitionEvolution)。Iceberg 可以在一个已存在的表上直接修改，因为 Iceberg 的查询流程并不和分区信息直接关联。当我们改变一个表的分区策略时，对应修改分区之前的数据不会改变，依然会采用老的分区策略，新的数据会采用新的分区策略，也就是说同一个表会有两种分区策略，旧数据采用旧分区策略，新数据采用新新分区策略，在元数据里两个分区策略相互独立，不重合。得益于 Iceberg 的隐藏分区(Hidden Partition)，技术人员在写 SQL 查询的时候，不需要在 SQL 中特别指定分区过滤条件，Iceberg 会自动分区，过滤掉不需要的数据。Iceberg 分区演化操作同样是一个元数据操作，不会重写数据文件。

二是列顺序演化(Sort Order Evolution)。Iceberg 可以在一个已经存在的表上修改排序策略，修改了排序策略之后，旧数据依旧采用老排序策略不变。往 Iceberg 里写数据的计算引擎总是会选择最新的排序策略，但是当排序的代价极其高昂的时候，就不进行排序了。

三是隐藏分区(Hidden Partition)。Iceberg 的分区信息并不需要人工维护，它可以被隐藏起来。不同其他类似 Hive 的分区策略，Iceberg 的分区字段/策略(通过某一个字段计算出来)

不是表的字段和表数据存储目录也没有关系。在建表或者修改分区策略之后，新的数据会自动计算所属的分区。在查询的时候同样不用关系表的分区是什么字段/策略，只需要关注业务逻辑，Iceberg 会自动过滤不需要的分区数据。正是由于 Iceberg 的分区信息和表数据存储目录是独立的，使得 Iceberg 的表分区可以被修改，而且不会涉及数据迁移。

四是镜像数据查询(Time Travel)。Iceberg 提供了查询表历史某一时间点数据镜像(snapshot)的能力，通过该特性技术人员可以将最新的 SQL 逻辑，应用到历史数据上。

五是支持事务(ACID)。Iceberg 通过提供事务(ACID)的机制，使其具备了 upsert 的能力并且使得边写边读成为可能，从而数据可以更快地被下游组件消费，通过事务保证了下游组件只能消费已 commit 的数据，而不会读到部分甚至未提交的数据。

六是基于乐观锁的并发支持。Iceberg 基于乐观锁提供了多个程序并发写入的能力并且保证数据线性一致。

七是文件级数据剪裁。Iceberg 的元数据里面提供了每个数据文件的一些统计信息，比如最大值、最小值、Count 计数等等。因此，查询 SQL 的过滤条件除了常规的分区、列过滤，甚至可以下推到文件级别，大大加快了查询效率。

（四）计算融合化

1.建设思路

计算融合化的目的，是整合不同的海量数据计算组件，形成统一的计算、查询与分析入口，旨在解决传统大数据架构下诸如计算引擎的语言门槛高、处理引擎多而杂、海量数据计算链路长而复杂、资源利用率低、存储异构、数据孤岛等痛点和问题。其建设思路具体如下:

（1）语法自适应

支持对接不同类型的外部计算(执行)引擎，包括 Presto、Livy、Hive、Flink,以及丰富多样的数据源,如MySQL、PostgreSQL、Hive、SparkSQL/Livy、Oracle、Phoenix (HBase)、ElasticSearch、Kylin、ClickHouse、Druid、Presto 等。引擎之间、数据源之间所使用的 SQL 语法存在一定的差异,作为计算分析的入口需要能够有效屏蔽语法差异做到语法自适应,从而为整合不同的海量数据处理组件提供基石。提供一套通用 SQL 语法，并通过 SQL 兼容转换功能来实现不同 SQL 语法之间的转换;做到在用户无需更改 SQL 语法的前提下实现底层执行引擎的切换,通过一套 SQL 语法,自动适配不同计算引擎和数据源语法。 顾名思义，SQL 兼容转换功能整体可以划分为两个模块，即 SQL 兼容与 SQL 转换。

SQL 兼容: 在进行 SQL 兼容时，为解决部分大数据平台语法与业务强耦合、定制化严重，以及不同语法强行融合易导致歧义的问题，遵循干净、可扩展、可替换、多场景兼容的兼容准则，

提供插件式的解析模块，将 SQL 语法模板化，分类管理，形成可扩展、多样的 SQL 生态。将 SQL 语法分为两大类即通用型（如 SQL 标准语法，以及常见的 Spark、Hive、Flink 等大数据查询语法）、独特型（自定义语法，不具有普适性），基于分类语法模板、语义扩展定义、配置文件生成多样的 SQL 解析器，并且支持动态切换解析器，灵活性强。任意解析器得到的语法树均将转换为统一的逻辑计划，可基于此逻辑计划生成符合不同引擎或数据源方言语法的执行语句（这一过程即 SQL 转换）。默认使用通用 Parser，其基于 SQL 标准语法，支持大部分通用大数据语法（如 Spark、Hive 语法），适用于大部分的大数据系统组件。而对于一些与业务逻辑强耦合的自定义语法，支持自定义 SQL 模板，生成自定义解析器，通过这种做法，结合上文提及的生成统一执行计划以及下文提及的 SQL 转换，可以灵活地将业务任务切换到指定引擎。

SQL 转换：SQL 转换发生在两个阶段，一阶段是通过解析器得到抽象语法树后，进行语法树重写以确保该语法树能转换为统一逻辑计划；另一阶段是基于统一逻辑计划与不同引擎或者数据源语法之间的等价映射关系，能够将逻辑计划转换为不同的引擎或数据源语法，做到执行引擎的无感切换，也为下文的智能引擎选择功能奠定基石。

这种执行引擎的无感切换，不光能让平滑进行智能引擎选择，充分发挥引擎的优势特点，增加 SQL 执行效率；还能支持业务无

感迁移，做到在用户无需更改 SQL 语法的前提下实现底层执行引擎的切换，并且尽量最小程度地更改用户的使用习惯。

通过 SQL 兼容和 SQL 转换，能够统一计算入口，整合大数据平台组件，降低大数据系统使用的门槛和繁琐程度。

（2）引擎选择自适应

智能引擎选择是计算融合化的进阶功能之一，表现融合的自适应性。通过组合算法，自动为每条用户 SQL，挑选合适的不同类型的计算引擎（如 Presto、Spark 等）来执行，以提升用户体验（如响应时间快、可靠性高等）和资源利用率（CPU、内存等）。传统基于 RBO/CBO 的 SQL 优化框架，存在规则人工定制、统计信息缺失、历史流水闲置、失效资源浪费等几个主要问题。针对这些问题，可以设计一套基于历史负载的查询优化（History-Based Optimization，HBO）和基于机器学习的引擎选择。HBO 目标是分析处理历史用户 SQL 流水，以通用、抽象化的 HBO 策略，增强补充（非取代）已有的具体化 RBO/CBO 策略。机器学习算法可以自动学习 SQL 特征，更好地弥补人为规则的黑角。把 HBO 和机器学习结合起来，可以更好地降低日均提效失败（即错误选择引擎后执行失败）的 SQL 数，提升用户 SQL 的平均执行时间，减少引擎集群无效负载的同时节省宝贵的计算资源。

HBO 框架的设计实现包括四个子模块；它们也代表了一条用户 SQL HBO 优化的四个串行阶段。基于引擎选择（SQL 优化）的实时性要求，整个 HBO 耗时必须控制在毫秒级。

查询签名：执行的所有计算类 SQL 语句（DQL/DML），无论执行结束后状态是成功还是失败，流水入库时都新增生成查询签名（Query Signature，QS）字段。

查询签名是自研设计的 SQL 文本的 “浓缩” 表示，包含 SQL 访问的库表名和关键子句（Filter/Join/GroupBy/Orderby）中包含的列名。通过 QS 来匹配判断当前用户 SQL 与哪些历史 SQL “HBO 等价”，然后通过分析汇总这些历史等价 SQL 的执行特征，来决定当前 SQL 是否应选某类引擎执行。

索引宽表：HBO 要求为每个最新提交的用户 SQL，从历史流水库中查找其最近一段时间内等价的历史 SQL 集。依赖外部的统一元数据服务，固化缓存 HBO 索引宽表来解决检索的实时性能问题。宽表的每一条记录对应一条历史查询，包括查询签名、执行时间、引擎类型、结果状态、数据量、引擎 shuffle 数据等信息。

历史检索：基于查询签名的完全匹配（exact match），调用统一元数据服务的 REST API，返回最近历史区间（如一周）内的索引宽表记录集。通过不同的 API 入参，指定返回记录集的最大行数、起止日期、超时时间等属性，确保检索的实时性能（平均 < 100ms）。

提效判定：分析统计获取的历史记录集，综合执行时间、失败率、引擎分布等数据，对比系统阈值参数，决定是否对当前 SQL 选择使用的某类计算引擎来执行。

（3）计算运行时自适应

传统的大数据架构下，整个计算链路通常是单向的，上层计算缺少底层状态（比如资源状态）的反馈。单向链路虽然简单，但会造成计算资源不均衡、资源利用不充分等问题。算力感知是计算融合化的又一个进阶功能，是自适应计算架构里底层反馈的桥梁，让上层计算具备感知资源状态的能力，进而自适应地调整资源使用。通过算力感知，可以获取计算资源整体的资源状态以及单节点详细的算力指标，上层计算借此自适应地动态调整计算决策、资源使用、任务调度等。 以 Presto 为例，作为一款典型的 MPP 架构、纯内存计算的交互式查询引擎，为了追求性能的最大化，Presto 会尽可能地利用节点上可用的资源，包括 CPU/内存/网络带宽等，节点间的物理资源规格也需要尽可能保持一致。然而在实际的使用场景中，节点的 CPU/内存等负载（算力）是随时波动的，而 Presto 的原生任务调度策略并未将节点的算力考虑在内，导致在节点算力明显下降的情况下，计算任务会受到严重的影响，从而产生长尾问题。为此，Presto 做了针对性的优化，在动态的计算环境中，通过感知节点算力的变化，自适应地调整计算任务的调度，避免低算力节点的影响。 Presto 自适应任务调度主要分为: Task 自适应调度与 Split 自适应调度，方案实现的核心思想是:根据节点的算力情况动态分配 Split 和 Task。

Presto Coordiantor 在运行过程中，会实时感知 Worker 节点的算力变化情况，同时计算出对应的节点可用算力权重，在 Task 和 Split 的调度过程中，针对不同的算力权重，根据模型

计算出相应的 Worker 上还可分配的 Task 或 Split 数目，对于算力严重下降的节点，少分配或不分配 Task 或 Split，尽量避免长尾问题，从而做到自适应的调度。 自适应调度效果：当计算 Task 在 CPU 波动比较大的节点上，会造成明显的计算长尾的问题，拖慢整个任务的运行，如图 5 所示，在没有开始自适应调度的情况下，Task 的执行时间波动很大。

ID	Host	State	⏸	▶	🔖	✔	Rows	Rows/s	Bytes	Bytes/s	Elapsed
2.0.0		FINISHED	0	0	0	1	3.38M	11.5K	2.25G	7.82M	4.92m
2.0.1		FINISHED	0	0	0	1	4.16M	24.3K	2.78G	16.6M	2.85m
2.0.2		FINISHED	0	0	0	1	4.22M	8.24K	2.81G	5.62M	8.55m
2.0.3		FINISHED	0	0	0	1	4.22M	8.28K	2.81G	5.65M	8.49m
2.0.4		ABORTED	0	1	0	0	2.53M	4.22K	0	0	10.00m

图 5: 无自适应调度下 Task 执行时间波动较大

在开启自适应调度后，Task 会避免调度到 CPU 算力差的节点，有效地消除长尾问题。如图 6 所示，Task 的执行时间更加均衡，避免长尾问题影响整个计算任务的性能。

ID	Host	State	⏸	▶	🔖	✔	Rows	Rows/s	Bytes	Bytes/s	Elapsed
2.0.0		FINISHED	0	0	0	1	4.22M	32.3K	2.81G	22.0M	2.18m
2.0.2		FINISHED	0	0	0	1	4.22M	32.5K	2.81G	22.2M	2.16m
2.0.3		FINISHED	0	0	0	1	4.16M	32.4K	2.78G	22.1M	2.14m
2.0.4		FINISHED	0	0	0	1	4.22M	34.4K	2.81G	23.5M	2.04m
2.0.1		FINISHED	0	0	0	1	3.38M	33.0K	2.25G	22.5M	1.71m

图 6: 有自适应调度下 Task 执行时间比较均衡

（4）资源自适应

面向大规模集群部署，多集群是运维管理的常规手段。但从资源管理的角度，多集群会带来诸多问题:

一是资源对业务不透明，业务在使用计算资源时，需要人为指定特定集群。人为选择集群的方式不仅麻烦，也会带来集群负载不均衡的问题;

二是由于资源不能统筹管理，资源整体利用率不高。资源自适应的目标是能够把所有资源统一管理起来，对计算提供统一的资源池，对资源统一调度，打破集群间的隔离问题，实现对资源的公平共享，充分利用空闲资源，提高资源利用率，同时对业务透明化。

资源自适应主要包括集群间弹性伸缩和集群内资源调度。每个租户对应一个虚拟 K8S 集群，每个租户都有最低的资源保障，租户之间能借用资源，也可以借用集群空闲资源。通过自适应调配资源，打破集群间的隔离，充分利用不同业务的潮汐效应，错峰使用资源，提升整体的资源利用率。

（5）数据编排自适应

在公有云、私有云、内网不同场景中，大数据底层存储是异构的，主要涉及 COS、HDFS、Ceph、Ozone 等。面向异构化的存储，自适应计算平台构建了一层统一的数据编排层，位于计算和存储之间，透明化存储差异。通过适配不同的权限和认证体系的统一的存储 Client，解耦计算和存储，避免不同计算引擎和不同存储间的相互适配工作，让计算和存储更加专注。

在大数据场景中，每天产生海量的数据，而数据治理往往赶不上数据积累的速度，海量元数据以及小文件会给存储 Master 节点（例如 HDFS NameNode）极大压力，造成性能抖动。数据编排层会自适应缓存存储元数据，以及自动小文件合并，减轻 Master 节点压力，同时在跨 DC 数据访问时，加速元数据访问，提升数据访问速度。

2.业务应用价值

以自适应作为串联不同系统的能力抓手，通过自动、智能的方式解决传统大数据架构中的痛点问题，以实现海量数据处理过程中的计算融合化，其业务应用价值如下:

（1）语法自适应：统一不同的计算入口，自动适配不同的 SQL 语法和标准，降低大数据系统使用门槛。

（2）引擎选择自适应：根据 SQL 特点和历史执行信息，实现 SQL 引擎的智能选择与加速，自适应调优计算参数，提升整体计算性能，降低失败率。

（3）计算运行时自适应：根据运行时状态和信息反馈，动态调整计算执行拓扑，解决大数据计算执行链路复杂，稳定性低的问题。

（4）资源自适应：统一资源管理，屏蔽各类资源的性能差异，使业务能透明地使用资源；通过自适应地弹性扩缩，资源借调，最大化资源使用效率。

（5）数据编排自适应：实现不同异构存储场景下的存储加载策略，自适应不同架构下的数据融合计算需求，通过自动数据冷热分层，多级缓存，提升存储访问性能。

（6）场景架构自适应：适配多云混合架构，实现最优的跨集群、跨 DC、跨云计算路由，打通数据链路，解决数据孤岛。

3.技术示例

计算融合化提供了完整的端到端的大数据解决方案，适配公有云、私有云、内网不同的场景。整个架构可以分为四层：核心引擎层、计算层、资源层、数据编排层。

核心引擎层是统一计算入口和智能决策中心。对外提供一套通用 SQL 语法，并自动适配计算引擎的不同 SQL 标准。同时汇总来自元数据、历史流水、底层集群状态等不同信息，通过组合算法做出 SQL 自适应优化、物化视图自主构建、引擎智能选择、计算参数调优等重要决策，从而影响整个计算的生命周期。

计算层会根据不同场景，采用不同的计算引擎，其中 Spark 负责 ETL、报表场景，Presto 负责交互式查询场景，Hermes 负责日志检索、用户画像场景，Doris 负责数据湖查询分析，PowerFL 负责安全数据计算。核心引擎层根据 SQL 特点和使用场景选择最佳的计算引擎。为了保证计算在不同架构下的计算稳定性，Remote Shuffle Service(RSS)提供统一的数据 Shuffle 服务，实现计算执行拓扑自适应。

资源层整合云上和云下资源，把能够把所有资源统一管理起来，对计算提供统一的资源池。通过资源自适应调整、租户间资源弹性调度、集群中资源借调等手段，统筹管理调度资源，提升资源整体利用率。

数据编排层适配不同异构存储，透明化存储差异，解耦计算和存储。自主学习数据访问模式，自适应缓存热点数据和元数据，加速数据访问性能，提升集群稳定性。

（五）研发运营一体化

1.建设思路

在数据驱动的大背景下，组织内的数据意识已经逐渐成熟，数据相关的需求激增。但是技术引擎的动力略显不足，数据项目链路长、协同差、数据准备的时间长、数据需求的质量低等问题放慢了企业转型的步伐。研发运营一体化（以下称为 DataOps）以破局者的身份出现在大家的视野当中，为企业的数据引擎换挡。

DataOps 在组织、流程和工具三个方面对企业产生影响。要求组织内人员更深入地吸收数据文化、加强协作，重构数据工作流程，加强一体化设计的数据开发、治理、运营运维、应用的平台优化，具体要求如表 4 所示:

表 4 DataOps 在组织、流程和工具的具体要求

序号	名称	要求
1	组织	数据思维纳入组织通识教育，专注于效能与协同的岗位，决策层的战略支持。

2	流程	标准化数据工作程序，从面向交付到面向业务，降低对个人的依赖。
3	工具	使用者不仅限于技术人员，强化一体化设计能力，并可进行持续优化。

DataOps 标准的建设目的在于:一是确定 Dataops 概念意义、明确 DataOps 实施流程、把握企业发展阶段和方向；二是通过标准引领的方式，引导企业快速接纳 DataOps 文化，提供 DataOps 的建设方法；三是以评促建，通过自评或三方评估的方式，对 DataOps 能力查缺补漏。

以下重点从工具能力建设上总结出 DataOps 的 3 个层次+4 个核心能力，助力企业加快数据洞察的步伐，具体分析如图 7 所示:

图 7 DataOps 的层次和核心能力

DataOps 的 3 个层次如表 5 所示:

表 5 DataOps 的基础层、开发层、治理层说明

层次	名称	说明
基础层	多环境（集群）管理	在基础层支持多环境多集群管理，支持一套统一的平台来对接多套不同规模、不同类型的集群，支持大数据平台、MPP 数据库等各类数据库作为计算引擎，提供统一的开发与应用体验，具备跨云部署以及对跨云 EMR 的兼容能力，面向多云场景提供统一开发、统一管控能力，用户可在不同的集群环境中（同类型引擎）实现代码及相关资源的无缝发布。
开发层	数据全链路流水线开发	按照数据开发的基本过程，分为：模型设计、数据开发、部署上线、质量稽核 4 个步骤，日常用户的主要操作均是在这 4 个步骤之中。
治理层	统一元数据管理、质量管理	治理层主要包括统一元数据及质量管理两块能力，细分下还包括全域血缘打通、资产分析、质量管理等。

4 个核心能力如表 6 所示:

表 6 DataOps 的 4 个核心能力说明

能力项	说明
统一调度编排	需支持分布式调度引擎，支持百万级别复杂依赖调度。调度平台为底层通用能力，离线、实时、质量稽核等各任务均使用统一的调度能力。
统一监控/告警	支持统一的告警通道,不同的产品模块内可能都会使用告警能力，例如离线任务突破基线、实时任务失败、API 调用失败、质量稽核未通过等。针对某个告警通道仅需开发一次，即可在各个产品内使用此告警方式，例如短信、邮件，企业微信、电话告警等。
安全保障	主要包括系统安全、数据安全、安全审计等。
团队协作	责任人机制、锁机制、用户组。

2.业务应用价值

随着时间的推移，数据的数量、频率、多样性都在增加，在一个万物皆可被度量的时代,数据积累的速度超过大部分企业跟上其脚步的速度。这也意味着能够帮助企业完成自动化日常任务，提高数据质量，促进不同团队之间的协作，带来更准确的洞察和分析，以及助力企业进入敏捷、自动化和加速的数据供应链环境的 DataOps，未来将会在企业的数智化蜕变中，发挥不可小觑的作用。

DataOps 提供支持全链路协作的数据规划、集成、建模、开发、治理、分析、服务等工具能力，通过严谨的 CI/CD 流程规范和自动化的测试发布运维加持能力，缩短从原始数据加工到业务应用数据的路径，并在数据治理能力的加持下，输出准确、及时、有效的数据，提升效率的同时保障数据质量，为上层各类数据应用赋能。

主要核心业务应用价值如下：

（1）协同，围绕数据价值链基于协作空间使数据团队不同的角色更好地协作，打破团队间孤岛，缩短从原始数据到数据价值的路径。

一站式：支持开发、分析、运维和运营多种场景；全链路操作，无需多平台多工具间来回切换。

多角色协作：基于 DataOps 倡导的团队协作理念，在保障数据安全和资源隔离的情况下，不同的数据团队角色围绕项目进行协作。

（2）效率，基于 DataOps 敏捷迭代、自动化流程和工具提升数据可靠性，加快数据生产和分析链路效率。

敏捷易用：通过增量式代码开发和发布、代码自动补全、可视化图拉拽方式进行流程设计，快速易用支持业务开发。

开发灵活：开发模式适应多场景，支持先开发后编排以及先编排后开发。

高性能可扩展：高性能调度引擎，支持日千万级任务调度，可对接多种引擎并支持引擎扩展，默认支持大多数 JDBC 接口的引擎。

（3）一体，服务企业数据管理、数据生产、数据应用、数据运营 多个角色，给予不同视角一体化的产品体验。

全链路生产治理：通过事前规划、事中异常阻断、事后质量和成本分析以及数据流通安全管控为数据的生产和消费提供有力的质量和安全保障。

一站式运营治理：基于数据自服务和民主化理念，在安全稳定的基础上，通过数据地图、数据洞察和共享，让数据的查找、理解、分析和共享更容易。

（4）质量，贯穿事前中后的数据质量控制，融入 DataOps 管道式开发流程，全面保障数据质量提升。

数据任务/工作流提交版本前要求通过在线调试，在线调试会自动拉起数据表对应的质量监控任务。

敏捷数仓建模工具在数据建模时支持直接引用事前定义好的数据标准，在源头上做到落标。

遵从数据标准的表在进行数据集成任务时，支持对脏数据设置零容忍阈值来做到贯标。

3.技术示例

数据研发运营一体化（DataOps）：是数据开发的新范式，将敏捷、精益等理念融入数据开发过程，通过对数据相关人员、

工具和流程的重新组织，打破协作壁垒，构建集开发、治理、运营于一体的自动化数据流水线，不断提高数据产品交付效率与质量，实现高质量数字化发展。

其技术能力实现框架示例如下:

（1）研发管理

在数据研发环节，通过串联数据模型设计、标准设计、质量设计，数据集成、数据存储、数据加工等流程，建成数据研发治理一体化能力，提升数据产品交付质量。

一是强化需求评价，明确数据需求内容，降低沟通成本。

二是通过“先设计，后开发”的方式，在建模环节做好数据标准、质量的设计。

三是构建、离线、实时、数据挖掘的一体化开发能力，并在开发任务链中嵌入数据质量稽核能力，及时解决质量问题。

四是为业务人员提供便捷的数据自服务空间，支持数据需求自主探查，缓解需求响应和交付压力。

（2）交付管理

在数据产品交付环节，通过强化对数据版本和代码版本的管理，构建 CI/CT/CD 自动化流水线，提升测试与部署效率，降低人为风险，提高数据交付效率与质量。

一是建设自动化测试流水线，加强对单元测试、集成测试的管理，对代码质量、数据质量均进行测试，提前发现问题、处理问题。

二是加强版本控制与环境管理。对代码版本与数据版本都进行管理，保证各阶段数据的实时可用性和可验证性。

三是建设自动化部署发布流水线，加快数据部署效率，降低人为操作风险。

（3）数据运维

在数据运维环节，通过构建完整监控体系，对研发资源运转情况、变更情况进行跟踪，提高全链路可观测性，及时发现问题、定位问题、处理问题，保障数据开发流水线平滑高效运转。

一是构建完整的监控体系，对开发流水线运行情况、质量情况等进行时刻监控预警。

二是对数据资源、计算资源、存储资源等进行调度优化，合理分配相关资源，优化运维成本。

三是打造标准化、敏捷化变更流程，应对开发流水线的各类变更场景。

四是构建异常管理知识库，构建自动化运维能力，提升运维效率。

五是基于数据流水线运行情况，持续对流水线任务编排情况、平台配置情况进行调优，不断提升开发流水线性能。

（4）价值运营

在价值运营环节，关注各项数据成本的开支，打造开放的反馈机制，通过量化指标驱动数据的精细化运营，提升数据研发质效。

一是细化数据产品交付和维护成本核算，精细控制相关资源投入，识别并减少浪费。

二是打造反馈机制，及时收集数据研发各环节堵点问题，深挖问题源头并持续改进。

三是构建完善的量化指标体系，对数据开发流水线交付效率、需求响应速度等进行定量评估，不断优化工作流程和资源分配策略。

（5）系统工具

在系统工具建设方面，按照 DataOps 理念打造平台能力，构建完整工具链，为敏捷化、一体化、自动化的数据研发流水线提供强大技术支撑。

一是支持代码线上流转，遵从“先设计，后开发”的建设原则。

二是构建 CI/CT/CD 能力，支撑自动化的测试流水线与部署流水线功能，能够对代码和数据进行版本控制。

三是支持对数据研发全链路的监测与报警功能，通过大屏展示等形式实时展现研发效能、质量等信息。

四是建立全链路数据安全监测与管控能力，在数据研发全生命周期中落实权限的管控、敏感数据脱敏加密、高危操作审计等功能。

（6）组织管理

在组织管理方面，关注架构设置，岗位角色职能划分与发展规划，借助敏捷方法持续优化人员、工具的协同水平。

一是合理配置企业内部的数据技术架构、数据人员架构。

二是设置相应的岗位角色，明确晋升路线与考核方式。

三是依托敏捷方法，着重关注团队、工具间的协同问题，并持续进行优化。

（7）安全管理

在安全管控方面，构建完善细致的安全风险管理体系，并在DataOps 各个环节中全面嵌入安全屏障。

一是加强对数据研发全生命周期中的风险识别，风险预测。提前制定风险预案，将风险的影响持续降低。

二是结合外部法律法规、监管要求与企业内部安全需求，健全风险管理策略并不断更新完善。

三是主动对数据研发过程的各环节进行安全测试，提前发现问题、处理问题。

五、发展趋势和展望

金融业一直都是大数据、人工智能等各种前沿技术理想的试验田，金融业积累的海量数据可以供从业人员不断测试、优化、完善各类新兴技术，并同时挖掘、探索金融业新的使用场景，进一步赋能金融业的创新和发展。未来海量数据处理技术对金融业的发展具有重要意义。

一是提高决策能力：海量数据处理技术可以帮助金融机构更好地理解和利用数据，从而提高决策的准确性和效率。通过分析大量的历史数据和实时数据，金融机构可以更好地预测市场趋势、评估风险和发现投资机会，这将帮助投资者做出更明智的决策，提高投资回报率。二是优化风险管理能力：海量数据处理可以帮助金融机构更好地识别和管理风险。通过分析大量的数据，金融机构可以更准确地评估风险，并采取相应的措施来降低潜在的风险。这将提高金融机构的稳定性和可持续性。三是增强个性化服务能力：海量数据处理技术可以帮助金融机构更好地了解客户需求和行为模式。通过分析客户数据，金融机构可以提供更个性化的金融产品和服务，满足客户的特定需求，提高客户满意度和忠诚度。这将帮助金融机构在竞争激烈的市场中获得竞争优势。四是提高交易效率：人工智能和海量数据的算法等技术可以帮助金融机构更快速地处理和分析大量的交易数据。通过提高交易速度和决策效率，金融机构可以降低交易成本，提高交易效率和利润。这对于高频交易和算法交易尤为重要。

随着技术的不断发展，海量数据处理技术在金融业中将发挥更重要的作用，并带来更多创新和机会。本章节将介绍几类目前发展较为热门的技术，包括生成式人工智能、实时数据湖仓、数据网格、数据编制等，并简单探讨这些技术的基本概念和优势。

（一）生成式人工智能驱动数据技术方面

在当今数字时代，数据成为推动技术创新和商业决策的重要驱动力。然而，许多行业面临着数据获取不足、质量不高以及数据样本不平衡等挑战。这些问题严重影响了人工智能模型的性能和应用。随着人工智能技术的迅速发展，生成式人工智能驱动数据技术正在崭露头角，为数据处理和分析领域带来了一系列创新解决方案。

生成式人工智能是一类人工智能模型，其主要目标是生成新的数据样本，这些样本与训练数据类似。生成模型通过学习训练数据的潜在分布和特征，能够生成逼真的图像、音频、文本等。其中，Transformer 模型和 GANs（生成对抗网络）是生成式人工智能的代表性模型。

1. 数据增强与样本生成

数据增强是利用生成模型合成新的数据样本，从而增加训练数据的多样性，改善模型的泛化能力。在自然语言处理领域，通过生成式语言模型，可以轻松地生成大量与原始数据类似的句子，从而扩充数据集。在计算机视觉中，生成式模型能够生成具有各种变换和扰动的图像样本，有效提高了模型的鲁棒性和准确性。

此外，生成式人工智能也可以用于样本生成。在医疗影像诊断中，数据收集是一项耗时且昂贵的任务。通过使用生成式模型，可以合成大量的医疗影像数据，帮助加速医学图像处理和疾病诊断的研究进展。

2. 数据清洗和修复

数据质量是影响人工智能模型性能的一个重要因素。然而，现实中的数据往往会受到噪声、缺失或错误等问题的影响。生成式人工智能为数据清洗和修复提供了新的解决思路。

生成式模型可以学习数据样本的分布规律，进而自动纠正损坏或缺失的数据。例如，在自然语言处理任务中，模型可以通过生成合理的上下文信息来修复错误或缺失的单词或短语。这种技术在提高文本数据质量和减轻数据预处理负担方面具有巨大潜力。

3. 数据合成与扩充

在某些应用场景下，真实数据难以获得，或者可能涉及隐私或版权问题。生成式人工智能提供了一种安全且有效的方式来生成合成数据，以替代真实数据进行模型训练或测试。

在自动驾驶技术开发中，生成式模型可以生成大量虚拟驾驶场景，用于测试自动驾驶系统的稳定性和安全性，而不必依赖于真实道路测试，从而降低了测试成本和风险。

4. 自动数据标注

数据标注是训练监督学习模型所必需的，但通常需要大量的人工劳动。生成式人工智能技术为自动数据标注提供了新的解决方案。

通过结合生成式模型和半监督学习方法，我们可以自动生成标注数据，从而减轻了人工标注的负担。例如，在图像分类任务中，生成式模型可以自动给图像生成标签，从而用于训练分类器。

5. 生成数据模型与查询语句

通过接入最全面的企业内部数据源及数据资产元数据目录，内部数据模型由提示词驱动 AIGC（Artificial Intelligence Generated Content，AIGC），查询结果由 LLM（Large Language Model，LLM）进行解读分析，构建由业务需求及即席查询需求驱动的抽取建模。整个过程包括业务模型分析、自动生成数据模型、规划查询步骤、生成查询并执行、查询结果判读、多次迭代或并行执行进行分析探索、

（二）实时数据湖仓方面

最早实时湖仓中实时强调的是流式 ETL 能力，湖仓则指的是基于 Lakehouse 的湖仓存储。随着 OLAP 组件近期纷纷加入湖仓的查询支持，例如 StarRocks 实现了自适应从湖仓加热和降冷到湖仓，实时湖仓中的仓支持也泛指了 OLAP 对 Lakehouse 分析，整体实时湖仓架构如图 8 所示:

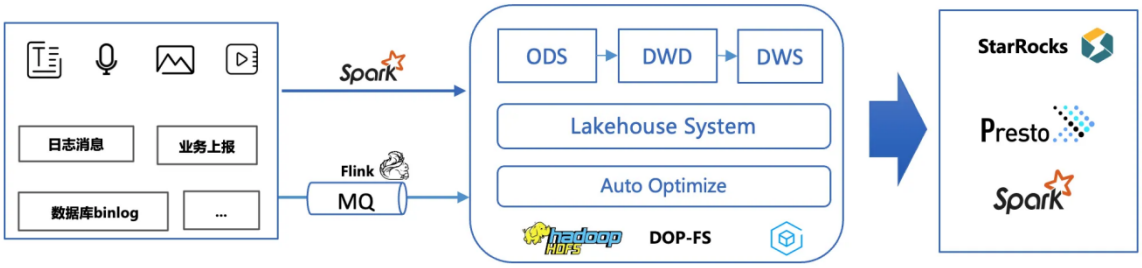


图 8　实时湖仓架构图

1. 基于 Flink 的流式 ETL 能力

数据 ETL 是大数据分析的典型数据处理链路，业务系统的原始数据经过 Extract 到达数仓贴源层(Operation Data Store，ODS)，经过 Transform 转换成数据明细层（Data Warehouse Detail，DWD），经过数据建模或者分析的结果 Load 到数仓服务层(Data Warehouse Summary，DWS)供后续应用层例如 BI 使用。基于 Flink 构建大数据实时链路是业界应用最为广泛的方案之一，2023 年 SIGMOD 的最佳系统是 Flink 已经证明。

2. 基于 Iceberg 的 Lakehouse 架构

早在 2020 年，当 Lakehouse 概念提出时，有关企业就调研了业界相关开源产品来构建公司的湖仓一体架构。无论从软件架构设计、代码质量还是 license 多个维度来看，Iceberg 在当时都是最适合应用的一个项目。

（1）软件架构上：它的核心能力与存储引擎、计算引擎解耦并抽象出一套核心能力的 API，使用 Iceberg 的厂商可以根据业务需求自行扩展 API 来构造自己的存储和计算支持。例如业界

有多个基于 Iceberg FileIO API 构造的存储支持，其中贡献到开源社区的有 AWS S3FileIO、Dell EcsFileIO、Google GCS FileIO、Aliyun OSSFileIO，类似的还有 Catalog、TableScan 等。

（2）代码质量方面：基本上它的每一个 PR 都是经过严格 Review，整体 UT 的覆盖率也比较高。

（3）合规方面：Iceberg 贡献给 Apache 基金会， 商业使用上是较为安全的。

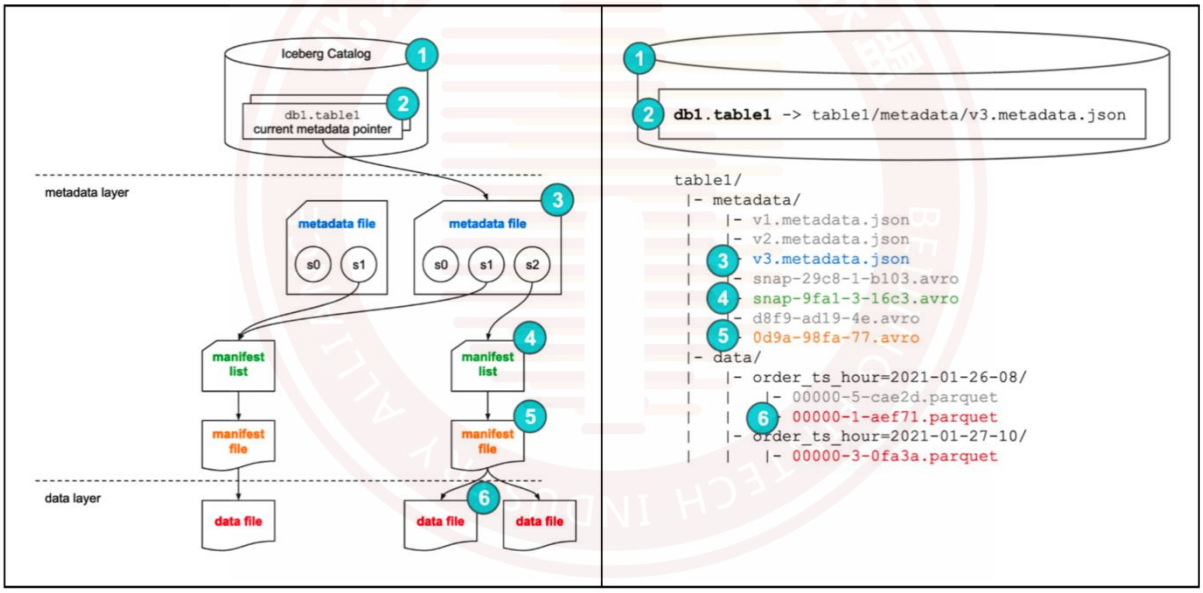


图 9　Iceberg 架构元数据组织方式

Iceberg 最初是为了解决上云后数据规模变大后 Hive 无法继续服务的问题而立项，Hive 分区信息存储在 Metastore 背后的数据库里。当元数据本身成为大数据时，相关的数据库就会成为瓶颈，特别在大表读取场景下分区信息扫描会给数据库带来压力，分区目录下文件扫描会给存储元信息服务带来压力(List 操

作对 HDFS 或者对象存储都有巨大开销）。而 Iceberg 架构的元数据组织方式（如图 9 所示）可以有效地解决这些问题。

Iceberg 通过快照机制支持 MVCC（Multi-Version Concurrency Control，MVCC），它在快照中保存了库表的基础元信息，同时引入 Manifest 文件(avro 格式）记录和组织了数据文件。通过从 snapshot 到 Manifest 再到 datafile 的组织的方式规避了传统数仓元数据相关问题。这种元数据的组织方式以及快照功能非常适合 Flink 流式入湖和流式增量读取从而实现流式 ETL(Extract,Transform,Load)。此外，流式入湖使用较短的 commit 间隔生产快照使得数据在较低延迟内可见。通过快照的时间戳或者快照 ID 获取增量数据使得 Flink 流式增量读取变得非常灵活，同时它还提供了类似消息系统的多种消费策略使得可以进行灵活的增量读取，所以 Iceberg 成为构建实时湖仓的最为流行的组件之一。

3. 智能湖仓服务(Auto Optimize)

流式入湖通过快速 commit 可以降低数据延迟，但也导致了元数据过多的问题。Iceberg 的元数据组织方式将元数据过多的问题转成元数据文件过多，从而将问题转化到擅长处理这种问题的大数据存储系统，例如对象存储系统。但是大量的小文件，一方面会降低后续的数据分析的性能，另一方面会给用户带来较大的存储和维护成本。未来需要构建智能湖仓服务(Auto Optimize，下面简称 AO)，通过异步的方式对湖仓进行多种优化，包括小文

件合并、数据生命周期管理、元数据清理，智能索引创建，数据排序和重分布等。 AO 整体的架构如图 10 所示:

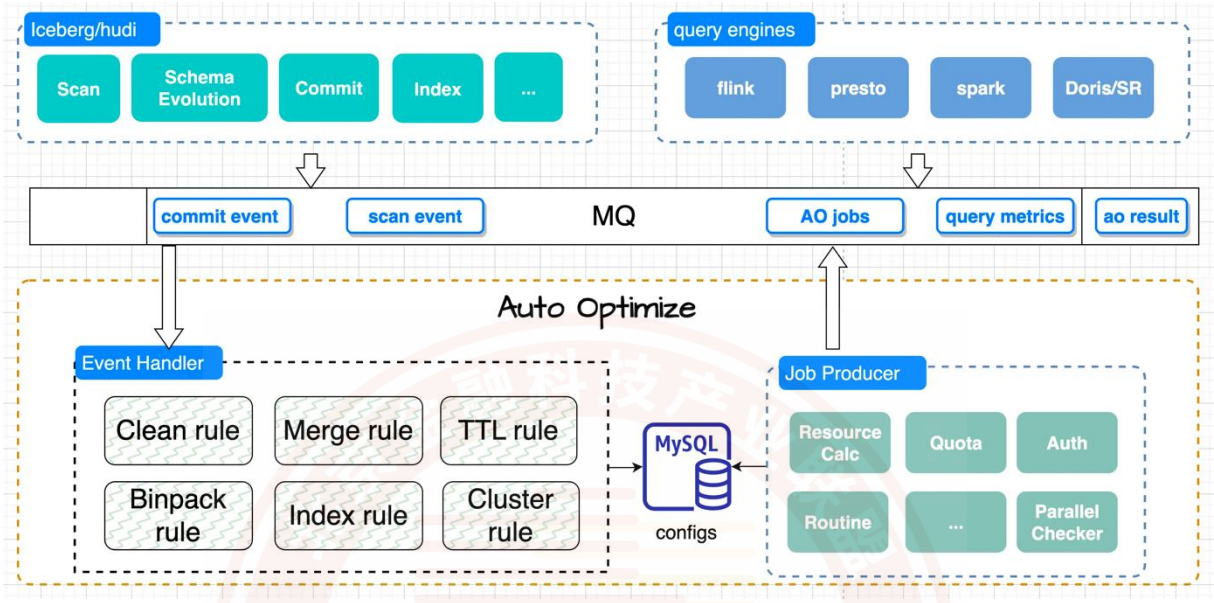


图 10 智能湖仓服务架构图

AO 是基于规则和反馈的事件驱动框架。表的 scan, commit, schema change, properties change 等事件汇报到 MQ, AO 消费这些事件并让事件经过预先定义好的一些优化规则生成一些优化任务放到 MQ,再由相关模块消费任务相关 topic 并根据资源、之前类似任务执行效果调整参数后下发任务到调度系统。 总的来说，实时湖仓搭配 AO 不仅可以持久稳定地工作，还能实现湖仓存储数据的持续优化。

4. 更多的湖仓场景展望

实时湖仓除了覆盖数仓的经典场景，一些新的特性也使得一些过去无法在数仓上实现的场景成为可能，例如，全链路 CDC（Change Data Capture，CDC），多流拼接， 特征工程等场景。

（1）基于 Branch 的全链路 CDC

基于数据库 binlog 的 CDC 方式进行流式数据库增量同步方式已逐步成为一种主流的数据库同步方式，相比过去的数据库同步方式它具有几个优势：1) 能够捕获所有数据的变化，捕获完整的变更记录；2) 每次 DML （Data Manipulation Language，DML）操作均有记录无需发起全表扫描进行过滤，拥有更高的效率和性能，具有低延迟，不增加数据库负载的优势；3) 无需入侵业务，业务解耦，无需更改业务模型。数据库通过 CDC 方式同步后，用户还会基于库表构建下游的数据处理链路，一般是通过增量消费实时湖仓表的方式进行。实时湖仓支持的增量消费分为两种类型：一种是只读取增量更新后的数据；另外一种是需要读取更新前后的数据，即 change log feed 模式。

根据不同的消费模式有两种流式写入方式，一种是通过 Merge on Read 模式 upsert 到 Iceberg 库表，但是 Merge On Read 模式在生成回溯 Change Log Feed 时性能较为一般。Iceberg 从 1.2.0 版本后开始支持 Branch & Tags 功能，用户可以像使用 git 一样管理数据的版本和分支。未来我们会使用 Branch & Tags 的功能实现了一套全新的 CDC 写入和读取逻辑，使之可以快速地生成回溯记录(Change Log Feed)。基于 Branch 全链路 CDC 的设计方案如图 11 所示：

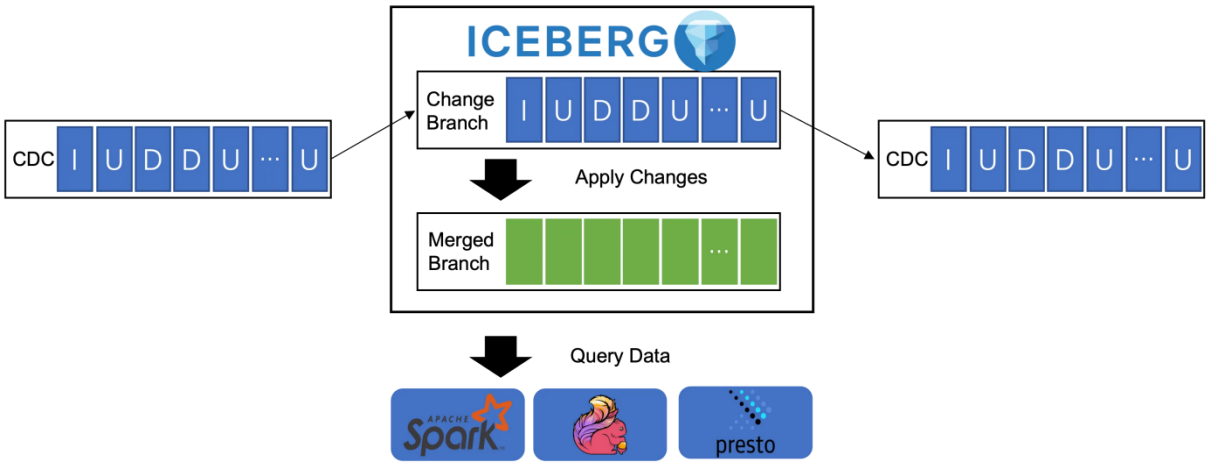


图 11 Branch 全链路 CDC 设计方案图

（2）多流拼接

多流拼接是广告大数据一种常见场景，它是指多个业务指标流拼接成一个大宽表。在过去的生产实践中常见的做法是通过KV引擎如 HBase 实现， 或者通过 Flink 本身的多流 Join 功能实现。使用 KV 引擎实现多流拼接的主要问题是成本比较大且无法 scale 到较大数量级， 而使用 Flink 进行多流拼接则因为 Flink 无法容纳较大的状态无法支持不同流时间相差较多的场景，如广告场景下曝光、点击、转化流的时间差可达几天以上。流程如图 12、图 13 所示:

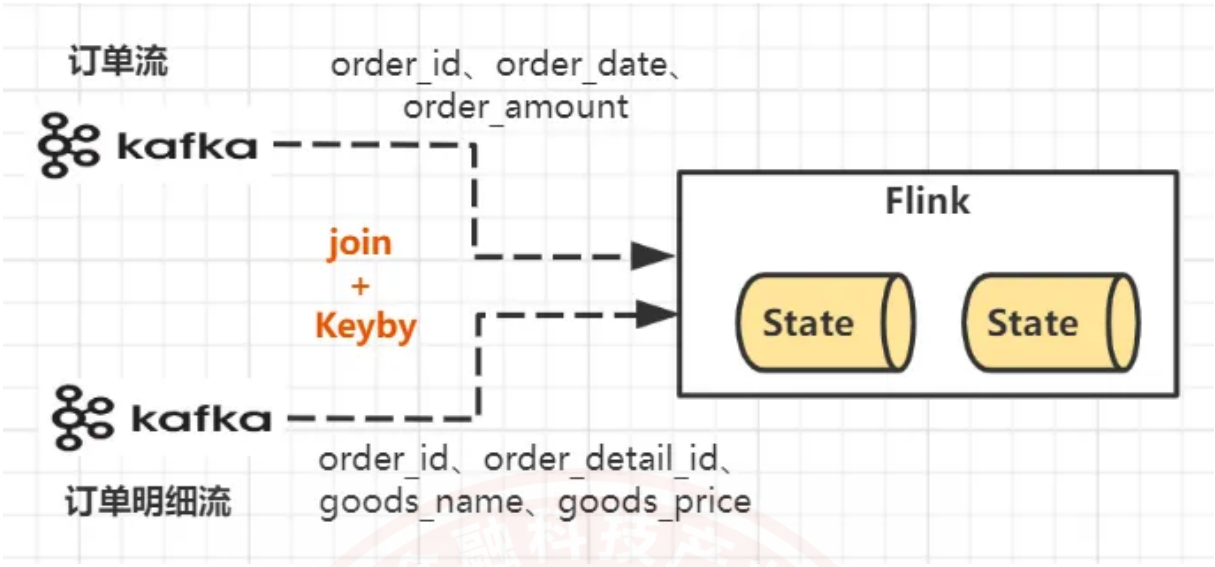


图 12 多流拼接流程图（1）

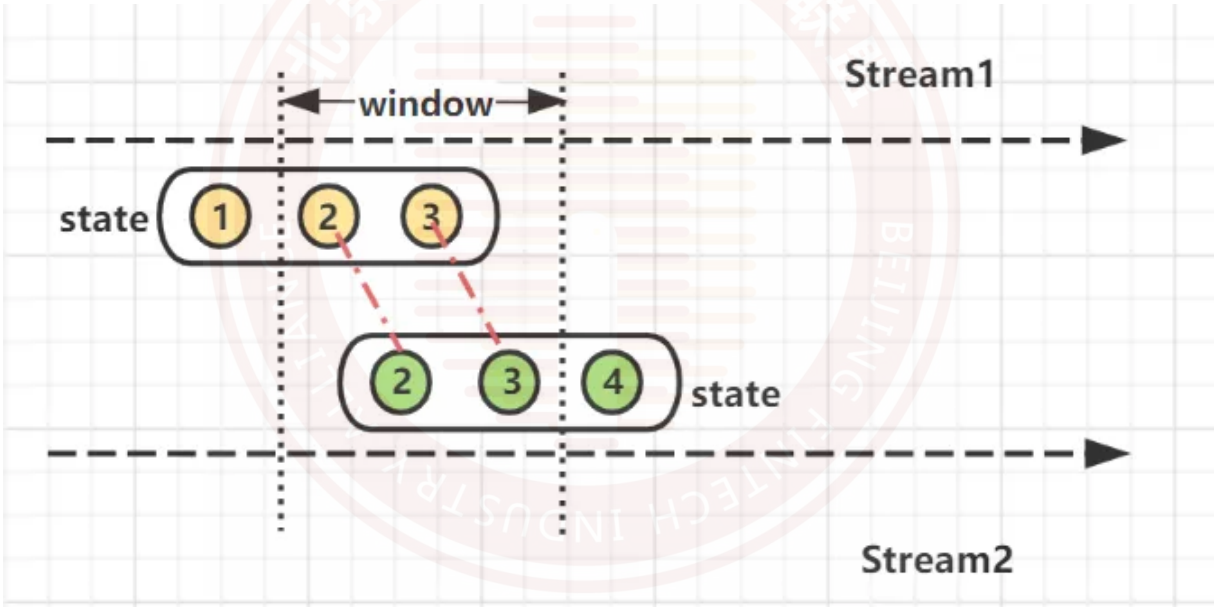


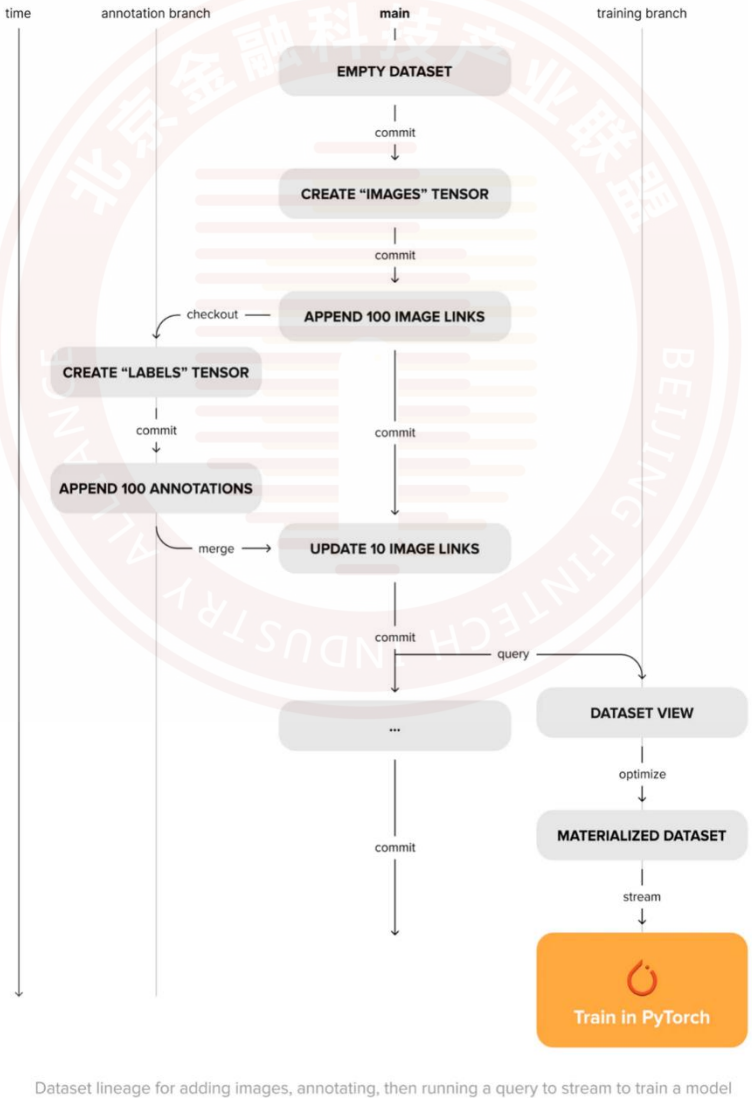
图 13 多流拼接流程图（2）

实时湖仓提供了一种全新的思路支持多流拼接的能力，通过将多个流分别写入到大宽表的不同列，并通过异步合并的方式达到拼接成大宽表的目的。写入逻辑与拼接逻辑的解耦使得拼接的获得较大的灵活性，用户可以定制拼接逻辑以及使用不同的引擎

来进行拼接，拼接执行时也可以根据延迟的需求来选择不同的算力来执行。

（3）Deep Lake

在机器学习领域，Deep Lake 场景对 git-like 使用方式的需求，非常适合使用 Branch & Tags feature 进行支持。流程如图 14 所示:



Dataset lineage for adding images, annotating, then running a query to stream to train a model

图 14 Deep Lake 流程图

（三）数据网格方面

海量数据处理在传统观念和经典的最佳实践中，一般会将大量的原始数据集中起来，而后经由专业的数据分析团队完成数据维护和开发，最终由专业的数据团队结合业务部门需求完成数据价值。这样的数据组织形式很好地解决了数据碎片、零散分布、数据标准不一致等挑战，并由专门的数据团队负责维护的专业数据平台很好地保障了数据价值的发现和落地，协助业务部门更好的实现业务需求，业务部门可以完全专注于业务价值。

随着技术的发展，海量数据处理相关的技术栈有了极大的易用性提升，再加上数据的价值不断被证明，业务部门的数据分析需求也日趋旺盛。一方面，数据需求被释放，再加上变得相对简单的交互方式，让业务部门具备了直接完成对简单需求的分析和处理能力；另一方面，数据所有者对业务和数据是十分熟悉的，而数据分析师或数据开发工程师对技术方面有着深入的研究，若数据服务于业务，就需要业务部门了解技术的能力边界，同时数据部门深入了解业务细节，海量数据处理技术的发展和多年的人才积累打破了技术和业务的壁垒。

数据网格便是为了更好地适应上述变化形成的一种数据架构概念，以“数据自治”为核心，不同的业务部门自行认领并管理自己的数据和服务，在统一的技术平台和资源池中，形成自己的数据产品，并自治管理。与其他部门的数据共享主要是利用API进行数据订阅或推送的形式，从整体来看，类似工业流水线，每

一个节点构筑自己所需的数据价值挖掘并持续创新，对自己的下游提供符合标准的数据产品即可。

数据网格将之前完全耦合的数据平台，升级为以标准进行不同数据产品衔接的统一数据平台。类似“流水线”的设计将有助于数据价值创新，只要确保数据产品之间的“标准”没有变化，数据产品节点内部的创新不会影响其他数据产品，最终避免因一个小变化却牵一发而动全身。

上述这些是基于数据架构方面的，而海量数据处理所用技术栈和所用资源方面仍是统一和集约的，故可以在全局角度，在不同数据产品间实现一种联合治理形式，以支持不同部门数据产品彼此的灵活性、操作性和协同体验，在促进数据创新的同时避免数据碎片化。

除此之外，数据网格的有效利用也会带来下述优势。

一是，更加广泛的参与度和更容易的价值挖掘。数据网格的数据架构方式，使得数据能力的构建不只限于数据科学家、数据工程师、数据分析师等技术团队，而是不同的业务部门甚至于管理部门也可以平等且容易地参与数据能力构建过程。通过面向价值的数据架构设计，使数据价值可以更容易被发现，减少数据孤岛和运营瓶颈，让业务部门实现更快决策，让技术部门可以专注在复杂事件的处理和技术栈持续迭代优化等任务。

二是，更低的成本与更高的效率。采用分布式的数据架构，将粗旷的数据处理阶段精细化；精细化的方式，使得我们不再需

要等待全局或大流程中的某一个阶段所有的数据处理完成后才进行下一个大处理阶段，只要上游数据产品可用，下游数据产品的数据处理过程就可以启动，无需由于与自身数据分析无关的数据而长时间等待，最终获得更好的数据处理效率。而由于精细化后，数据需求按照各自不同的特点自然散落在不同的时间段处理，避免了同类资源的同时间段的竞争抢占，避免出现某一时间段数据集成资源紧张但计算资源闲置，而后又随着整体过程计算资源繁忙而数据集成资源闲置的情况，可以在整体上更加平衡地使用整体集群资源，以获得更好的成本。

三是，更高的技术栈灵活性。集中式的数据架构由于数据关联的复杂性，数据关联间的复杂性会导致对底层技术栈的直接锁定，技术栈是持续向前发展的，当面对有巨大价值的新技术时，被锁定的技术栈往往无法实现快速且安全的升级，此时，我们面对的不再是多个小问题，而是一个大的挑战。数据网格，采用分布式的数据架构方式，只要确保数据的上下游标准的可靠性，就可以将整个平台的大挑战，切分为每一个业务部门甚至于每一个数据产品的小问题；最终，新的需求、新的数据产品采用新的技术栈，而稳定的数据产品保持其自身技术栈，避免了由于技术因素导致数据架构的被迫全面升级，也不必担心由于技术升级带来潜在新的风险挑战。

（四）数据编织方面

业务部门不同，数据特征和数据分析需求也不尽相同，这就导致不同的场景有着各自的海量数据处理特征，进而衍生出各自的海量数据处理平台。若非全局已经采用数据网格的思路构建，由于技术栈差异和数据标准不同，建设周期的不同，不同平台的数据湖、数据仓库、数据湖仓之间，很难形成高效的联合利用，要么需要持续投入精力确保不同单元之间的数据交互可靠与可用性，要么就随着时间的推移逐步变成各自的处理逻辑，最终造成即便是一份数据源，经过各自逻辑处理后，不但数据标准可能不一致，甚至数据细节也出现不一致的情况。

数据编织便是为了解决该问题的一套数据架构解决方案，实现集中、连接、管理、治理来自不同系统和应用的数据，以避免数据的标准不一致、数据被处理的逻辑不统一、数据沼泽化等问题。相较于传统方式，数据编织引入了 AI、机器学习和数据科学等技术手段，将基于人工的静态数据管理逐步向支持数据动态整合的方向演进，进而实现数据工程能力和特征工程能力的有效整合，并保持自动执行数据管理流程的持续有效，确保数据的统一、清晰、丰富、有效。最终，数据可随时被用于分析、AI 以及机器学习等应用场景，也可以随时被用户即时探索，亦或是被开发者进行数据产品或业务系统的高效构建，以适应快速变化的业务需求。

数据编织本质上是一种理念实践。不同于数据湖仓对数据持续集成、清洗、加工等对数据中内容的关注，数据编织是通过数据源的自动检测和元数据的主动发现，增强数据与业务的关联和实时性；通过数据知识图谱的构建，加强数据价值呈现；通过数据自动编排和动态集成，形成动态可持续的数据服务。

部署数据编织技术后，用户能实现更准确、更高效、更智能的业务运营。金融机构可以通过各种渠道掌握有价值的数据，如通过客户的银行卡和信用卡使用情况、投资、保险以及税务申请等。金融服务提供商可以基于数据编织技术，管理、分析、保护这些敏感且高价值的数据。

数据网格和数据编织均提供了跨越多种技术和平台的数据架构方案，但这二者并非二选一的关系，尤其是面对如金融业需求复杂且数据海量的场景，可以将二者进行必要的整合，以应对复杂的业务需求。例如在细节方面，采用数据网格的理念，保持最大程度的数据效率和创新力，实现微观程度的持续数据价值呈现。而在整体上，采用数据编织的理念，将数据网格所构建出的数据产品形成有效的数据管理，以助力实现业务与业务之间、数据和数据之间、业务和数据之间的数据价值挖掘与利用。

六、实践案例

（一）中国工商银行实践案例

1. 案例概况

工商银行大数据平台是以数据共享、资源统筹、软件服务化的云理念打造的具备海量数据存储、批量计算、流计算等能力的企业级大数据云基础设施，融合了关系型、非关系型数据处理技术，为应用系统提供开箱即用的大数据服务。工商银行大数据平台全面实现了国产化，具有容量大、算力强、功能完备、算法齐全的特点。

工商银行大数据平台从用数领域、垂直领域、通用领域和大数据基础服务领域等多个维度，构建了完善的技术体系，全方位满足上层便捷用法的需求，平台架构如图 15 所示。

图 15 工商银行大数据平台架构图

用数领域：打造快捷用数技术平台，赋能大数据研发和业务用数分析场景。即时 BI 提供全面的数据分析与图表展现服务，降低数据探索门槛，提升价值转换效率。大数据开发工作站提供全面的研发与测试服务，提升大数据研发人员工作效能。大数据运营工作站提供资源调配、资源治理、服务运维监控服务。

垂直领域：打造大数据领域能力突出、边界清晰的技术平台，实现面向大数据垂直领域的技术赋能。实时数仓提供秒级/分钟级响应的实时采集、实时计算、实时分析能力。批量计算提供海量结构化或半结构化数据的存储和大规模并行计算能力。流计算提供毫秒级响应的事件驱动式逐条分析、统计、处理计算能力。联机分析提供海量数据高并发的键值查询、全文检索、向量检索能力。对象存储提供面向海量非结构化数据进行高并发联机存取的能力。

通用领域：打造基础数据服务技术平台，实现面向通用数据处理领域的技术赋能。数据交换提供基于文件、增量日志等形式的通用数据交换能力。数据安全提供通用的数据全生命周期安全保护解决方案。

大数据基础服务：提供统一的资源调度、日志中心、运营监控等公共服务能力。

2. 案例成果

工商银行基于大数据平台服务能力，覆盖了秒、分钟、小时、日等全时效的数据处理场景，共支撑工行近 400 种分行应用及子

公司的业务开发，覆盖了监管报送、风险防控、客户营销等多个关键业务领域，如:

（1）损益预查询业务通过大数据平台批量计算服务，大幅提升处理时效。基于数据复制技术将业务系统数据实时写入大数据平台,并进行数据批量计算,从原先每天 10 轮(平均 30 分钟）查询提升到 30 轮（平均 15 分钟）。

（2）信用卡交易实时反欺诈系统基于大数据平台流计算服务，全面提升实时反欺诈能力，降低了金融交易风险。通过实时数据处理和智能分析，并将加工后的结果实时返回（平均响应时间毫秒级），实现了事中风险拦截。

（3）法人客户营销系统使用实时数仓服务实时统计资金流入流出情况,供法人客户经理便捷获取,挖掘客户资金变动规律,更好地帮助客户合理管理资产，拓展新的营销点。

随着数字化转型的深入推进,工商银行借助大数据平台服务能力，不断为各类决策提供更加实时、更加精准的数据支持。

3.经验总结

工商银行大数据平台在海量数据处理中数据架构、数据处理时效、数云融合等方面的建设经验如下:

实时数仓方面: 引入了 Hudi 技术，提供数据更新能力，显著提升了数据入湖、数据加工时效。同时引入 ClickHouse 丰富了数据分析场景。整体形成秒级、分钟级、小时级各类时效场景支

撑能力，支撑业务端到端高时效的数据感知、分析决策、行动和反馈。

数云融合方面：通过建设云上统一存储服务，实现存算分离部署形态，解耦大数据存储与计算资源，并在架构上独立建设统一元数据服务，全局形成统一的数据视图。基于云上存算分离形态，多个大数据集群实现了统一存储，元数据统一管理，全面支撑数据中台“全”数据融合分析，可减少集群间之间的冗余存储。借助云平台弹性的优势，可避免资源按业务高峰满配，造成闲置资源浪费。基于云平台，资源调配更加灵活，可错峰使用，提升整体资源利用率。

（二）中国银行实践案例

1. 案例概况

中国银行积极响应国家大数据发展战略，启动数据治理体系改革，从组织架构入手，实施中国银行企业级数据平台建设项目，并同步落实国家“十四五”规划，构建自主可控的大数据平台。平台基于国产化大数据技术栈，使用国产定制 Hadoop 生态技术的开源技术组件，搭建数据湖基础底座，借助大数据平台的海量数据处理能力，沉淀、存储全行数据资产，平台建设成果如图 16。

图 16 中国银行国产大数据平台建设成果

2. 案例成果

由于 Hadoop 技术在复杂数据关联计算方面缺乏优势，不适合应用于数据仓库的模型建设和指标加工，因此中国银行采用了“湖仓混搭”架构。以“Hadoop 技术”构建贴源数据层、归集数仓及数据应用结果数据，发挥其对多态、复杂结构数据的归集、存储和处理能力，搭载适用于海量数据处理、交互式分析和实时

计算与访问的配套组件，提升数据应用效率，降低实施运维成本。利用“MPP 数据库”搭建数据仓库基础主题层和汇总共享层，解决复杂数据关系下的关联计算问题，保障数据仓库模型架构稳定和数出同源。

中国银行湖仓建设采取全自主研发应用体系、湖仓混搭部署架构及自研软硬件基础设施，实现大数据平台全方位自主可控。数据治理项目上线至今，数据湖已积累了海量数据资产，单集群架构已不满足存储需求，同时考虑运维管理风险等因素，因此向多集群架构进行扩展，解决规模限制等问题。并通过 Spark 等查询引擎、开源 HDFS 联邦、Hive MetaStore 等技术满足跨集群数据访问需求。另外，在跨集群联邦访问机制在使用时，将多集群搭建在局域网下的同一个数据中心，避免了跨地域网络下的大规模数据交互带来的网络延迟影响，同时避免大规模使用跨集群查询、跨集群数据访问，大数据开发框架如图 17 所示。

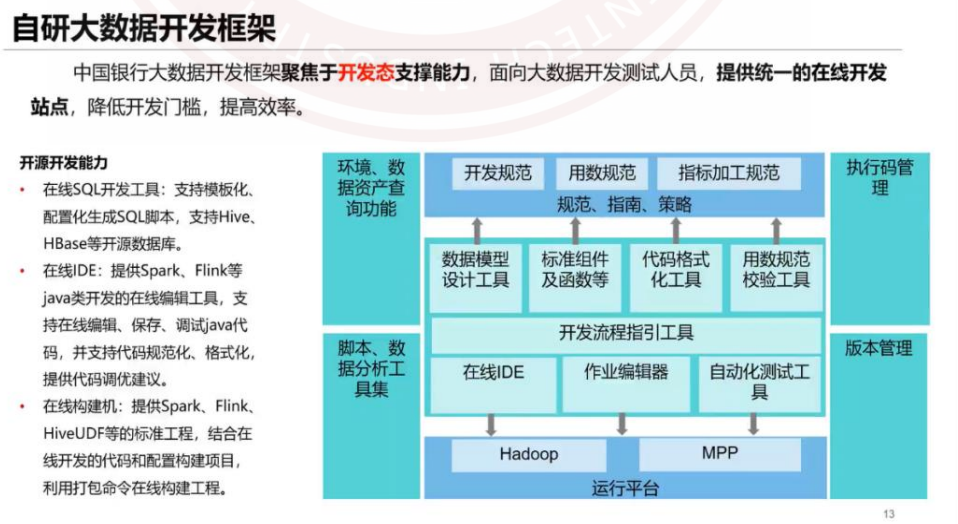


图 17 中国银行自研大数据开发框架

3. 经验总结

由于金融业面临着需求场景多、数据处理过程复杂、技术生态庞大且繁杂等问题，大数据应用开发存在门槛高、开发人员学习成本高等问题。因此，中国银行聚集于开发生态支撑能力，面向大数据开发测试人员自研一套大数据开发框架，提供统一的在线开发站点，降低开发门槛，提高效率。

后续，中国银行还将探索湖仓一体架构，考虑采用开放式存储引擎的方式替代原有的湖仓混搭架构，以更好地支持流批一体，提高数据时效性，降低开发运维成本。

（三）兴业银行实践案例

1.案例概况

兴业银行近几年在云化计算和存算分离数据湖等多项大数据技术方面进行了深入的探索和架构转型，取得丰硕的成果。经过不断地积累，目前兴业银行已自主研发落地了基于云化计算与存算分离数据湖为主要关键技术的大数据架构，不仅全面提升了大数据性能，还大幅度提高资源利用率，实现了降本增效。

兴业银行作为一家广泛使用大数据技术的全国性股份制商业银行，在大数据处理方面遭遇了许多挑战，主要有以下几点：

（1）数据爆发增长：随着兴业银行的生态场景的融入，数据量迎来爆发式增长，数据存储资源面临极大挑战，为解决存储资源的压力问题，兴业银行引入存算分离的架构，将存储资源单独部署运维，大大释放了压力。

（2）计算引擎差异：行内使用大数据的计算需求差异明显（包括计算组件、计算资源、权限等各个方面），实际的管理使用复杂度高。为此，兴业银行基于数据湖仓技术构建流批一体的数据架构，提高数据时效性，解决多链路数据冗余和不一致的问题。同时引入融合计算技术，实现跨数据源、跨数据平台、跨执行引擎的交互式数据访问方式，提升大数据基础平台的事务访问等能力。

（3）资源利用率较低：目前兴业银行大数据平台开发测试环境众多，大数据组件本地化安装部署、运维成本较高。兴业银

行通过实现计算引擎的云化部署，支持计算资源的弹性伸缩，在满足不同数据应用、不同计算引擎的隔离要求同时提高资源利用率。

在经过实践和转型后，兴业银行通过采用这些前沿大数据技术，为金融机构提供了更高效、灵活和可靠的大数据处理解决方案，帮助其更好地应对不断增长的数据挑战。

2.案例成果

兴业银行深入研究大数据组件云化部署、存算分离、数据虚拟化等底层大数据技术，通过自主研发预研湖仓一体底层技术，大幅提升了兴业的技术创新能力，构建了具备兴业特色的大数据平台架构，主要成果体现在以下几个方面:

一是存储区与计算区分离。实现存储计算集群分离部署，以更好地发挥与利用两者的性能，释放计算节点的算力与存储节点存力。存算分离架构在原有 HDFS 存储基础上引入对象存储，热数据存储 HDFS 中，海量归档数据存储在对象存储中，解决 HDFS 的扩展性瓶颈。在计算集群中引入分布式缓存 Alluxio，以减少跨集群间的大量数据传输，提高数据处理效率。分布式缓存 Alluxio 同时还起到了数据编排的作用，作为多个集群的存储路径统一映射入口，简化了计算集群的读写对象复杂度。

二是云化部署与缓存加速。采用 k8s 作为弹性计算资源的底层，通过改造开源大数据引擎实现计算资源的弹性伸缩，采用 Alluxio 分布式缓存，以弥补对象存储、跨集群传输等在数据访

问速度上的不足。通过对大数据组件适配改造，可将组件容器化部署，增加组件资源弹性，提升集群资源利用率，降低成本。云化架构无缝对接多种引擎，降低了环境部署的难度和开发成本。

三是流批一体融合数据湖。以开放式数据表格式 iceberg 为载体，承接落地 bacth 与 streaming 数据，以达到流批一体数据融合。流批一体经过改进和优化，已可以实现秒级数据处理，提升数据处理的时效性。架构上将流数据与批数据融合，既提升了数据新鲜度，又保证了数据最终的精确性。兴业银行湖仓一体底层技术架构如图 18 所示。

图 18 兴业银行湖仓一体底层技术架构

3.经验总结

兴业银行经过长期的论证、测试和投入，通过攻克自身的大数据技术难点，总结了宝贵的经验:

（1）通过存算分离，消除节点不稳定导致的失败，同时将计算集群和存储集群的失败概率进一步降低，大幅提升可靠性，计算任务成功率可达 99.9%，存储使用率也提升 25%。

（2）基于云化计算引擎的改造，可以实现资源预测，保障在不挤占在线资源前提下，最大化挖掘空闲资源。

（3）借助数据湖+计算引擎的方式，进一步加强了数据处理流程的完整性和精准性，缩小集团内部的处理分析工具的差异，提升了大数据整体的效率和性能。

（四）中信建投证券实践案例

1.案例概况

金融业的数字化升级正在迈向新的阶段，而其中持续有效地释放数据要素潜能是重要路径之一。金融业信息化起步早、程度高，随着数字化的发展，不可避免地会有数据源多样异构、数据源孤岛等问题。此外，随着金融科技业务发展的需要，数据采集与集成的逻辑也从注重批量采集转变为批量与实时并重，作为数据要素潜能释放的第一步，金融组织需要拓宽数据要素融合的渠道、以增加其时效价值与应用价值。

中信建投证券在其数字化发展与数据价值变现的道路上，一个主要的痛点就是内部有多种数据源需要统一集成到大数据平台，同时需要覆盖离线和实时接入的场景。其目标是在统一的集成平台之上进行各种集成任务的管理，简化使用成本及运维管理复杂度。

除此以外，近年来数据的安全可控已经成为金融行业整体安全中最重要的环节之一。尤其对于证券行业来说，业务场景的特殊性与复杂性，对数据要素安全和系统自主可控性提出了更高的要求。作为国内领先的券商机构，率先提出了“科技赋能、运营升级”的战略目标，并于 2020 年，战略携手腾讯云探索数字化转型之路。2021 年，腾讯云大数据平台正式落地中信建投证券，并支撑了国产平台稳定运行。

2. 案例成果

中信建投证券的国产大数据平台（以下简称“中信建投大数据平台”）全面基于腾讯云大数据平台，从编译环境搭建、部署适配、功能验证等多维度出发，深度满足用户的安全可控需求，加快金融行业实现全面自主可控的进程。比如全面支持国产生态，涵盖了国产芯片、操作系统、服务器等领域，成功适配鲲鹏、海光、麒麟、中科曙光等厂商产品，全面支持包括数据集成、数据存算、数据分析等多场景下大数据平台的产品落地。

在数据采集方面，中信建投大数据平台搭载了高效稳定的数据集成平台，提供复杂网络环境下、丰富的异构数据源之间高速稳定的数据移动及同步能力，可以通过快速连接和融合云上或云下自建的各种数据，解决数据平台构建、数据库迁移备份，以及业务升级、整合，数据访问加速、全文检索等多个场景中数据整合和同步问题。国产的金融级数据集成平台，也从基础设施上解决了数据要素的安全保护问题。

在数据要素的信息价值探索方面，中信建投大数据平台搭载了全场景大数据存储、分析和管理工具，拥有优化的开源组件、腾讯云自研组件与工具。尤其在大规模关系数据的多维分析场景中，通过腾讯云的自研国产数据仓库引擎，实现企业级数仓能力和万亿级关联查询秒级分析，同时升级海量数据汇聚能力，全面提升数仓构建和大数据湖仓一体方案。

中信建投大数据平台基于腾讯云的统一数据集成平台，对各种不同数据源进行配置、管理，以及元信息同步，通过离线任务及实时任务实现不同场景下的数据集成。

目前，在中信建投证券落地大数据平台 120 余个物理节点，其中国产节点超 30 个。平台运行 1700+数据处理任务，其中，国产平台运行 600+数据处理任务，总数据存储量超 2PB。中信建投证券利用腾讯云大数据平台构建了企业级数据湖平台，将企业客户的基本信息、持仓情况、场内交易流水、场外交易流水等数据按监管及业务要求和分层架构完成数据加工，对外提供数据接口供其他业务系统调用；基于腾讯云大数据平台及分析工具，搭建自主报表、分析平台及领导驾驶舱，支持中信建投证券全面提升企业经营管理水平。在实时处理场景中，将腾讯云大数据平台应用于投资者适当性持续管理、两融客户账户资产变动管理、断点客户二次营销及基金投顾实时数据统计等业务场景，有效地及时了解客户情况，对投资者保护、业务营销策略动态调整和改进具有十分重要的意义。除此之外，中信建投证券还利用腾讯云大数据平台应用于实时与离线日志分析中。目前整个平台超过 50 个租户，支撑近 20 个团队使用，在安全审查环节得分名列前茅。

3.经验总结

在建设中信建投大数据平台的过程中，总结了以下经验，可以为后续的建设提供参考：

（1）零售/CRM 数据支撑：为满足各销售条线对销售管理、销售数据及时性、精细化的要求，大数据平台在数据资产方面应提供客户管理、渠道管理、报表管理等功能以支撑相应业务需求。

（2）服务零售场景：由于开盘时间固定，应当根据实际情况对那些与开盘时间紧密相关的任务设置启停时间；由于金融数据的重要性，大量场景下不允许数据偏差存在，针对数据可靠性要求极高的特征，因此要对大量实时任务设置夜间数据修正的离线任务。

（3）零售管理驾驶舱实时指标统计：需要面向零售业务设计实时数仓，需要获得开户统计、客户服务、APP 运营几个主题的统计指标

（4）实时 ETL：大数据平台应提供实时 ETL 能力，以满足业务人员在开盘期间快速查询客户某个时间段内的交易流水明细数据。

（5）金融资讯数据：新闻公告研报、货币证券市场、宏观行业、其它衍生信息等需要集中处理，提供的资讯数据服务会影响客户各个业务条线，因此需要将原有的缓存程序、业务逻辑整合到 Flink 中统一实现，最终使整个资讯数据处理过程得到了集中管理，缩短链路，节约了传输时间，降低了架构的复杂度，提升了数据生产到应用全链路输出效率。

（6）实时并发查询量：业务人员在开盘期间快速查询客户某个时间段内的交易流水明细数据，资金流水明细达到几十亿的数据，数据量很大的情况下支持快速查询。

（五）上汽财务公司实践案例

1.案例概况

上海汽车集团财务有限责任公司(以下简称“上汽财务公司”或“公司”)是上海汽车集团股份有限公司(以下简称“上汽集团”或“集团”)所属的非银行金融机构，于1994年5月在上海成立。公司经过多年持续快速发展，已经形成汽车金融、公司金融、投融资三大业务板块，走出一条“创新引领未来、和谐共创价值”的产融结合道路。

近年来汽车金融行业，普遍面临汽车销量下滑、新能源车型冲击、零售客户质量下降,并且需要风险可控和宽松准入标准下，为渠道经销商量身定制提供金融服务等挑战,上汽财务公司紧跟数字产业化和产业数字化的大趋势，以数字化转型项目为牵引，建设公司数据中台并规划实施汽车金融实时数据湖仓项目,综合考虑汽车金融业务场景需求和数据统一性、易用性、性能服务等技术要求，完善数据体系架构，沉淀业务数据和用户行为数据，建设覆盖申请进件、放款跟踪、逾期回收等汽车金融全生命周期的主题域模型，落地业财一体化分析、营销洞察分析体系、客群资质监控、客户经理移动服务平台等应用场景，提供业务洞察、风险分级及预警、利润测算、客群分析等数据服务，实现业务数据化、数据资产化、资产价值化的增长闭环，支持数据赋能与业务转型。

2.案例成果

汽车金融实时数据湖仓项目实现了以下目标:

一是完成业务系统数千张表的接入，上线数百个数据模型，每日处理记录数达亿级，实现了汽车金融零售业务、批发业务、业财分析数据等主题域模型搭建，覆盖汽车金融业务申请进件、放款跟踪、逾期回收等全生命周期。

二是统一元数据管理,将数十个业务系统中数万张表的业务元数据、技术元数据与管理元数据紧密结合，建立“事先建标、事中落标、事后对标”的数据标准体系，践行“计划-执行-反馈-优化”的闭环质量问题统一管理流程与分级分类、动静脱敏的数据安全措施。

三是落地业财一体化分析、营销洞察分析体系、客户资质监控、客户经理移动服务平台等应用场景，提供业务洞察、风险分级及预警、利润测算、客群分析等数据服务。

传统汽车金融模式业务场景复杂，数据标准化难度高、数据复用难度大、数据管理标准化率低，汽车金融实时数据湖仓项目围绕上述情况，重点解决影响数据价值落地的核心难题，具体如下:

（1）建设统一开发平台，遵循“DataOps”的研发运营一体化原则，基于 TBDS DataStudio（一站式数据开发平台）实现多个团队统一协同工作,践行“一份数据、多次使用”的设计理念,采用流式接入的方式入湖业务数据,搭建离线实时混合的数据仓

库模型，进行明细、汇总层模型建设，并且提供字段级精细化权限控制能力，以满足监管合规要求及公司制度要求。目前已实现数千张表、每日亿级数据记录接入数据中台，数百张数据模型表的计算，支持数百张报表的实时查询，系统运行稳定，满足项目需求。

（2）夯实数据基础，基于 Iceberg 组件构建近实时数据湖仓，实时 CDC 入湖数百张表，优化实时集成技术，减少资源消耗，开发数据一致性检验程序，保障数据前后一致性；同时建立监控应急体系，保障实时入湖的数据完备性，解决公司数据库分散无法集成，导致无法动态捕捉变化而造成数据冗余的问题。

（3）实现公司级业务元数据、技术元数据与管理元数据的统一管理，提供全领域的元数据全文检索服务和血缘管理功能，通过数据质量工单流程管理，快速定位问题到表到人，实现数据质量问题分发、流转、跟踪、解决的闭环管控，通过分级分类、动静脱敏实现数据安全保障，建立公司级基础数据标准和指标数据标准体系，让所有数据有标准可依。

（4）构建监控应急平台，实现数据重复、数据任务、数据组件等监控告警，从而及时发现问题、定位原因并完成修复；构建数据治理平台以及包括元数据、数据标准、数据质量与数据安全体系的系统化工具。

3.经验总结

汽车金融实时数据湖仓项目全面梳理汽车金融业务数据资源，沉淀业务数据和用户行为数据，完成数据治理和质量提升，充分发挥和挖掘数据的价值。

在平台建设方面，规划拓展大数据基础存算平台、数据全链路工具、数据资产管理工具和数据开发服务能力，落地涵盖数据汇聚、处理、治理、服务等数据全生命周期的数据平台体系。以数据流水线为核心，通过可视化拉拽方式支持在线代码开发测试、作业调度和任务维护等核心功能，支持多级多组以及“用户-组件”级的身份认证和权限管控，实现集群监控预警的统一运维中心。

在数据治理提升方面，将数据治理与数据生产相结合，建设公司级元数据标准、基础数据标准和指标数据标准规范，遵循MOF标准规范和CWM规范构建全域元数据管理；完成数据治理平台系统建设，实现元数据全景视图和全链路追踪分析，形成数据资产的版本化管理。

在业务场景建设方面，全程参与财务公司数字化转型建设，从数据层面完成各类构想场景可行性分析，构建覆盖申请进件、放款跟踪、逾期回收等汽车金融全生命周期的主题域模型，落地业财一体化分析、营销洞察分析、客群资质监控、客户经理移动服务平台等应用场景，提供业务洞察、风险分级及预警、利润测

算、客群分析等数据服务，将数据用于各业务部门的运营管理工作当中，实现数据化决策能力的建设目标。