DELL Technologies

技术白皮书

基于Apache Iceberg和Dell EMC ECS对象存储构建数据湖

版本号 2

版本控制

日期	描述
2021 年 10 月	第一版
2022 年 1 月	更新 4.2，4.3 章节的 SQL 脚本

目录

摘要

随着大数据存储和处理需求的多样化，如何构建一个统一的数据湖存储，并在其上进行多种形式的数据分析成了企业构建大数据生态的一个重要方向。Apache Iceberg 成为了大数据、数据湖领域炙手可热的方向。同时，随着对象存储在数据湖中所扮演的角色愈发重要，我们希望能够提供一套完全基于对象存储的 Catalog 实现，基于此，我们对比了 Dell EMC ECS 对象存储与 HDFS，并阐述我们如何使用 Dell EMC ECS 构建数据湖。

1 介绍

数据湖是一种在系统或存储库中以自然格式存储数据的方法，它有助于以各种模式和结构形式配置数据，通常是对象块或文件。数据湖的主要思想是对企业中的所有数据进行统一存储，从原始数据（源系统数据的精确副本）转换为用于报告、可视化、分析和机器学习等各种任务的目标数据。数据湖中的数据包括结构化数据（关系数据库数据）、半结构化数据（CSV、XML、JSON 等）、非结构化数据（电子邮件、文档、PDF）和二进制数据（图像、音频、视频），从而形成一个容纳所有形式数据的集中式数据存储。

数据湖从本质上来讲，是一种企业数据架构方法，物理实现上则是一个数据存储平台，用来集中化存储企业内海量、多来源、多种类的数据，并支持对数据进行快速加工和分析。从实现方式来看，目前 Hadoop 是最常用的部署数据湖的技术。

数据湖主要有 4 个方面的特点：

第一个特点是存储原始数据，这些原始数据来源非常丰富；

第二个特点是支持多种计算模型；

第三个特点是有完善的数据管理能力，要能做到多种数据源接入，实现不同数据之间的连接，支持 Schema 管理和权限管理等；

第四个特点是灵活的底层存储，一般用 S3、HDFS 这种分布式文件系统，采用特定的文件格式和缓存，满足对应场景的数据分析需求。

基于这些特点，数据湖数据主要应用于以下场景：

- 数据的导入；
- 数据的实时同步；
- 数据的实时读写。

如图 1 所示的典型数据湖结构，在一个数据湖解决方案中，用户通过诸如 Apache Flink 等平台，通过自带的数据定义或者第三方框架的数据定义，对存储在存储层的数据进行管理，包括查询和修改等。

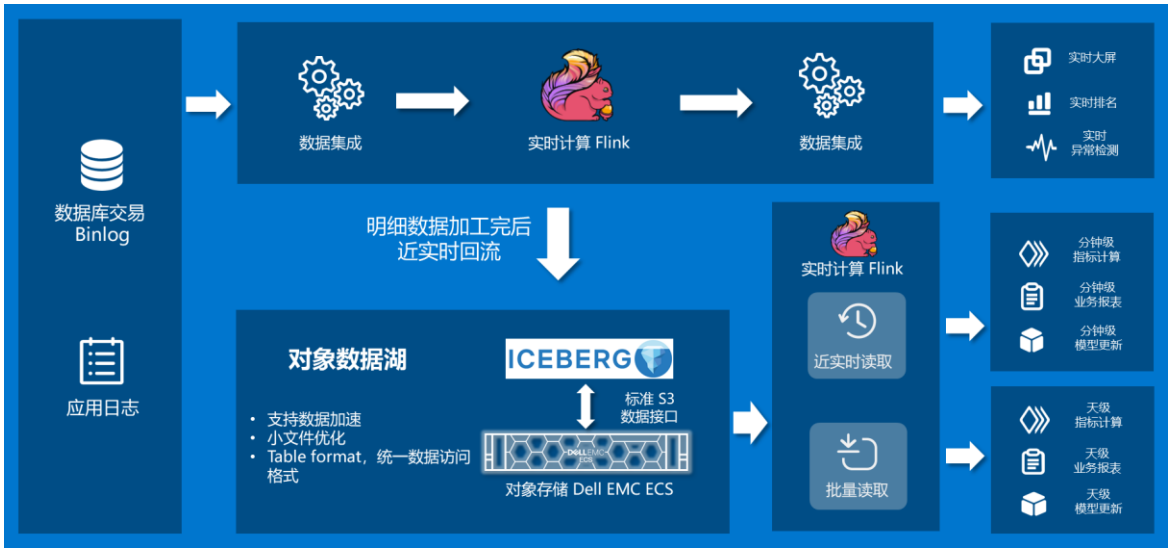

图 1　数据湖架构

其中，Apache Iceberg 则是一个新兴的数据定义框架，它适配了多个计算引擎，并具备了极强的扩展性，使得存储层可以对其进行适配。基于 Apache Iceberg 的扩展性，我们将其和 Dell EMC ECS 对象存储结合，以提供一个强大的一站式数据湖解决方案。

2 Apache Iceberg 介绍

在 Iceberg 中，表数据由这几部分构成：

- 版本信息：指的是当前表格的版本，包括版本所对应的 Schema、Partition、Manifest List 以及历史的 Manifest List。
- Manifest List：指的是表格的数据快照，用于引用 Manifest 文件。
- Manifest 文件：指的是 Data 文件的聚合，用来提供一个简单的集合合并多个 Data 文件。
- Data 文件：指的是表格的原始数据文件，数据支持以 Apache Parquet、Apache ORC 和 Apache Avro 格式进行存储。

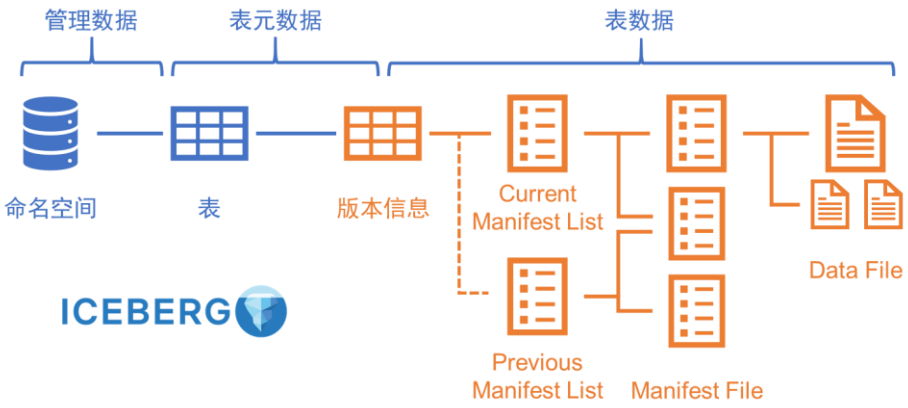

图 2　Iceberg 架构

Iceberg 的表元数据非常简单，仅仅包括当前表的名称和版本信息。所有的 Schema 和 Properties 都由 Iceberg 自身进行管理。

Iceberg 支持了多种表的操作，包括从表中查询数据、向表中插入数据、更新表中的数据、删除指定行的数据和删除指定条件的数据等。但是总结下来，在流程上可主要分为以下两种：修改表格和查询表格。下文将以 Apache Flink 为例，简单梳理 Iceberg 的这两种工作流程。

2.1 Iceberg 修改表格流程

图 3 为向表格中插入数据的基本流程，主要分为两个部分：数据文件的写入和表格元信息的提交。

在写入数据文件时，Flink Data Workers 将从数据源中逐行读取数据，根据当前定义的 Schema，解析行中的数据，计算分区信息，将该行写入对应分区的数据文件中。

当到达检查点时，Data Workers 将会切换写入新的数据文件，同时完成正在写入的数据文件，所有完成的文件会被打包成文件清单。文件清单包含了数据文件的路径和其统计信息，这些信息将被移交至 Commit Worker 对表格的元数据进行变更。在变更时，需要读取当前的表格版本，如图 3 所示为 v006。紧接着将所有的数据文件信息与版本号为 v006 的表格进行合并，生成新的表格元数据，并将版本号更新为 v007。注意，这一过程是线性一致性的，我们将在后面讨论关于这一机制的技术细节。

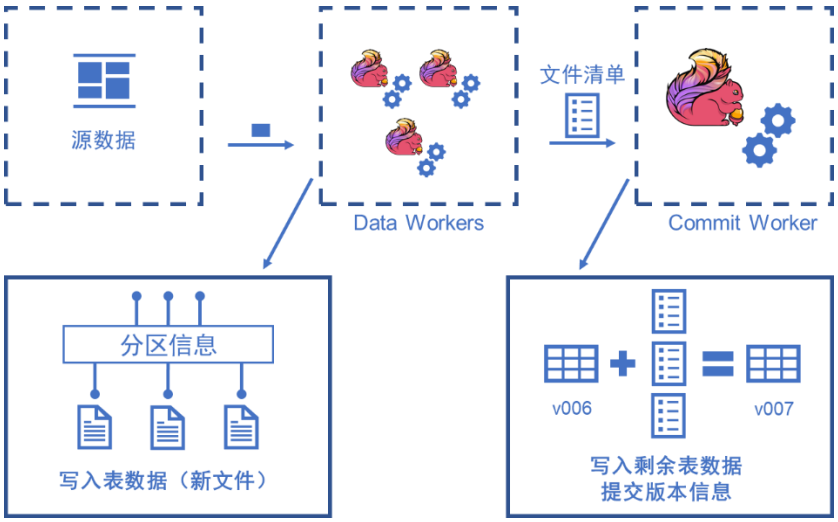

图 3　Iceberg Table 中新增数据

对于其他修改表格的操作，和插入数据的流程一致，即先对数据文件进行变更，之后再将变更提交到表格中，此处不再复述。

2.2　Iceberg 查询表格流程

如图 4，从 Iceberg 表格中查询数据时，首先会获取表格的最新版本（或根据用户的要求使用指定版本），然后从快照中得到所有匹配的 Data 文件清单。之后，这个文件清单会被拆分成任务，委托计算平台执行，每个子任务将读取数据文件并选择匹配的记录，最终将所有任务的结果合并返回。

其中，Data 清单会根据查询条件对 Manifest 文件和 Data 文件进行裁剪，以此来确保尽可能少的 Data 文件进入到最终的检索过程，提高检索效率 。

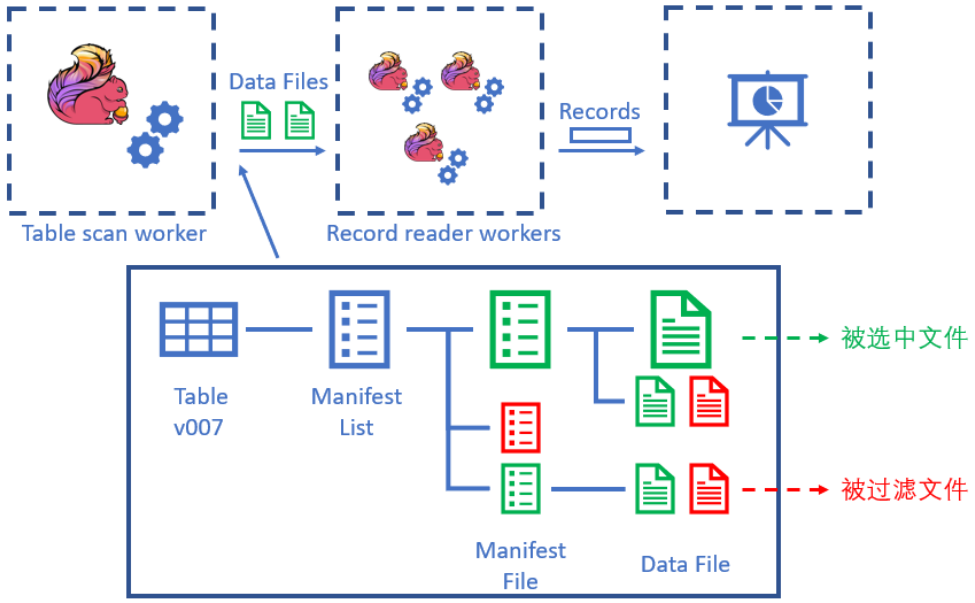

图 4　从 Iceberg 表格中查询数据

2.3 Iceberg Catalog

如图 5，在这些流程中，Iceberg 开放了一些抽象，从而允许开发者定制其中的一些功能，这个抽象在 Iceberg 中被称作 Catalog。

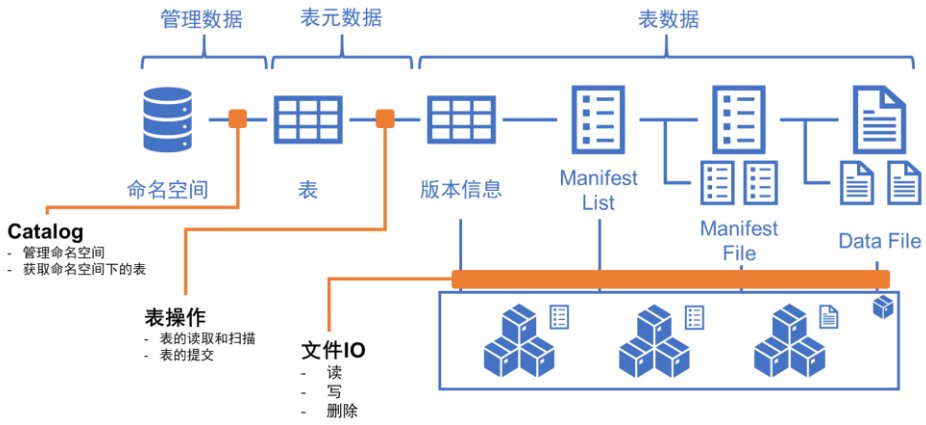

图 5　Catalog 抽象功能

这些操作都依赖存储层的实现，因此对象存储在这个场景下变成了一个重要的选项。

3 使用对象存储支撑 Apache Iceberg

在 Iceberg 的代码仓库中，包含了以下几种 Catalog 的实现：

表格 1　Catalog 的实现

Catalog 实现	数据文件 I/O	元数据/管理数据
AWS	Amazon S3	Amazon Glue
Apache Hadoop	Apache HDFS	Apache HDFS
Apache Hive	Apache HDFS	Hive Metadata Store
Project Nessie	Apache HDFS	Project Nessie

这些 Catalog 实现或多或少地依赖于其他组件或 HDFS，而随着对象存储在数据湖中所扮演的角色愈发重要，我们希望能够提供一套完全基于对象存储的 Catalog 实现。基于此，我们比较了 Dell EMC ECS 对象存储与 HDFS，并阐述我们如何使用 Dell EMC ECS 来实现数据湖的构建。

3.1 Dell EMC ECS 对象存储介绍

ECS 是一种完全软件定义的云存储平台，支持在商用硬件上大规模存储、操作和分析非结构化数据。ECS 专为满足移动、云、大数据和社交网络应用程序需求而设计。可将其作为全包式存储应用装置部署，也可作为能在一组经鉴定的商用服务器和磁盘上安装的软件产品部署。

ECS 使企业能够在全球范围内大规模管理和存储分布式内容的主要优势和特性包括：

• 云规模 —— ECS 是一个面向传统和下一代工作负载的对象存储平台。ECS 的软件定义分层架构促进了无限的可扩展性。其功能亮点包括：

- 全球分布的对象基础设施
- Exabyte+ 规模，对存储池、集群或联合环境容量没有限制
- 系统、命名空间或存储桶中的对象数量没有限制
- 高效处理小型和大型文件工作负载，对象大小没有限制

• 灵活部署 —— ECS 具有无与伦比的灵活性，具有以下功能：

- 设备部署
- 支持认证或定制行业标准硬件的纯软件部署
- 多协议支持：对象（S3/S3a、Swift、Atmos、CAS）和文件（HDFS、NFSv3）
- 多种工作负载：现代应用程序和长期存档
- 使用 CloudPools 的 Data Domain Cloud Tier 和 Isilon 二级存储

- 对当前一代 ECS 模型的无中断升级路径

• 企业级 —— ECS 通过安全且合规的系统中的企业级存储为客户加强对其数据资产的控制，该系统具有以下功能：

- 带有密钥轮换和外部密钥管理的静态数据（D@RE）

- 加密的站点间通信

- 默认情况下禁用端口 9101/9206 以授权组织满足合规性策略

- 针对 SEC 规则 17a-4(f) 合规性的报告、基于策略和事件的记录保留和平台强化，包括高级保留管理，例如诉讼保留和最小最大治理

- 遵守国防信息系统局（DISA）安全技术实施指南（STIG）强化指南

- 使用 Active Directory 和 LDAP 进行身份验证、授权和访问控制

- 与监控和警报基础设施（SNMP 陷阱和 SYSLOG）集成

- 经强化的企业功能（多租户、容量监控和警报）

• 降低 TCO —— 相对于传统存储和公共云存储，ECS 可以显著降低总体拥有成本（TCO）。在长期保留方面，它甚至提供比磁带更低的 TCO。其功能包括：

- 全局命名空间

- 小文件和大文件性能

- 无缝的 Centera 迁移

- 完全符合 Atmos REST

- 低管理开销

- 数据中心占地面积小

- 高存储利用率

ECS 的设计针对以下主要用例进行了优化：

• 现代应用程序 —— 专为现代开发而设计的 ECS，例如下一代 Web、移动和云应用程序。强一致性存储简化了应用程序开发。随着多站点、同时多用户读/写访问，随着 ECS 容量的变化和增长，开发人员永远不需要重新编码他们的应用程序。

• 二级存储 —— ECS 用作二级存储以释放不常访问数据的主存储，同时保持其合理的可访问性。示例是基于策略的分层产品，例如 Data Domain Cloud Tier 和 Isilon CloudPools。

• Geo-Protected Archive —— ECS 充当安全且经济实惠的内部部署云，用于存档和长期保留。使用 ECS 作为存档层可以显著降低主存储容量。为了提高冷存档用例的存储效率，除了默认的 12+4 方案外，还提供 10+2 纠删码（EC）方案。

• 全球内容存储库 —— 包含图像和视频等数据的非结构化内容存储库通常存储在成本高昂的存储系统中，使企业无法经济高效地管理海量数据增长。ECS 支持将多个存储系统整合到一个单一、全球可访问且高效的内容存储库中。

• 物联网存储 —— 物联网（IoT）为能够从客户数据中提取价值的企业提供了新的创收机会。ECS 为大规模非结构化数据收集提供了高效的物联网架构。ECS 不受对象数量、对象大小或自定义元数据的限制，是存储 IoT 数据的理想平台。ECS 还可以通过允许直接在 ECS 平台上分析数据而不需要耗时的提取、转换和加载（ETL）流程来简化一些分析工作流程。Hadoop 集群可以使用通过其他协议 API（例如 S3 或 NFS）存储在 ECS 上的数据运行查询。

• 视频监控证据存储库 —— 与物联网数据相比，视频监控数据的对象存储数量要少得多，但每个文件的容量占用空间要大得多。虽然数据真实性很重要，但数据保留并不那么重要。ECS 可以是此数据的低成本着陆区或辅助存储位置。视频管理软件可以利用丰富的自定义元数据功能来标记具有重要细节的文件，例如摄像机位置、保留要求和数据保护要求。此外，元数据可用于将文件设置为只读状态，以确保文件的监管链。

• 数据湖和分析 —— 数据和分析已成为组织的竞争优势和价值创造的主要来源。然而，将数据转化为宝贵的企业资产是一个复杂的话题，很容易需要使用数十种技术、工具和环境。ECS 提供一组服务来帮助客户收集、存储、管理和分析任何规模的数据。

ECS 客户端访问协议包括 S3 以及其他协议外扩展，例如负载平衡和字节范围更新、Atmos、Swift、CAS、NFS 和 HDFS。 S3、Atmos 和 Swift 的对象访问是通过 REST API 实现的。使用 REST 动作（例如 GET、POST、PUT、DELETE 和 HEAD）通过 HTTP 调用写入、检索、更新和删除对象。CAS 访问是通过专有 SDK 提供的。ECS 原生为 NFS v3 提供服务，并提供 S3a 供 Hadoop 计算节点直接访问 ECS 作为 Hadoop 兼容文件系统。

更多 ECS 信息，请访问 Dell EMC ECS 概述和架构白皮书。

3.2 Dell EMC ECS 对象存储与 HDFS

HDFS 是当前被广泛使用的大数据组件，它拥有一个文件系统抽象，从而让开发者更加便利地使用其作为存储引擎。但是随着对象存储的不断发展，其相较而言更加简单的抽象也能够被大数据生态所使用，进而出现取代 HDFS 的趋势。

在当前的场景下，HDFS 在追加上传和原子性 Rename 上的一些设计对于对象存储而言仍旧存在一些挑战，但是在集群扩展性、小文件存储、多站点部署、存储开销上，对象存储都有着天然的优势。Dell EMC ECS 通过自身的设计解决了追加上传和原子性 Rename 的挑战，并保留了对象存储的天然优势。

3.2.1 Dell EMC ECS 及传统对象存储的优势

3.2.1.1 集群扩展性

HDFS 使用了 Name Node 作为元数据管理服务，所有文件的元数据交由 Name Node 进行管理。因此，Name Node 实际上成为了 HDFS 系统中的单点服务，即无法进行横向扩展。在逐步增长的数据规模下，这样的架构会慢慢地体现出劣势，往往体现在需要使用越来越高的配置运行 Name Node，需要对 Name Node 进行高可用的配置等。

但是，在 ECS 中，集群的横向扩展往往比较容易。因为对象服务的元数据一般使用一致性哈希等方法进行分区处理，因此能够根据数据规模，适时的扩展容量。根据哈希被拆分的元数据可以在不同的服务间移动，则可以有效的实现服务间的 Failover，从而提升服务的可靠性，避免单点故障对于系统可用性的影响。

3.2.1.2 小文件友好

如前所述，受限于 Name Node 的架构，小文件因为其元数据量的占比较高，导致其对于 Name Node 的空间消耗极大。社区使用了多种手段优化小文件存储，但其基本原理都是将小文件合并成大文件，这或多或少对性能和交互便利性产生了一定的影响。

ECS 由于使用了分布式的元数据管理系统，其容量的优势体现了出来。同时，ECS 利用多种介质对元数据进行存储或缓存，在不同速度的存储介质下为元数据查询进行加速，使得其性能保持在较高的水平。以上两点，ECS 对于大量小文件有着容量和性能上的优势。

3.2.1.3 多站点部署

对于存储的数据，如果需要异地备份，或者多机房备份，就需要进行多站点部署。

HDFS 本身并不支持多站点部署。有一些商业软件试图提供多站点支持，但 HDFS 本身并不原生支持这种跨服务的部署。

ECS 则天然支持多站点部署，在这种多站点部署模式下，用户可以配置多种规则进行站点间的拷贝。这使得对于数据高可用性有要求的用户可以非常便利地使用 ECS 达到其合规需求。

3.2.1.4 低存储开销

存储的数据往往需要一些副本机制抵御硬件故障产生的偏差。

HDFS 默认使用 3 副本存储数据，因此对于一次写入-多次读取的场景，其存储开销较大。HDFS 新版本中已经支持 Erasure Coding，但是由于它是基于文件做 EC，所以在小文件时开销会变大，极端情况下当文件小于最小分片大小时，存储开销甚至会超过 3 副本。与此同时，HDFS 上使用 EC 的文件会带来很多限制，比如不能支持 Append 追加上传功能。

在 ECS 中，默认使用 EC，即针对数据进行纠删编码，这样的编码能够减少存储开销，对于 10+2 的 EC 编码，其存储开销仅仅是 1.2 倍，而其 12 段数据中，只要存在任意 10 段即可恢复出原有数据。在这种模式下，数据密度被提升，有限的存储可以发挥出更高的价值。对于小文件，ECS 通过合并小文件，以块来做 EC，可以确保在小文件上保持同样的低存储开销。

3.2.2 传统对象存储面对的挑战

3.2.2.1 数据文件上传时的追加上传

在对象存储中，上传数据需要预先知道对象的大小，因此在追加上传的场景下，其 API 无法像 HDFS 一样天然支持。

如图 6，在具体实现中，追加写的操作需要在本地缓存，得到完整对象后再以整体上传。

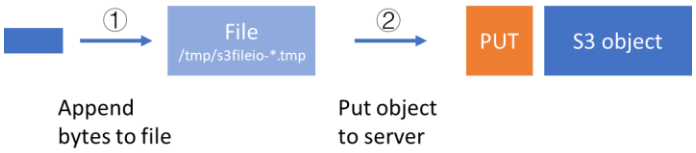

图 6　使用 S3 Put Object 上传小对象

对于比较大的对象，在上传时需要使用 Multi-Part Upload，将数据分块上传，如图 7 所示。利用 Multi-Part Upload，无需将整个文件缓存至本地，可以有效减少磁盘和内存开销，同时避免了缓存整个大文件的时延。Multi-Part Upload 可以支持并行上传多个分片，所以通过实现异步上传机制，让多个追加分块并行上传，实际上可以提供比 HDFS 的单个追加上传更高的性能。

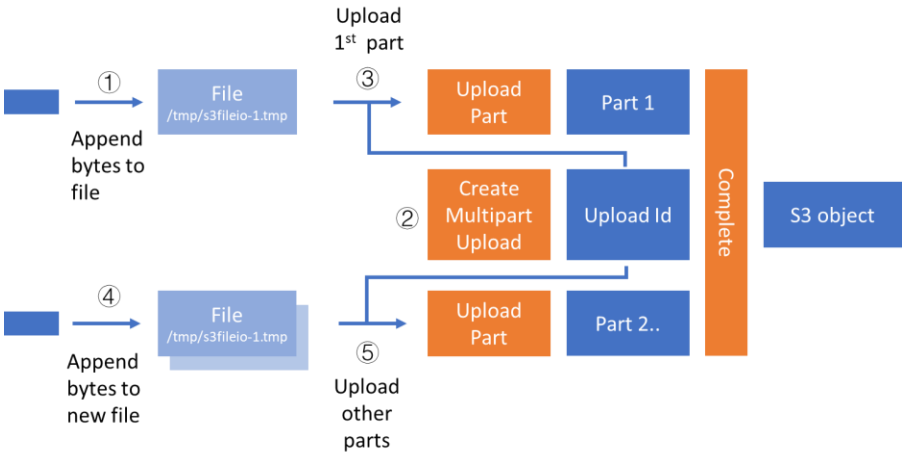

图 7　使用 S3 Multi-Part Upload 上传大对象

尽管在追加上传时需要在本地进行缓存，但是对象存储额外提供的不可变性，能够保障数据以完整的状态存在，而避免出现部分提交的对象。例如，在上传过程中报错，存储中并不会残留部分数据，所有的失败数据实际上都未完成上传。

3.2.2.2　元信息提交时的原子重命名（Rename）

在数据插入到 Iceberg Table 的流程图中，我们可以看到有一个最终递交的步骤。这个步骤不仅仅需要写入最新的版本，还需要将其它同时执行的操作拒绝并回滚。

如图 8，在多个 Worker 对同一个表格进行提交时，如果它们同时获取到了相同的 Table 源版本，那么在最终提交时，仅仅有一个 Worker 能成功完成提交，而其它 Worker 必须失败，并重新尝试以新的版本进行提交。这意味着需要有一个机制来实现线性事务提交。

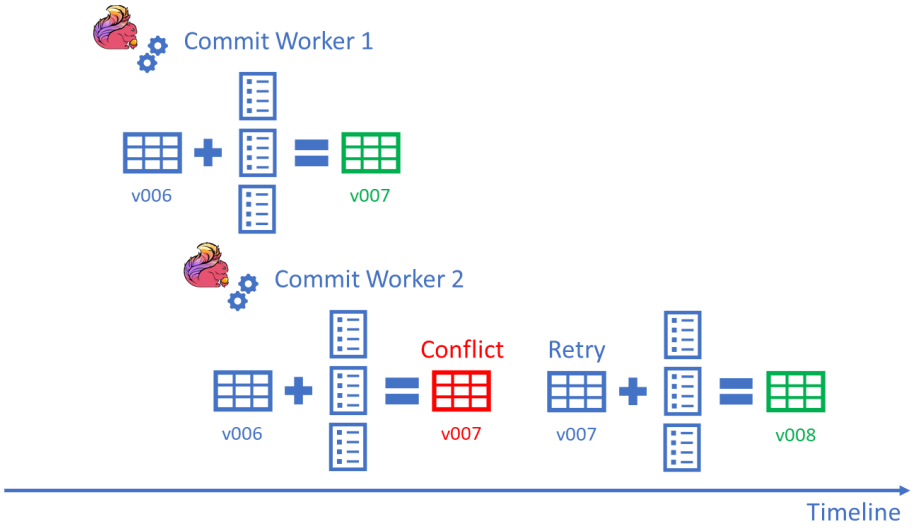

图 8　Apache Iceberg 的线性一致性

在官方内置的 Catalog 实现中，主要存在两个流派的设计。一个是基于 Apache Hadoop 的实现，如图 9 所示，使用了原子的重命名确保特定的版本被唯一提交。

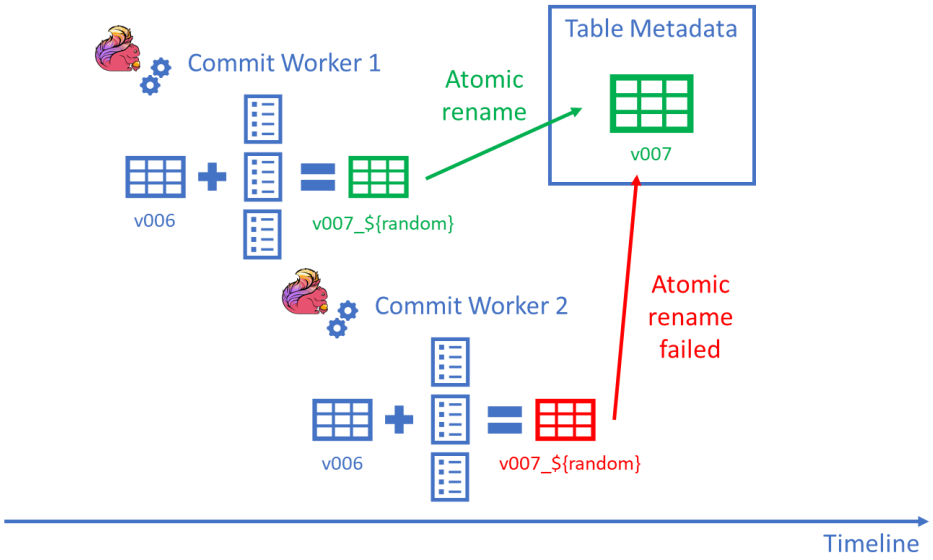

图 9　Apache Hadoop Catalog 使用原子 Rename 处理并发冲突

另一种是基于一个第三方锁来实现。

虽然对象存储能保证数据最终被写成功，但是却无法让并发写冲突中失败的一方知道自己的失败。例如，在图 8 所示的场景中，会产生两个 v007 版本，之后会选择其中一个 v007 更新至 v008，导致另一个 v006 - v007 的变更会被后续版本完全丢弃。

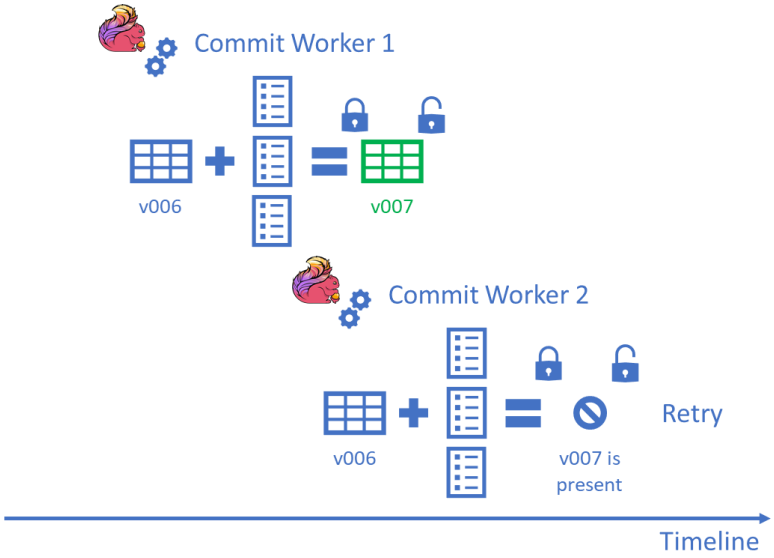

图 10　使用锁机制处理并发冲突

综上，在对象存储上，一般使用锁机制，或者使用业务流程，保障提交的数据一致性。如图 10，在使用锁机制时，会在最后提交版本 v007 时，锁定当前表格，检查是否已有 007 版本，如果 007 版本已经存在，则判定本次提交失败，数据再次进行提交。

3.2.3　Dell EMC ECS 对象存储的解决方案

Dell EMC ECS 作为对象存储的一种商业实现，在如上所述的两个挑战中有扩展解决方案，我们为 Iceberg 定制了 ECS Catalog。

一方面，ECS 支持使用 Append 操作，如果用户不希望在本地写入临时文件或占用过多内存缓存数据文件，可以直接使用 Append 操作提交数据，避免写入文件时产生的资源浪费。

另一方面，ECS 支持使用 If-Match 对对象进行操作来解决 Rename 的挑战，如图 11。在提交版本的时候，需要携带旧版本的 E-Tag 信息作为更新条件，即可在更新时实现原子性操作，ECS 保证了只有在前一版本匹配指定 E-Tag 时才会更新数据，错误则会同步返回，让 Commit Worker 使用新版本重新提交。

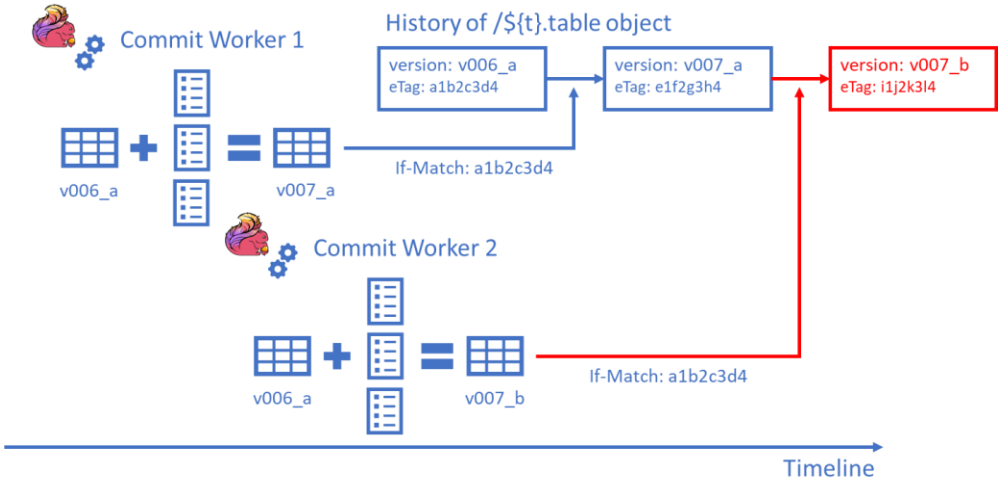

图 11　ECS 使用 If-Match 提交 Table 信息处理并发冲突

基于以上两方面，我们可以使用 ECS 支持 Iceberg 的全部需求，而不依赖其它存储方案或锁方案。这有助于让用户使用尽可能小的成本来完成数据湖的构建，并为用户提供更全面的支持。

4 案例参考

4.1 接入配置

下文将提供一些步骤，用来帮助您使用 Apache Iceberg 和 Dell EMC ECS 构建数据湖解决方案。

本案例中同时也使用到 Apache Flink 的 Flink SQL API 实现数据入湖和数据访问的相关操作。

注意，这假设您已经拥有能够正常工作的 Apache Flink（版本号 1.12.4）集群和 Dell EMC ECS 设备（版本号 3.6），并使用 Apache Iceberg（版本号 0.12.0）构建解决方案。

第一步，在 Dell EMC ECS 上创建用户、Bucket。

Apache Iceberg 需要使用指定的用户将数据存储在指定的 Bucket 上。因此，需要预先在 Dell EMC ECS 上创建用户和数据所存储的 Bucket。

这些操作既可以在 ECS Portal 上完成，也可以通过管理 API 创建用户，S3 API 创建 Bucket，以满足自动化部署的需求。

请访问 https://www.dell.com/support/kbdoc/en-us/000181997/02377692 了解更多细节。

第二步，获取 Apache Iceberg 运行时和 Dell EMC ECS 连接所使用的相关运行时。

可以在 Apache Iceberg 的官方网站上获取其对应的 Flink 运行时。注意，这个运行时属于 Apache Flink 的扩展功能，并不包含 Apache Flink 本身。同时，也需要获取 Dell EMC ECS 连接所使用的运行时。这两个 Jar 文件的关系如下图所示：

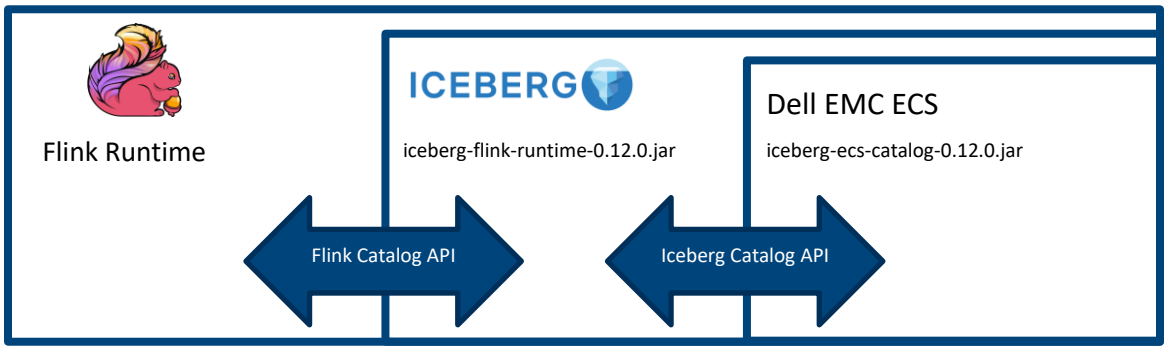

图 12　Flink 与 Iceberg 和 ECS 的扩展关系

这些文件需要导入到 Flink 的运行时中，根据使用的工具不同，可以使用如下方式引入 Jar 包：

1. 将 Jar 文件复制到 Flink 集群中，需要覆盖所有节点。

2. 或者，如果使用 sql-client.sh 工具，则需要通过-j 选项配置 Jar 路径：-j iceberg-ecs-catalog-0.12.0.jar”

3. 或者，如果使用 pyflink，则需要如下所示的代码，通过配置"pipeline.jars”导入 Jar：

```
from pyflink.dataset import ExecutionEnvironment
from pyflink.table import BatchTableEnvironment, DataTypes, EnvironmentSettings,
TableConfig

env_settings =
EnvironmentSettings.new_instance().in_batch_mode().use_blink_planner().build()
t_env = BatchTableEnvironment.create(environment_settings = env_settings)

work_space = !pwd
work_space = work_space[0]
jars = ["iceberg-ecs-catalog-0.12.0.jar", "iceberg-flink-runtime-0.12.0.jar"]
pipeline_jars = ";".join([f"file://{work_space}/{jar}" for jar in jars])
t_env.get_config().get_configuration().set_string("pipeline.jars",
pipeline_jars)
```

如果需要更多相关信息，请参考 https://iceberg.apache.org/flink/ 。

第三步，在 Flink 中创建 Catalog。可以使用如下所示的 SQL 创建一个名为 ecs 的 Catalog：

```
CREATE CATALOG ecs WITH (
    'type'='iceberg',
    'catalog-impl'='org.apache.iceberg.dell.EcsCatalog',
    's3.access.key.id' = '$access_key_id',
    's3.secret.access.key' = '$secret_access_key',
    's3.endpoint' = '$endpoint',
    's3.base.key' = '$iceberg_bucket'
)
```

其中，重要的参数包括如下表格 2 所示：

表格 2　键值说明

键	值	说明
type	iceberg	用来标记该 Flink Catalog 使用了 Iceberg
catalog-impl	org.apache.iceberg.dell.EcsCatalog	用来标记 Iceberg 使用了 ECS 作为 Catalog
s3.access.key.id		用户名
s3.secret.access.key		用户授权
s3.endpoint	http://x.x.x.x:xxxx	Dell EMC ECS 的 S3 接入地址
s3.base.key	bucket 或者 bucket/prefix-a	Bucket 和对象名的前缀，后者可选

注意，Flink 和 Iceberg 并不限制用户创建多个 Catalog，因此，可以根据实际的情况，进行数据的拆分和授权。

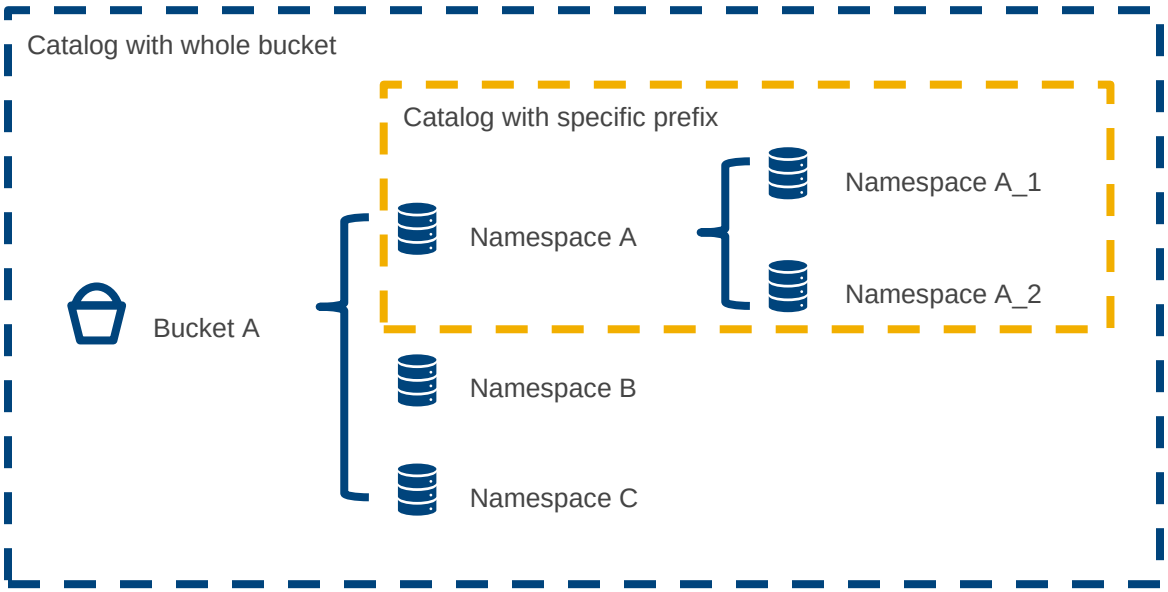

图 13　Catalog 举例

如图 13，既可以创建一个 Catalog 覆盖整个 Bucket 的数据，也可以选择特定的 Namespace 作为前缀，以提供该命名空间下的数据访问能力。

4.2　归档外部数据，进行查询

对于结构化或半结构化的外部数据，可以使用 Flink 的相关 SQL 进行简单的转换和处理，并导入到基于 Dell EMC ECS 的 Iceberg 中。

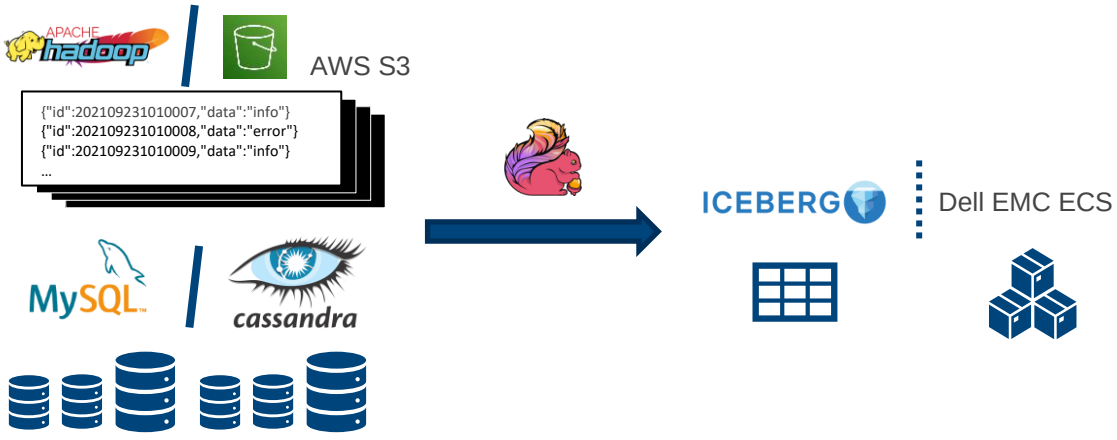

图 14　数据归档

例如，对于一个外部的 JSON 数据，我们可以使用如下的 SQL 将其导入到 Iceberg 表格中：

```
CREATE TABLE `external`.`sample` (
    id BIGINT,
    data STRING
) WITH (
   'connector' = 'filesystem',
   'path' = 'hdfs://namenode:50070/data/sample.jsonl',
   'format' = 'json'
);

CREATE TABLE `ecs`.`default`.`sample` (
    id BIGINT,
    partition_id INT,
    data STRING
) PARTITIONED BY (partition_id);

INSERT INTO `ecs`.`default`.`sample`
SELECT
    id,
    id % 10,
    data
FROM `external`.`sample`
```

使用这样的方式，我们创建了一张位于默认命名空间下的 sample 表，同时将 HDFS 上的 JSON 数据导入到该表中。同时，用户可以利用 SQL 对数据进行必要的转换、筛选和分区。

导入数据后，可以使用如下所示的 SQL 对数据进行查询。这只是一个简单的演示，您可以使用更加复杂的逻辑来适配您的业务需求。

```
SET execution.type = batch;

SELECT COUNT(1) FROM `ecs`.`default`.`sample`;
```

4.3 实时数据的导入和查询

随着对数据时效性要求的上升以及消息中间件的普及和发展，更多的数据正在实时地被送入到数据湖中。使用 Flink SQL，可以轻松地对消息中间件的数据进行识别、过滤、拆分，并将其送入到 Iceberg 表格中。在 Iceberg 表格中的数据，提供了准实时的查询能力，您可以通过 Flink 实时查询 Iceberg 表格，获取表格数据的更新。

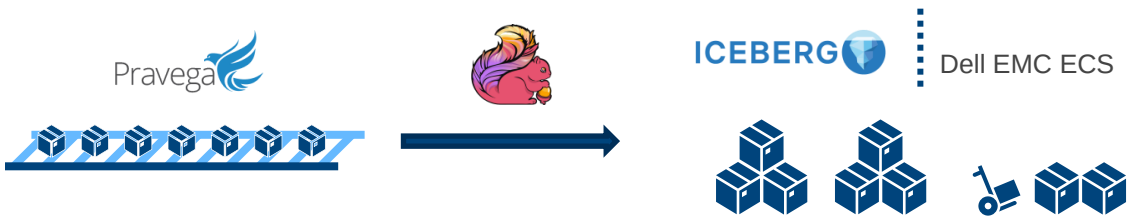

图 15　数据导入与查询

```
SET execution.type = streaming;

CREATE TABLE `external`.`sample` (
    id BIGINT,
    data STRING
) WITH (
    'connector' = 'pravega'
    'controller-uri' = 'tcp://localhost:9090',
    'scope' = 'scope',
    'scan.reader-group.name' = 'group1',
    'scan.streams' = 'stream',
    'format' = 'json'
);

CREATE TABLE `ecs`.`default`.`sample` (
    id BIGINT,
    partition_id INT,
    data STRING
) PARTITIONED BY (partition_id);

INSERT INTO `ecs`.`default`.`sample`
SELECT
    id,
    id % 10,
    data
FROM `external`.`sample`
```

使用这些 SQL，我们可以将 Pravega 中的数据导入到 sample 中，数据会按照指定的间隔时间进行提交，这取决于 Flink 的检查点配置。在数据被注入后，我们可以使用一些简单的 SQL 实时地获取数据。使用如下的 SQL 可以实时地获得当前表格中有多少行数据，随着新的快照被提交，返回的结果也会实时进行更新。

```
SET execution.type = streaming;

SELECT COUNT(*) FROM `ecs`.`default`.`sample`;
```

4.4 多数据源、多表联合查询

在 Flink 中，Iceberg 的 Table 具备和其它数据源相同的功能。因此，可以通过多数据源、多表的查询，完成一些复杂的任务。例如，针对一个订单表，对一个需要间接获取的类型或标签进行分组统计操作：

```
SELECT
    COUNT(*) AS c,
    `external_static`.`catalog`.type AS type
FROM `ecs`.`default`.`order`
LEFT JOIN `external_static`.`catalog`
ON `ecs`.`default`.`order`.catalog = `external_static`.`catalog`.id
GROUP BY `external_static`.`catalog`.type
```

这意味着，您可以轻松使用存储在 Dell EMC ECS 上的 Iceberg 表格，和其它来源的数据源进行诸如数据的统计、分析和预测等。

5 性能报告

许多人询问 Iceberg 在 ECS 系统上的每秒输入/输出（IOPS）大小，但与传统存储系统相比，对象存储系统是建立在 REST API 之上，这些 API 定义与其它文件系统不同类型的离散 I/O 操作。例如，按照 POSIX API 语义实现文件系统的存储系统在对同一文件的 I/O 操作之间会有保留状态，这种状态保持使得一系列可以离散计数的逻辑 I/O 操作成为可能。相反，当使用 REST API 进行对象存储时，将使用单个 API 调用读取文件，正是这种操作和计算方式的差异使其无法使用传统指标。

我们使用每秒事务数（TPS）和每秒兆字节（MB/s）的带宽来表示 ECS 的性能数据，这两个指标更好地描述了对象存储系统的性能参数。

EXF900 是 ECS 产品线中超融合节点的全闪存对象存储解决方案，用于低延迟和高 TPS 的 ECS 部署。根据我们的测试结果，单节点 S3 写性能 10KB 大小的数据能达到 1 万以上 TPS，100MB 大小的数据能达到 1GB 以上带宽。单节点 S3 读性能 10KB 大小的数据能达到 3 万以上 TPS，100MB 大小的数据能达到 5GB 以上带宽。

6 总结

以上，是对 Apache Flink 与 Iceberg 进入深入探索后，提供的一些让 Dell EMC ECS 对象存储作为数据湖底层存储的方案，用此方案可以将数据湖的元数据和数据都直接对接对象存储，充分利用对象存储提供的原生优势，并且无需部署额外的元数据管理服务，提供了更大的部署便利性，同时利用 Apache Iceberg 良好的 Table Format 语义，用于帮助用户组织数据。其 Catalog 所暴露出的功能能够让存储层更好地介入，使得我们可以根据对象存储的特性支持其服务。

数据湖在计算和存储上的拆分，赋予了存储服务新的生机，让对象存储从传统的冷备场景，进入到新兴的数据分析领域，并成为了其中有存在感的一环。也希望，在当下和未来，愈加高性能、高自由度的对象存储，能够为数据分析领域，能够在当下的数据热潮中，成为更多人的选择，让存储变得简单，让数据发挥更大的价值。

A 技术支持和资源

Dell EMC ECS 企业对象存储

Dell EMC ECS 概述和架构白皮书

Apache Iceberg 概述