Hadoop 迁移到阿里云 MaxCompute

技术方案

（V2.8.5）

编写人：MaxCompute 产品团队

日　期：2019.05

目录

1 概要

Hadoop 在企业构建第一代大数据平台中成为主流的技术框架，但是随着企业信息化的高速发展，在数字化、智能化的转型过程中，Hadoop 越来越复杂的技术架构和运维成本、平台的稳定性和安全性、资源的弹性伸缩能力都遇到了瓶颈，严重阻碍了客户数据业务的发展。随着云计算技术的发展和普及，越来越多的企业客户选择数据上云，在云上构建数据仓库。以云数仓、云计算为核心的企业服务架构成为新一代大数据建站的主流趋势。MaxCompute 作为云数仓、云计算的核心引擎，承载了越来越多企业客户的数据业务和数据资产，免运维、低成本、高度安全和稳定性，让客户的资源更加聚焦在业务开发上，加速业务发展。

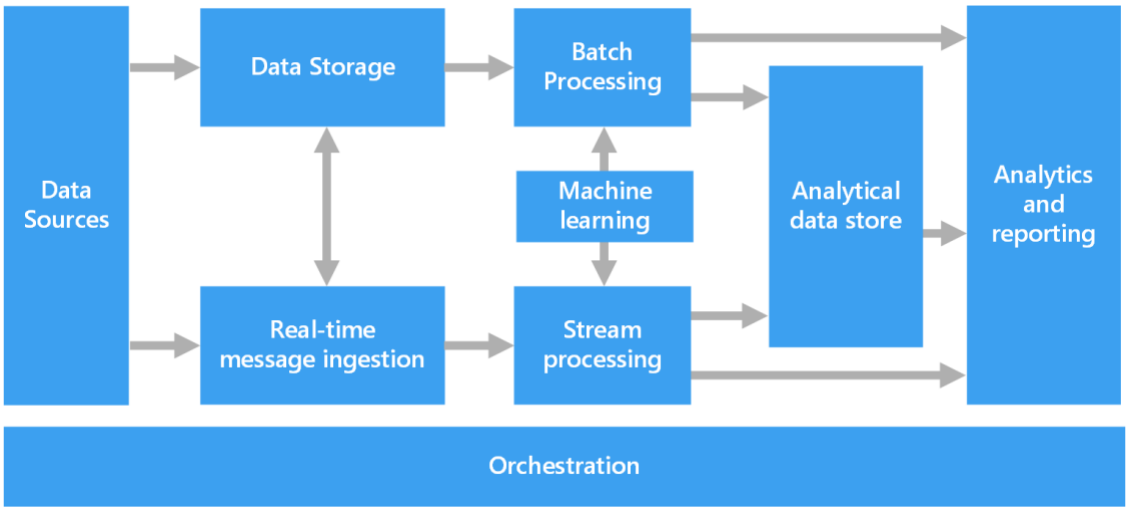
本文所描述的解决方案主要解决 Hadoop 客户如何快速、平滑的迁移到 MaxComute 大数据生态，快速完成数据和业务的迁移以及生态系统的对接。

2 阿里云大数据与开源生态对比

2.1 Hadoop 及开源生态与阿里云大数据生态对比

2.1.1 主流大数据体系架构

Hadoop 及开源生态由一系列的开源组件共同组成，很多用户基于 Hadoop 及开源生态组件构建企业数据仓库/数据湖、机器学习、实时分析、BI 报表等大数据应用。我们常见的大数据架构的逻辑组件关系如下图所示：

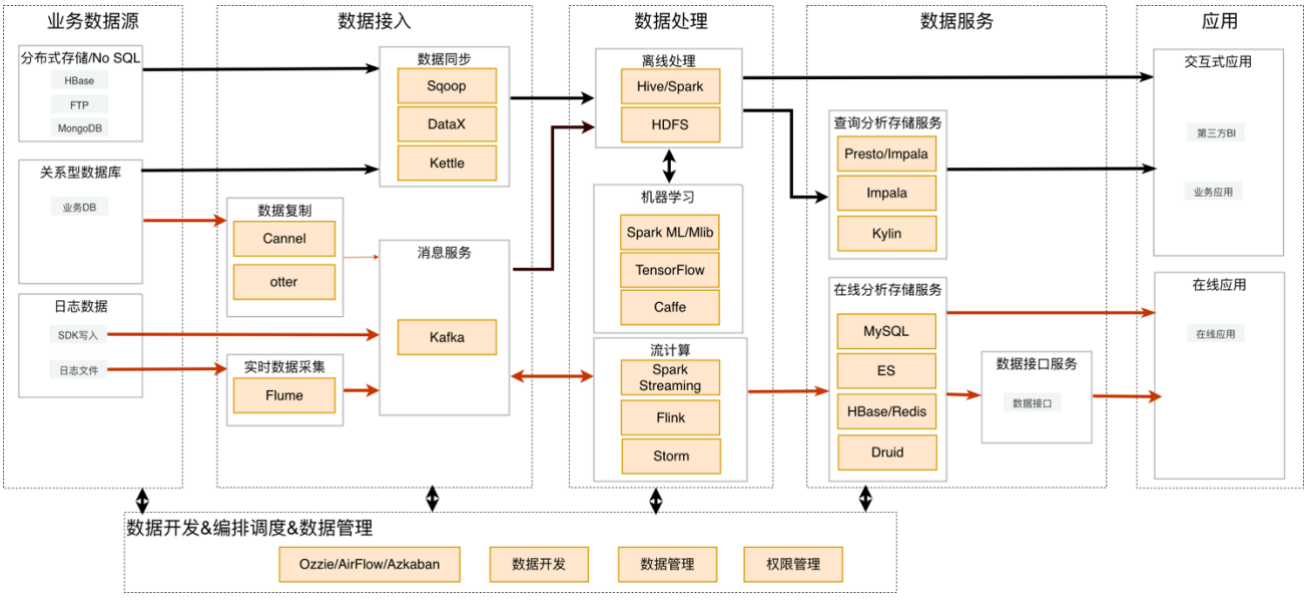


这些逻辑组件包括：

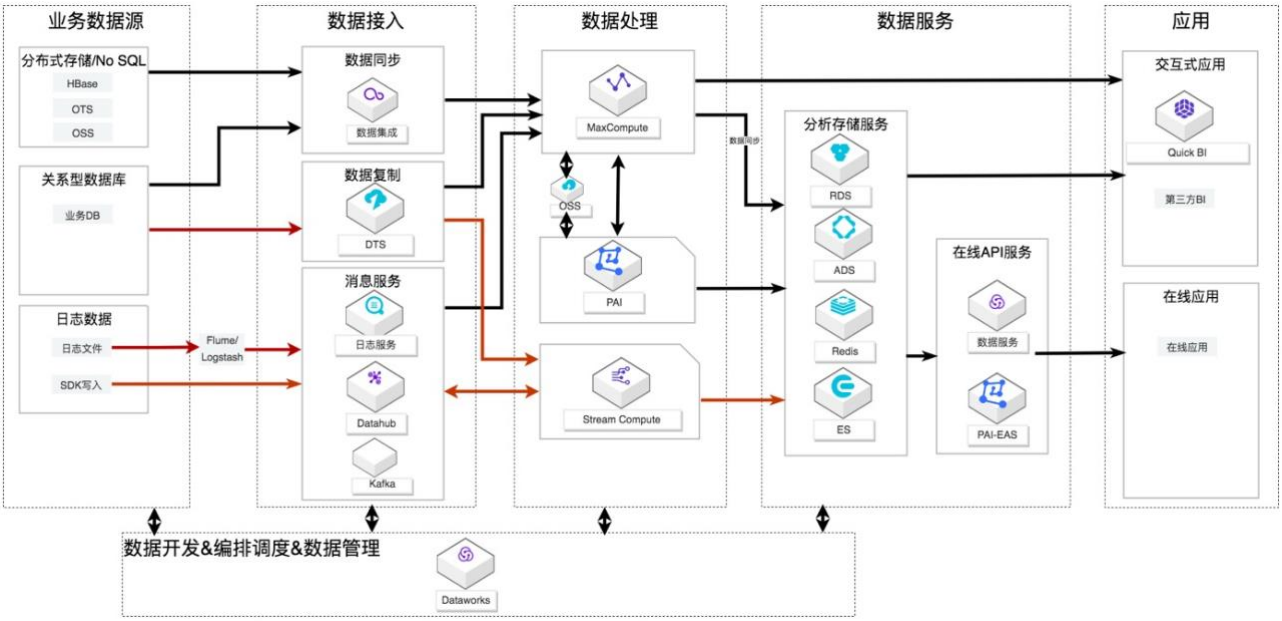
- **数据源**：数据源包括关系型数据库、日志文件、实时消息等。
- **数据存储**：面向海量数据存储的分布式文件存储服务，支持结构化数据和非结构数据数据存储，我们也常称之为数据湖。如 HDFS、对象存储服务等。
- **批处理**：由于大数据场景必须处理大规模的数据集，批处理往往需要从数据存储中读取大量数据进行长时间处理分析，并将处理后的数据写入新的数据对象供后续使用。如 Hive、MapReduce、Spark 等。

- **实时消息采集**：用于实时数据采集，可扩展、高吞吐、可靠的消息服务。如 Kafka。
- **流处理**：对实时数据进行低延迟流式计算的服务。如 Flink、Spark Streaming、Storm 等。
- **机器学习**：满足机器学习工作负载的服务。如当前流行的 Spark MLib/ML、Tensorflow 等。
- **分析型数据存储**：对数据进行处理加工后，面向应用场景，将数据以结构化的方式进行存储，以便分析工具或分析应用能够获取数据。如利用 MPP 数据仓库、Spark SQL 等支持 BI 工具访问，利用 Hbase 实现低延迟的在线服务等
- **分析与报表**：对数据进行分析和展现以获取洞察。如 BI 工具、jupyter 等。
- **数据作业编排**：将多个数据处理动作（数据移动、处理转换等）编排成为工作流并周期性地执行以实现数据处理工作的自动化。如 Apache Oozie、Sqoop 等。

2.1.2 开源大数据组件架构



2.1.3 阿里云大数据组件架构



2.1.4 阿里云大数据与 Hadoop 生态的产品映射

基于借助该大数据架构，对 Hadoop 及开源生态组件与阿里云大数据生态产品进行了对比映射（仅作为对功能定位的映射，不代表对应组件可无缝迁移），以便读者对相关服务的迁移至阿里云大数据产品服务有更好的理解。

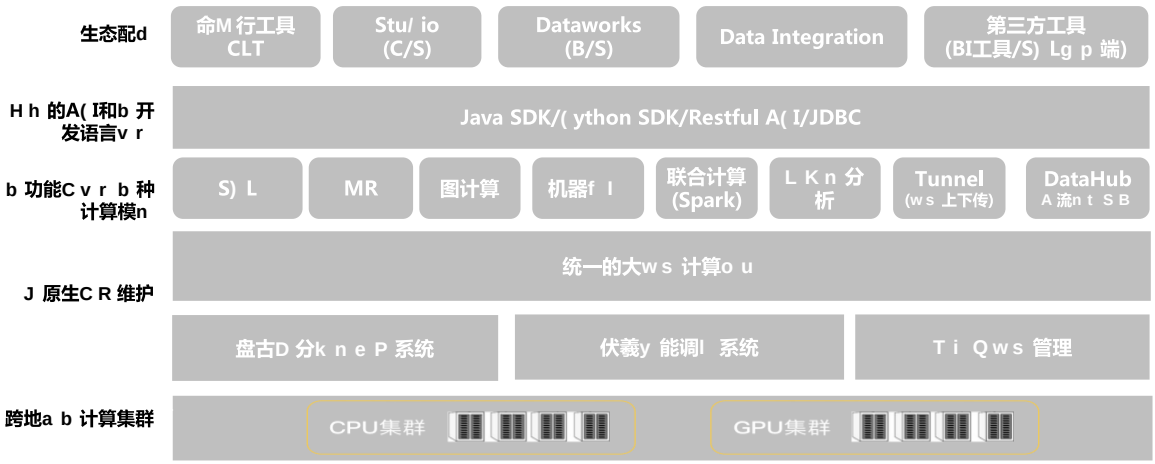
组件分类	Hadoop 开源组件	阿里云产品/产品组件
数据存储	HDFS 文件系统 对象存储	MaxCompute 存储(仅开放表数据存储) OSS 对象存储 EMR HDFS
批处理	Hadoop MapReduce Hive Spark	MaxCompute 批处理（MaxCompute MapReduce/SQL/Spark） EMR 对应组件

机器学习	Spark Mlib/ML Tensorflow	PAI 机器学习平台 MaxCompute Spark
实时消息采集	Kafka	Datahub 日志服务(LogHub 组件) 消息队列 Kafka
流处理	Spark Streaming Flink Storm	实时计算(原流计算) EMR(开源流计算组件）
分析型数据存储	数据仓库： GreenPlum/Impala/Presto/Hive NoSQL：Hbase	数据仓库：MaxCompute/ Hologres/分析型数据库 NoSQL:云数据库 Hbase 版/表格存储
分析与报表	BI 工具 Notebook	QuickBI PAI Notebook 组件 EMR Notebook 组件
数据作业编排	Oozie/Azkaban/Airflow Sqooq	Dataworks Studio 组件 Dataworks 数据集成组件

2.2 MaxCompute 特性介绍

MaxCompute 是阿里云提供高效能、低成本，完全托管的“EB 级”大数据计算服务，利用 MaxCompute 可以构建敏捷、高效的企业数据管理平台。

2.2.1 MaxComptue 的逻辑架构



2.2.2 MaxCompute 产品特性

MaxCompute 提供了云原生、多租户的服务架构，在底层大规模计算、存储资源之上预先构建好了 MaxCompute 计算服务、服务接口，提供了配套的安全管控手段和开发工具管理工具，产品开箱即用。

功能	MaxCompute 产品组件	特性介绍
数据存储	MaxCompute 表 (基于盘古分布式存储)	MaxCompute 支持大规模计算存储，适用于 TB 以上规模的存储及计算需求，最大可达 EB 级别。同一个 MaxCompute 项目支持企业从创业团队发展到独角兽的数据规模需求； 数据分布式存储，多副本冗余，数据存储对外仅开放表的操作接口，不提供文件系统访问接口 自研数据存储结构，表数据列式存储，默认高度压缩，后

		续将提供兼容 ORC 的 Ali-ORC 存储格式 支持外表，将存储在 OSS 对象存储、OTS 表格存储的数据映射为二维表 支持 Partition、Bucket 的分区、分桶存储 更底层不是 HDFS，是阿里自研的盘古文件系统，但可借助 HDFS 理解对应的表之下文件的体系结构、任务并发机制 使用时，存储与计算解耦，不需要仅仅为了存储扩大不必要的计算资源
SQL	MaxCompute SQL	TPC-DS 100% 支持，同时语法高度兼容 Hive，有 Hive 背景开发者直接上手，特别在大数据规模下性能强大。 * 完全自主开发的 compiler，语言功能开发更灵活，迭代快，语法语义检查更加灵活高效 * 基于代价的优化器，更智能，更强大，更适合复杂的查询 * 基于 LLVM 的代码生成，让执行过程更高效 * 支持复杂数据类型(array,map,struct) * 支持 Java、Python 语言的 UDF/UDAF/UDTF * 语法：Values、CTE、SEMIJOIN、FROM 倒装、Subquery Operations、Set Operations(UNION /INTERSECT /MINUS)、SELECT TRANSFORM 、User

		Defined Type、GROUPING SET(CUBE/rollup/GROUPING SET)、脚本运行模式、参数化视图 * 支持外表(外部数据源+StorageHandler 支持非结构化数据)
MapReduce	MaxCompute MR	支持 MapReduce 编程接口(提供优化增强的 MaxCompute MapReduce,也提供高度兼容 Hadoop 的 MapReduce 版本) 不暴露文件系统，输入输出都是表 通过 MaxCompute 客户端工具、Dataworks 提交作业
交互式分析	MaxCompute Lightning	MaxCompute 产品的交互式查询服务，特性如下： 兼容 PostgreSQL：兼容 PostgreSQL 协议的 JDBC/ODBC 接口，所有支持 PostgreSQL 数据库的工具或应用使用默认驱动都可以轻松地连接到 MaxCompute 项目。支持主流 BI 及 SQL 客户端工具的连接访问，如 Tableau、帆软 BI、Navicat、SQL Workbench/J 等。 显著提升的查询性能：提升了一定数据规模下的查询性能，查询结果秒级可见，支持 BI 分析、Ad-hoc、在线服务等场景。

Spark	MaxCompute Spark	MaxCompute 提供了 Spark on MaxCompute 的解决方案，使 MaxCompute 提供的兼容开源的 Spark 计算服务，让它在统一的计算资源和数据集权限体系之上，提供 Spark 计算框架，支持用户以熟悉的开发使用方式提交运行 Spark 作业。 * 支持原生多版本 Spark 作业：Spark1.x/Spark2.x 作业都可运行； * 开源系统的使用体验：Spark-submit 提交方式（暂不支持 spark-shell/spark-sql 的交互式），提供原生的 Spark WebUI 供用户查看； * 通过访问 OSS、OTS、database 等外部数据源，实现更复杂的 ETL 处理，支持对 OSS 非结构化进行处理； * 使用 Spark 面向 MaxCompute 内外部数据开展机器学习，扩展应用场景；
机器学习	PAI	MaxCompute 内建支持的上百种机器学习算法，目前 MaxCompute 的机器学习能力由 PAI 产品进行统一提供服务，同时 PAI 提供了深度学习框架、Notebook 开发环境、GPU 计算资源、模型在线部署的弹性预测服务。MaxCompute 的数据对 PAI 产品无缝集成。
存储	Pangu	阿里自研分布式存储服务，类似 HDFS。MaxCompute 对外目前只暴露表接口，不能直接访问文件系统。

资源调度	Fuxi	阿里自研的资源调度系统，类似 Yarn。
数据上传下载	Tunnel	不暴露文件系统，通过 Tunnel 进行批量数据上传下载。
流式接入	Datahub	MaxCompute 配套的流式数据接入服务，粗略地类似 kafka，能够通过简单配置归档 topic 数据到 MaxCompute 表
用户接口	CLT/SDK	统一的命令行工具和 JAVA/PYTHON SDK
开发&诊断	Dataworks/Studio/Logview	配套的数据同步、作业开发、工作流编排调度、作业运维及诊断工具。开源社区常见的 Sqoop、Kettle、Ozzie 等实现数据同步和调度。
整体	不是孤立的功能，完整的企业服务	不需要多组件集成、调优、定制，开箱即用。

3 MaxCompute 迁移场景分析

3.1 迁移基于 Hadoop 的数据湖/数据仓库业务负载

根据 MaxCompute 产品的定位和特性，您可以将基于 Hadoop 为核心的数据湖、数据仓库及周边配套工具（数据集成、数据开发、作业调度、数据治理等）业务负载迁移至 MaxCompute 及 Dataworks 的云原生大数据平台解决方案。

工作负载	Hadoop 开源生态	MaxCompute 产品组件/MaxCompute 生态工具
批处理	Hive	MaxCompute SQL

	MapReduce	MaxCompute MR
	Apache Spark	MaxCompute Spark
交互式分析	Impala Presto Hawk GreenPlum 等交互式分析	MaxCompute Lightning，提供只读的交互式查询服务
图计算	Spark GraphX	MaxCompute Spark GraphX MaxCompute Graph
流式采集	Kafka	Datahub，流式数据投递至 MaxCompute 日志服务，流式数据投递至 MaxCompute
流计算	Flink/Storm/Spark Streaming	不支持，需迁移至阿里云实时计算、EMR 流计算组件或自建流计算服务
存储	HDFS/Hive 数据存储	MaxCompute Table,MaxCompute 不提供文件服务，非结构化数据可迁移至 OSS，通过 MaxCompute 外表或 MaxCompute Spark 进行处理分析
数据集成	Sqoop Kettle	Datawroks 数据集成
编排&调度	Oozie Kettle Azkaban Airflow 等作业调度工具	Datawroks Studio

数据开发	HUE 或自研数据开发工具	Datawroks Studio

3.2 不同的网络环境及部署形态迁移

MaxCompute 提供了迁移工具，支持用户迁移来自不同网络环境及部署形态的 Hadoop 业务负载，包括：

- IDC 自建 Hadoop 集群
- 阿里云上基于 ECS 自建 Hadoop 集群
- 友商的 Hadoop 托管服务

4 Hadoop 到 MaxCompute 迁移工具介绍

4.1 MMA (MaxCompute Migration Assist)

阿里云 MaxCopute 产品面向 Hadoop 用户提供配套的搬迁工具 MaxCompute Migration Assist(简称 MMA)，利用该工具帮助用户进行迁移事前评估、利用工具加速迁移进程并降低迁移风险。

4.1.1 工具覆盖的场景：

工作负载	Hadoop 开源生态	MaxCompute 产品组件/MaxCompute 生态工具
批处理	Hive SQL	MaxCompute SQL
	MapReduce	MaxCompute MR
	Apache Spark	MaxCompute Spark

存储	HDFS/Hive 数据存储	MaxCompute Table,MaxCompute 不提供文件服务，非结构化数据可迁移至 OSS，通过 MaxCompute 外表或 MaxCompute Spark 进行处理分析
数据集成	Sqoop Kettle	Datawroks 数据集成
编排&调度	Oozie Airflow 等作业调度工具	Datawroks Studio

4.2 MMA 功能介绍

4.2.1 迁移评估分析

在迁移对 Hadoop 平台进行诊断分析，评估数据迁移规模、作业迁移改造的数量、预估迁移后的成本，从而对迁移工作进行整体评估和决策。

4.2.2 数据迁移自动化

利用迁移工具，可以对 Hive Meta 及数据进行检测扫描，自动在 MaxCompute 创建对应的 Meta，同时根据不同的网络环境，用户可选择多种数据迁移上云的方案，迁移工具提供了对应的数据迁移自动化工具，能够将 Hive 的数据自动转换并高吞吐地加载到 MaxCompute 上，支持从 TB 级到 PB 级数据的迁移上云。

4.2.3 分析任务兼容性分析及转换

利用迁移工具，可以对 Hive 作业进行兼容性分析，识别出需要修改的任务并提供针对性的兼容性修改建议。对于用户自定义逻辑的分析任务，如 UDF、MR/Spark 作业等，我们将给出一般性的改造建议供用户参考。

4.2.4 数据集成及工作流作业迁移

迁移工具支持对主流数据集成工具 Sqoop 进行作业的迁移转换，并自动创建 Dataworks 数据集成作业；迁移工具支持主流 Pipeline 工具，如 Oozie、Azkaban、Airflow 等工具的工作流及调度任务进行自动迁移转化，并自动创建为 Dataworks 工作流及调度作业。

5 迁移整体方案及流程

根据迁移工作的内容，我们提供了以下工作方法来保障迁移工作能够科学有序地开展。

整个迁移工作包含以下几个阶段：

5.1 阶段 1：调研评估&迁移方案

重点进行迁移前的评估分析，通过迁移工具对 Hadoop 平台的相关信息进行收集和诊断分析，形成迁移分析报告，供决策者评估使用。

同时，根据诊断分析报告的内容，用户可以根据自身业务现状，制定迁移方案和计划。

5.2 阶段 2：试点/全面业务迁移

在确定开展迁移工作后，需要准备 MaxCompute 相关环境，并开展数据、分析作业、工作流任务的改造和迁移工作。借助迁移工具，能够加速迁移改造的进程。

同时，需要对当前系统与 MaxCompute 环境进行业务对比验证，确定迁移的正确性。

迁移开展时，您可以选择部分试点业务迁移或全量业务进行迁移。对于规模较大的用户，建议您选择部分试点业务先行进行迁移验证，待迁移验证通过后，再扩展更大的业务范围以降低迁移风险、提高迁移质量。

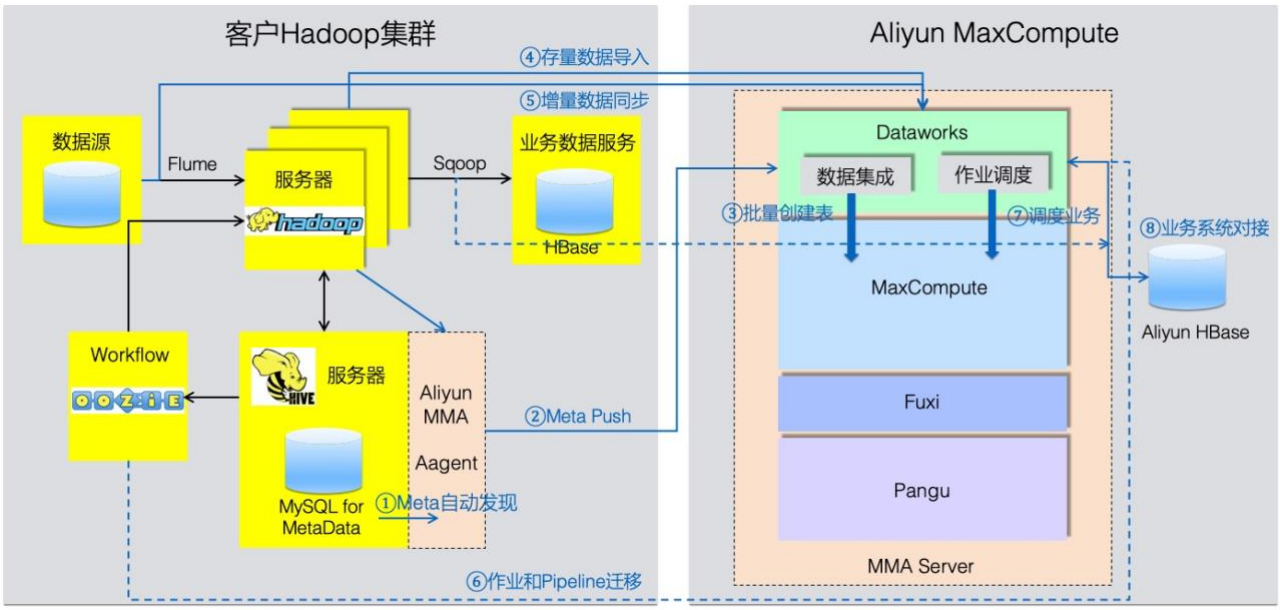
5.3 阶段 3：并行测试，割接

迁移完成后，建议基于增量数据与当前系统进行并行测试，待并行一段时间后，对并行测试结果进行对比验证，符合业务预期即可将业务全部切换至 MaxCompute 平台。

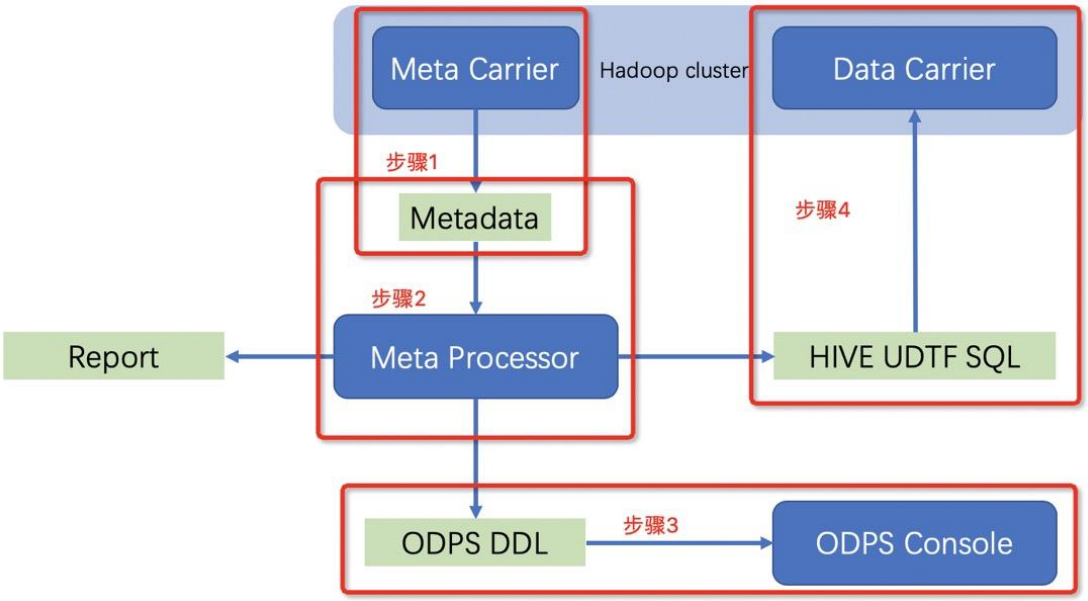
对于规模较小的系统迁移，一般迁移上线周期不超过 2 周。但更多的情况下，我们建议您根据迁移的技术评估结果、业务规模、企业管理需求等因素制定符合自身需要的迁移方案和计划。

6 迁移详细方案

6.1 MMA 迁移服务架构



6.2 MMA Agent 技术架构及原理介绍



MMA Agent 的工作流程主要分为四个步骤：

6.2.1 Metadata 抓取

- Meta carrier 连接用户的 Hive metastore 服务，抓取用户的 Hive metadata 并在指定目录生成一个目录，包含搬站所需的 metadata。用户可自行修改该目录下的文件来自定义搬站工具的一些行为。

6.2.2 MaxCompute DDL 与 Hive UDTF 生成

- 利用第一步抓取到的 metadata，生成另一个目录，包含用于创 MaxCompute 表和分区的所有 DDL 语句，还包含用于数据迁移的 Hive UDTF SQL。

6.2.3 MaxCompute 表创建

- 运行上一步生成的 MaxCompute DDL，创建 MaxCompute 的表与分区。

6.2.4 Hive 数据迁移

- 在用户 Hadoop 集群上运行步骤 2 中生成的 Hive UDTF SQL，传输数据。

6.3 迁移评估报告

6.3.1 迁移评估信息收集

6.3.1.1 使用 MMA Agent 自动采集 Hive Metadata

1. 工具运行环境要求：JDK8.0、Python3 以上版本。
2. 解压工具包：odps-data-carrier.zip，工具目录结构如下：

```
odps-data-carrier/
├── bin
│   ├── meta-carrier         用于获取Hive metadata的工具
│   ├── meta-processor         用于生成ODPS DDL及Hive UDTF SQL的工具
│   ├── odps_ddl_runner.py       用于运行ODPS DDL的工具
│   ├── hive_udtf_sql_runner.py      用于运行Hive UDTF SQL的工具
│   └── sql-checker          用于检查用户Hive SQL是否能直接在ODPS上运行的工具
├── libs
│   ├── data-transfer-hive-udtf-1.0-SNAPSHOT-jar-with-dependencies.jar
│   ├── meta-carrier-1.0-SNAPSHOT-jar-with-dependencies.jar
│   ├── meta-processor-1.0-SNAPSHOT-jar-with-dependencies.jar
│   └── odps-sql-migration-tool-wrapper-1.0-SNAPSHOT.jar
├── odps_config.ini
└── res
    ├── console
    └── style.css
```

其中，bin 目录下是迁移工具所需的可执行文件，libs 目录下是工具所依赖的库，res 目录下是工具所需的其他依赖，如 odpscmd 等。

3. 获取 Hive metadata

```
Usage:
    meta-carrier -u <uri> -o <output dir> [-h] [-d <database>] [-t <table>]
        -h,--help                       打印help信息
        -o,--output-dir <output-dir>    必填，指定一个输出的目录
        -u,--uri <uri>                  必填，指定hive metastore service的thrift地址
        -d,--database <database>        可选，指定一个database
        -t,--table <table>              可选，指定一个table

Example:
    导出全部metadata:
    sh meta-carrier -u thrift://127.0.0.1:9083 -o meta

    导出某个database的metadata:
    sh meta-carrier -u thrift://127.0.0.1:9083 -o meta -d test_db

    导出某张表的metadata:
    sh meta-carrier -u thrift://127.0.0.1:9083 -o meta -d test_db -t test_tbl
```

4. 结果输出

```
Output:
    A directory, its structure is as follows:
    [output directory]
    |______global.json
    |______[database name]
           |______[database name].json
           |______table_meta
           |      |______[table name].json
           |______partition_meta
                  |______[table name].json
```

说明：①global.json 是一个全局的配置文件，包含了整个迁移过程中的一些配置，例如将要使用的 MaxCompute 的版本，是否打开 hive compatible 开关等。②每一个 database 会有一个独立的目录，下面会有每一个表的 table meta，以表名为文件名的 json 文件，如果是分区表还会有 partition meta，同样是以表名为文件名的 json 文件。

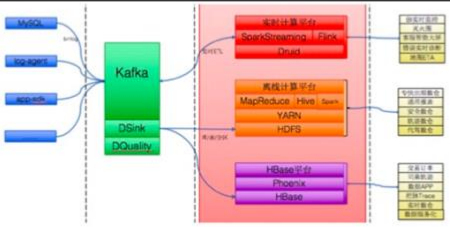
5. 基于 Kerberos 做身份认证的 meta 连接参数配置

```
# sh odps-data-carrier/bin/meta-carrier -u thrift://xxx.xxx.xxx:9083 -o meta --principal
hive/xxx.xxx.xxx@xxx.xxx --system java.security.krb5.conf=/etc/krb5.conf
java.security.auth.login.config=/root/gss-jaas.conf
javax.security.auth.useSubjectCredsOnly=false
```

6.3.1.2 客户信息收集模板

- 客户需要登录到 Dataworks 填写“Hadoop 搬站信息”表单，需提供如下信息：
- Hadoop 类型和版本（CDH 自建、CDH 云上自建、Hadoop IDC 自建、Hadoop 云上自建、云上托管 EMR）
- 集群规模（服务器台数）

- 网络环境（私有网络、经典网络、VPC 专）
- 有无专线
- 常用组件（Hive、Spark、Storm、HBase、Flink、Kafa、Impala、Sqoop、Kylin、Flume）
- 机器配置（CPU 核数、内存大小）
- 数据量及存储类型
- 作业量及作业类型（SQL 脚本上传）
- 调度系统及周期（Pipeline 配置上传）
- 已有数据应用（如血缘、监控、质量等）
- 上层应用系统（如帆软 BI、推荐系统等）
- 期望时间
- 成本要求

「**」Hadoop搬站调查表					
客户名称：		**对接部门：**		**负责人：**	
搬站背景(存在问题)**：**					
Hadoop集群现状					
Hadoop类型：	○ CDH IDC自建 ○ CDH云上自建 ○ Hadoop IDC自建 ○ Hadoop云上自建 ○ 云上托管EMR				
集群规模：	（台）	**网络环境：**	☑ 私有网络 ☑ 经典网络 ☑ VPC	**专线连通：**	☐ 专线打通 ☐ 专线未通
常用组件：	☑ Hive ☑ Spark ☑ Storm ☑ HBase ☐ Flink ☑ Kafka ☑ Impala ☑ Sqoop ☐ Kylin ☐ Flume			**机器配置：**	**（核）**（GB）
现有数据架构图					
					
Hadoop集群表/作业情况					
/***数据源/数据增量（存量）/表数量/作业数量/作业类型/应用系统等***/ （1）数据量及数据存储类型： （2）作业量及类型： （3）表数量(数据分层)： （4）调度系统及周期： （5）已有数据应用：如血缘、监控、质量等 （6）上层应用系统：帆软BI、推荐系统等					
搬站期望达到的效果					
/***搬站周期/成本降低效果等***/ （1）期望搬站周期： （2）成本对比方面：					

6.3.1.3 检查网络连通性

使用客户端工具 network-measurement-tool 可以检查 Hadoop 集群与 MaxCompute 各个 Region 的网络连通质量，以及 download/upload 的性能。

- 工具使用方法

```
Usage:
  network-measure-tool --mode FIND|TEST

Options:
  --mode <mode>                          FIND (find available endpoints)
                                           or TEST (test performance of a
                                           single endpoint)
  --endpoint <endpoint>                  ODPS endpoint, required in TEST
                                           mode
  -h,--help                              Print help information
  -p,--access-key <access-key>           ODPS access key, required in
                                           TEST mode
    --project <project>                    ODPS project name, required in
                                           TEST mode
  -t,--num-thread <num-thread>           Number of thread
    --tunnel-endpoint <tunnel-endpoint>    ODPS tunnel endpoint, optional
  -u,--access-id <access-id>             ODPS access id, required in TEST
                                           mode
```

- Example

```
Example:
  查找可用endpoint:
  odps-data-carrier/bin/network-measurement-tool --mode find

Output:
  各个endpoint的连接情况（只输出可以连接的）：
------------------------------------------------------------------------
  ENDPOINT: EXTERNAL-BEIJING: http://service.cn.maxcompute.aliyun.com/api
  AVAILABILITY: true
  ELAPSED TIME (ms): 4
------------------------------------------------------------------------
  ENDPOINT: EXTERNAL-HANGZHOU: http://service.cn.maxcompute.aliyun.com/api
  AVAILABILITY: true
  ELAPSED TIME (ms): 5

Example:
  测试某个endpoint的读写性能:
  odps-data-carrier/bin/network-measurement-tool --mode test \
  --endpoint <endpoint to test> \
  -u <access id> -p <access key> \
  --project=<project name> --num-thread <number of thread>
```

- 输出结果

```
Output:
  性能测试报告
[INFO ] 2019-05-20 17:17:21.664 [main] PerformanceTester - Create table
ODPS_NETWORK_MEASUREMENT_TOOL_TEST_TBL
[INFO ] 2019-05-20 17:17:24.718 [main] PerformanceTester - Start testing upload performance
 100%
[================================================================================]
25/25 (0:00:07 / 0:00:00)
[INFO ] 2019-05-20 17:17:31.974 [main] PerformanceTester - Done testing upload performance
[INFO ] 2019-05-20 17:17:31.974 [main] PerformanceTester - Start testing download performance
 100%
[================================================================================]
25/25 (0:00:03 / 0:00:00)
[INFO ] 2019-05-20 17:17:35.049 [main] PerformanceTester - Done testing download performance
[INFO ] 2019-05-20 17:17:35.049 [main] PerformanceTester - Delete table
ODPS_NETWORK_MEASUREMENT_TOOL_TEST_TBL
------------------------------------------------------------------------
  ENDPOINT: null-null: <endpoint>
  UPLOAD PERFORMANCE (MB/s): 15.52
  DOWNLOAD PERFORMANCE (MB/s): 58.82
```

6.3.2 资源评估

- 评估系统会根据客户的集群规模、服务器配置、数据量和作业量等信息，估算出在 MaxCompute 相应的资源购买规格建议：1）计费模式：预付费/后付费；2）规格：CU 数和存储规格等。

6.3.3 数据、作业和 Pipeline 迁移评估

6.3.3.1 使用 MMA Agent 获得评估报告：

- 报告中将搬站风险分为两档，高风险(**HIGH RISK**)与中等风险(**MODERATE RISK**)。高风险意味着必须人工介入，例如出现了表名冲突，ODPS 完全不支持的类型等问题。中等风险意味着迁移过程中可以自动处理，但是需要告知用户的潜在风险，例如 Hive 数据类型到 ODPS 数据类型会带来的精度损失等问题。以下是一个报告的例子：

ODPS compatibility report

SUMMARY

HIVE DATABASE NAME	# HIGH RISK	# MODERATE RISK
test_types	0	6

TEST_TYPES

HIVE TABLE NAME	# HIGH RISK	# MODERATE RISK
dummy	0	1
string_types	0	1
test_partition	0	2
date_and_time_types	0	1
numeric_types	0	1

TEST_TYPES.DUMMY

RISK LEVEL	DESCRIPTION
MODERATE	Incompatible type 'STRING', can be transform to 'STRING' automatically, but may cause problem. Reason: String in ODPS cannot exceed 8MB

TEST_TYPES.STRING_TYPES

RISK LEVEL	DESCRIPTION
MODERATE	Incompatible type 'STRING', can be transform to 'STRING' automatically, but may cause problem. Reason: String in ODPS cannot exceed 8MB

【说明】：报告中对于 String 类型的 8M 限制的警告：不会截断该字段，但整个 SQL（表或分区）的写入都会失败，因为 sql-checker 就会报错，不会走到 commit。

6.3.3.2 从 Dataworks 获得评估报告：

- 查看节点列表 :Dataworks 会将客户上传的 Hive SQL 和调度配置 ,自动转换成 ODPS SQL，并且映射成 Dataworks 的工作流类型。

节点列表

工作流名称	工作流类型	节点名称	节点类型	代码路径
testWorkflow	自动调度	testNode	ODPS_SQL	./src/workflows/testWorkflow/nodes/testNode/testNode
test_flow_01	自动调度	testNode	ODPS_SQL	./src/workflows/test_flow_01/nodes/testNode/testNode
test_types_meta_1	手动业务流程	dummy	ODPS_SQL	./src/workflows/test_types_meta_1/nodes/dummy/dummy
test_types_meta_1	手动业务流程	test_partition	ODPS_SQL	./src/workflows/test_types_meta_1/nodes/test_partition/test_partition
test_types_meta_1	手动业务流程	test	ODPS_SQL	./src/workflows/test_types_meta_1/nodes/test/test
test_types_meta_1	手动业务流程	string_types	ODPS_SQL	./src/workflows/test_types_meta_1/nodes/string_types/string_types
test_types_meta_1	手动业务流程	date_and_time_types	ODPS_SQL	./src/workflows/test_types_meta_1/nodes/date_and_time_types/date_and_time_types
test_types_meta_1	手动业务流程	complex_types	ODPS_SQL	./src/workflows/test_types_meta_1/nodes/complex_types/complex_types
test_types_meta_1	手动业务流程	misc_types	ODPS_SQL	./src/workflows/test_types_meta_1/nodes/misc_types/misc_types
test_types_meta_1	手动业务流程	complex_types_2	ODPS_SQL	./src/workflows/test_types_meta_1/nodes/complex_types_2/complex_types_2
test_types_meta_1	手动业务流程	numeric_types	ODPS_SQL	./src/workflows/test_types_meta_1/nodes/numeric_types/numeric_types
oozie_hive-wf	自动调度	start	VIRTUAL	./src/workflows/oozie_hive-wf/nodes/start/start
oozie_hive-wf	自动调度	hive-node	ODPS_SQL	./src/workflows/oozie_hive-wf/nodes/hive-node/script.q
oozie_hive-wf	自动调度	sqoop2_mysql2hive	DI	./src/workflows/oozie_hive-wf/nodes/sqoop2_mysql2hive/sqoop2_mysql2hive
oozie_hive-wf	自动调度	end	VIRTUAL	./src/workflows/oozie_hive-wf/nodes/end/end
oozie_hive2-wf	自动调度	start	VIRTUAL	./src/workflows/oozie_hive2-wf/nodes/start/start
oozie_hive2-wf	自动调度	hive2-node	ODPS_SQL	./src/workflows/oozie_hive2-wf/nodes/hive2-node/script.q
oozie_hive2-wf	自动调度	end	VIRTUAL	./src/workflows/oozie_hive2-wf/nodes/end/end

- 查看报告明细：列表中会给出 Hive meta、SQL、DI、OOZIE 作业的转换、迁移风险等级，及异常的详细信息。

报告明细

#	名称	类型	风险等级	异常信息	路径
1	Database: test_types / Table: complex_types	HIVE_META	WEEK_WARNINGS		
2	Database: test_types / Table: complex_types_2	HIVE_META	WEEK_WARNINGS		
3	Database: test_types / Table: test_partition	HIVE_META	STRONG_WARNINGS		
4	Database: test_types / Table: misc_types	HIVE_META	WEEK_WARNINGS		
5	Database: test_types / Table: dummy	HIVE_META	STRONG_WARNINGS		
6	Database: test_types / Table: date_and_time_types	HIVE_META	STRONG_WARNINGS		
7	Database: test_types / Table: test	HIVE_META	STRONG_WARNINGS		
8	Database: test_types / Table: numeric_types	HIVE_META	STRONG_WARNINGS		
9	Database: test_types / Table: string_types	HIVE_META	STRONG_WARNINGS		
10	ods_coord	OOZIE	ERROR	variable [wname] cannot be resolved	/Users/sam.liux/Workspace/code/dataworks-migration/core/src/test/resources/project/./src/assets/oozie/ods_coord
11	Workflow: oozie_hive-wf / Node: start	OOZIE	OK		/Users/sam.liux/Workspace/code/dataworks-migration/core/src/test/resources/project/./src/assets/oozie/oozie_hive-wf
12	Workflow: oozie_hive-wf / Node: hive-node	OOZIE	OK		/Users/sam.liux/Workspace/code/dataworks-migration/core/src/test/resources/project/./src/assets/oozie/oozie_hive-wf

6.4 Meta 和数据迁移

6.4.1 环境准备

6.4.1.1工具运行环境

- JDK 1.8
- Python 3.x
- Hive Client
- 能访问 Hive Server 的机器
- 网络连接 MaxCompute

https://help.aliyun.com/document_detail/57195.html?spm=a2c4g.11174283.6.579.33513a79ZnTEsX

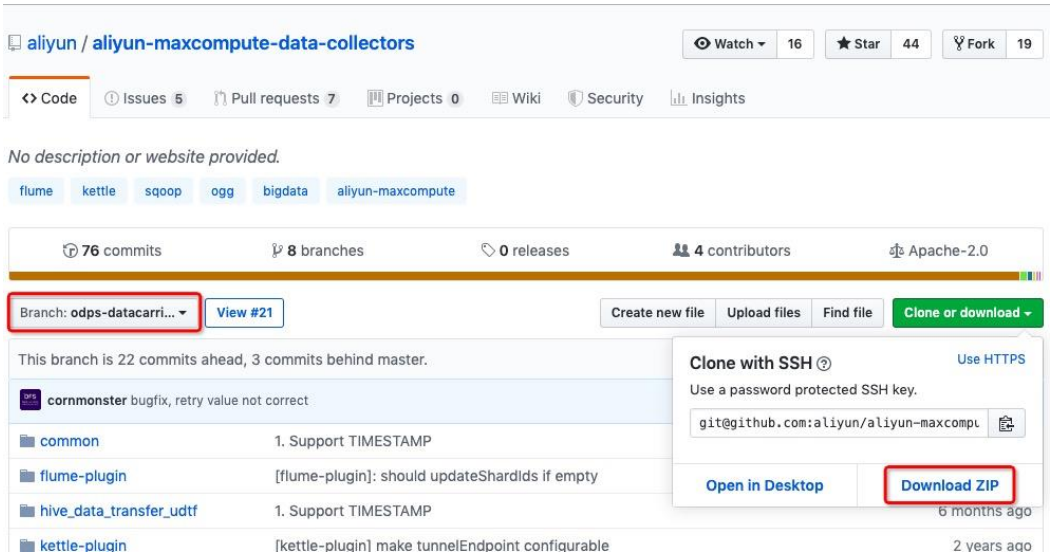
6.4.1.2 下载和编译工具包

- MMA 官方文档地址：

https://help.aliyun.com/document_detail/121023.htm?spm=a2o8d.corp_prod_req_list.0.0.16d06b88pXRwqH

- 下载源码：切换到 odps-datacarrier-develop 分支，

https://github.com/aliyun/aliyun-maxcompute-data-collectors?spm=a2c4g.11186623.2.8.422c4c07MdjlpQ



- 解压下载的 aliyun-maxcompute-data-collectors-odps-datacarrier-develop.zip 文件

- 在控制台运行 odps-data-carrier 目录下的 build.py 文件，编译生成 MMA 工具

```
usage: build.py [-h] --hive_version HIVE_VERSION
                [--excluded_tools EXCLUDED_TOOLS [EXCLUDED_TOOLS ...]]

odps-data-carrier builder

optional arguments:
  -h, --help            show this help message and exit
  --hive_version HIVE_VERSION
                        hive-version
  --excluded_tools EXCLUDED_TOOLS [EXCLUDED_TOOLS ...]
                        tools to be excluded from the package, available
                        values are: meta-carrier meta-processor odps-ddl-
                        runner hive-udtf-sql-runner network-measurement-tool
                        sql-checker
```

- 编译环境要求：JDK 1.8+、 Apache Maven 3.x、Python 3.x

6.4.2 方案 A：通过 MMA Agent 迁移 Meta 和数据

1. 安装 MMA Agent 并使用 meta-carrier 获取 hive metadata 参见 6.3.1.1。
2. 客户需要预先开通 MaxCompute 服务，并创建好 project。
3. 根据 meta-carrier 抓取到 metadata 生成 global.json，同时用户可以编辑这个 json 来自定义表、字段的生成规则，可编辑的部分如下：

```
global.json:
{
   "datasourceType" : "HIVE", // 目前仅支持HIVE
   "odpsVersion" : "1.0", // 默认ODPS版本为1.0
   "hiveCompatible" : false // 在ODPS 2.0下，是否打开hive compatible开关
 }

[database name].json:
 {
   "databaseName" : xxx, // Hive数据库名
   "odpsProjectName" : xxx // 对应的ODPS项目名，默认与Hive数据库名相同
 }

[table name].json:
 {
    "tableName" : "xxx", // Hive表名
    "odpsTableName" : "xxx", // 对应的ODPS表名，默认与Hive表名相同
    "lifeCycle" : 10, // ODPS表的life cycle，默认为空，即不启用life cycle
    "comment" : "xxx", // ODPS表的comment，默认为空
    "ifNotExists" : true, // 创建ODPS表时是否加if not exists，默认不加
    "columns" : [
        {
            "name" : "column_1", // Hive列名
            "odpsColumnName" : "odps_column_1", // 对应的ODPS列名，默认与Hive列名相同
            "type" : "bigint", // ODPS列的类型，用户暂时不可自行修改
            "comment" : "xxx" // ODPS列的comment
        },
        ...
    ],
    "partitionColumns" : [
        {
            "name" : "column_1", // Hive分区列名
            "odpsColumnName" : "odps_column_1", // 对应的ODPS分区列名，默认与Hive分区列名相同
            "type" : "bigint", // ODPS分区列类型
            "comment" : "xxx" // ODPS分区列的comment
        },
        ...
    ]
 }
```

【注意】：配置文件中默认 hiveCompatible 的设置是 false，如果需要把 hive 上的 udf 的 jar 直接上传到 odps 上，需要打开 hive 兼容。

4. 编辑好 metadata 之后，便可以开始生成 ODPS DDL 和 Hive UDTF SQL 了，用法如下：

```
Usage:
  ./meta-processor -i <metadata directory> -o <output directory>

Options:
   -h,--help                      Print help information
   -i,--input-dir <input-dir>     Directory generated by meta carrier
   -o,--output-dir <output-dir>   Output directory generated by meta
                                   processor

Example:
  ./meta-processor -i meta -o output

Output:
  A directory, its structure is as follows:
   [output directory]
   |______Report.html
   |______[database name]
          |______odps_ddl
          |      |______tables
          |      |      |______[table name].sql
          |      |______partitions
          |             |______[table name].sql
          |______hive_udtf_sql
                 |______single_partition
                 |      |______[table name].sql
                 |______multi_partition
                        |______[table name].sql
```

5. ODPS DDL 创建好以后，odps_ddl_runner.py 将会遍历 meta-processor 生成的目录，调用 odpscmd 自动创建 ODPS 表与分区。

```
Usage:
  python3 odps_ddl_runner.py [-h] --input INPUT [--odpscmd ODPSCMD]

Options:
  --input <input-dir>     Directory generated by meta processor
    --odpscmd <path to odpscmd executable> Optional, user can use
                their installed odpscmd to execute odps ddl

Example:
  python3 odps_ddl_runner.py --input input --odpscmd /opt/console/bin/odpscmd

Output:
  执行过程中的日志，如:
    INFO: executing 'console/bin/odpscmd -f output/test_types/odps_ddl/tables/dummy.sql'
  DEBUG: stdout: b''
  DEBUG: stderr: b'OK\nID = 20190505071530558gyang67a\nOK'
  DEBUG: returncode: 0
```

【注意】：odps_ddl_runner.py 需要依赖 odpscmd，因此在执行前，需要配置 odpscmd 的 config.ini 文件，配置方法请参见文档：https://help.aliyun.com/document_detail/27804.html?spm=a2c4g.11186623.2.16.2fbaa95emqdrea#concept-qbk-1kv-tdb

6. 表和分区创建完成以后，hive_udtf_sql_runner.py 将会遍历 meta-processor 生成的目录，调用 hive client 运行 hive udtf sql，从而将数据从 hive 上传至 MaxCompute。

```
Usage:
  python3 hive_udtf_sql_runner.py [-h] [--input_all INPUT_ALL]
                                   [--input_single_file INPUT_SINGLE_FILE]
                                   [--settings SETTINGS]

Options:
  --input_all <input-dir>          自动运行<input-dir>下的所有hive sql文件，将会自动迁移所有的数据
    --input_single_file              运行指定的hive sql文件，只会迁移指定的表或分区
    --settings                       额外的Hadoop/Hive配置，默认用odps-data-carrier/extra_setti

Example:
    执行meta-processor生成的所有hive sql:
  python3 hive_udtf_sql_runner.py --input_all processed

    迁移单个表或分区:
    python3 hive_udtf_sql_runner.py --input_single_file /path/to/hive_sql.sql

Output:
  执行过程中的日志
```

7. 编程接口：Maven xml

```
<dependency>
  <groupId>com.aliyun.odps</groupId>
  <artifactId>data-carrier-commons</artifactId>
  <version>1.0-SNAPSHOT</version>
</dependency>

<dependency>
  <groupId>com.aliyun.odps</groupId>
  <artifactId>meta-carrier</artifactId>
  <version>1.0-SNAPSHOT</version>
  <classifier>jar-with-dependencies</classifier>
</dependency>

<dependency>
  <groupId>com.aliyun.odps</groupId>
  <artifactId>meta-processor</artifactId>
  <version>1.0-SNAPSHOT</version>
  <classifier>jar-with-dependencies</classifier>
</dependency>
```

8. 对于 hive 集群上作业提交队列的说明：

我们工具在创建 hive 作业迁移数据的时候，会把作业提交到 default queue，如同客户的 hive 集群上没有 default queue，就需要客户指定队列名称，方法如下：

① 使用 hadoop queue -showacls | grep SUBMIT 命令查看 queue name

```
[09:27:14 hadoop@fhap02:~]
$ hadoop queue -showacls | grep SUBMIT
DEPRECATED: Use of this script to execute mapred command is deprecated.
Instead use the mapred command for it.

root.hadoop.plarch  ADMINISTER_QUEUE,SUBMIT_APPLICATIONS

[10:03:48 hadoop@fhap02:~]
$
```

② 修改 odps-data-carrier/extra-settings.ini，添加 mapreduce.job.queuename=root.hadoop.plarch

6.4.3 方案 B：使用 Dataworks 服务迁移 Meta 和数据

6.4.3.1 操作步骤

1. 客户需要预先开通 MaxCompute 服务，并创建好 project。
2. 安装 MMA Agent 并使用 meta-carrier 获取 hive metadata，参见 6.3.1.1。
3. 使用 meta-processor 生成 ODPS DDL 和 Hive UDTF SQL，参见 6.4.1 的第 3、4 步。
4. 根据模板生成 DataWorks 项目描述文档，打包为：dataworks_project.tgz 上传到 Dataworks。【注意】：一期仅支持：1）打包文件手动上传；2）支持 OOIZE 调度引擎的配置模板和 Dataworks 工作流配置模板。
5. 上传完成后，Dataworks 服务会根据 ODPS DDL 批量生成 MaxCompute 的 table。
6. MaxCompute 的表创建完成后，Dataworks 服务会自动拉起 DataX 的数据同步任务，完成批量数据迁移。

6.4.3.2 Dataworks 项目描述文档的目录结构及说明：

```
.
├── project.xml # 项目描述文件，文件名固定为project.xml
└── src
    ├── assets # 用户已有的大数据平台任务
    │   ├── hive_meta # hive meta的数据目录，由MMA Agent工具生成
    │   │   ├── meta # 使用meta-carrier生成hive的metadata
    │   │   │   ├── global.json
    │   │   │   └── [database name]
    │   │   │       ├── [database name].json
    │   │   │       ├── table_meta
    │   │   │       │   └── [table name].json
    │   │   │       └── partition_meta
    │   │   │           └──[table name].json
    │   │   └── result # 使用meta-processor生成ODPS DDL和Hive UDTF SQL
    │   │       ├── Report.html
    │   │       └── [database name]
    │   │           ├── odps_ddl
    │   │           │   ├── tables
    │   │           │   │   └──[table name].sql
    │   │           │   └── partitions
    │   │           │       └──[table name].sql
    │   │           └── hive_udtf_sql
    │   │               ├── single_partition
    │   │               │   └──[table name].sql
    │   │               └── multi_partition
    │   │                   └──[table name].sql
    │   └── oozie # 此目录下存放oozie的调度任务
    │       ├── hive2_sample # oozie的某一个调度任务目录
    │       │   ├── README
    │       │   ├── job.properties
    │       │   ├── job.properties.security
    │       │   ├── script.q
    │       │   ├── workflow.xml
    │       │   └── workflow.xml.security
    │       ├── hive_sample # oozie的某一个调度任务目录
    │       │   ├── README
    │       │   ├── job.properties
    │       │   ├── script.q
    │       │   ├── workflow.xml
    │       │   └── workflow.xml.security
    └── workflows # DataWorks工作流描述目录，目录名固定为workflows
        ├── oozie_hive-wf # 工作流名称（自定义）
        │   ├── nodes # 目录名称固定为nodes
        │   │   ├── hive-node # 节点名称（自定义）
        │   │   │   └── script.q # 节点代码文件（自定义）
        │   │   └── sqoop2_mysql2hive # 节点名称（自定义）
        │   │       └── sqoop2_mysql2hive # 节点代码文件（自定义）
        │   └── workflow.xml # 工作流描述文件，文件名固定workflow.xml
        └── test_types_meta_1 # 目录名称为工作流名称（自定义）
            ├── nodes # 目录名称固定为nodes
            │   ├── complex_types  # 目录名为节点名称（自定义）
            │   │   └── complex_types # 节点代码文件（自定义）
            │   ├── complex_types_2 # 目录名为节点名称（自定义）
            │   │   └── complex_types_2 # 节点代码文件（自定义）
            │   └── test_partition # 目录名为节点名（自定义）
            │       └── test_partition # 节点代码文件（自定义）
            └── workflow.xml # 工作流描述文件，文件名固定为workflow.xml
```

6.4.3.3 项目描述文件说明/project.xml：

```
<Project name="zxy_7051645" tenantId="277744729020672" owner="1736629400048545">
  <Workflows>
    <Workflow ref="./src/workflows/test_types_meta_1/workflow.xml"/>
    <Workflow ref="./src/workflows/oozie_hive-wf/workflow.xml"/>
  </Workflows>
</Project>
```

- **tenantId:** 用户在 dataworks 上的租户 ID；
- **name:** 用户事先在 dataworks 上创建好的项目空间名称；owner: 用户的阿里云账号 ID。

6.4.3.4 工作流描述文件说明/workflow.xml：

```
<Workflow name="oozie_hive-wf" scheduled="true"> <!-- scheduled标识是否要周期调度 -->
  <Nodes>
    <!-- ref指向项目目录的代码文件路径，type的取值参见后文说明 -->
    <Node name="start" type="VIRTUAL">
      <Outputs> <!-- 节点的输出标识，用于dataworks在节点之间挂接依赖 -->
        <Output data="start.out"/>
      </Outputs>
    </Node>
    <Node name="hive-node" type="ODPS_SQL">
      <code><![CDATA[
DROP TABLE IF EXISTS test;
CREATE EXTERNAL TABLE test (a INT) STORED AS TEXTFILE LOCATION '/user/test_user/examples/ir
INSERT OVERWRITE DIRECTORY '/user/test_user/examples/output-data/hive' SELECT * FROM test;
]]></code>
      <Inputs>
        <Input data="start.out"/>
      </Inputs>
      <Outputs>
        <Output data="hive-node.out"/>
      </Outputs>
    </Node>
    <Node name="sqoop2_mysql2hive" type="DI">
      <code><![CDATA[{}]]></code>
      <Outputs>
        <Output data="sqoop2_mysql2hive.out"/>
      </Outputs>
    </Node>
    <Node name="end" type="VIRTUAL">
      <Inputs>
        <Input data="hive-node.out"/>
      </Inputs>
      <Outputs>
        <Output data="end.out"/>
      </Outputs>
    </Node>
  </Nodes>
</Workflow>
```

· **节点类型说明：**

type	DataWorks对应概念
DI	数据集成类型的节点
VIRTUAL	虚拟节点
ODPS_SQL	Max Compute SQL类型节点

· **概念说明：**

概念	DataWorks概念
<Project>	项目空间
<Workflow>	业务流程，包含手动业务流程和业务流程（周期调度的）
<Node>	任务节点，一个业务流程包含多个任务节点，任务节点之间可以有依赖关系
<Input>	节点输入，<Input>就是一个字符串，一个节点可以有多个<Input>，DataWorks通过Input确定本节点的上游节点是哪些，节点的Input是上游节点的Output
<Output>	节点输出，<Output>同<Input>，一个节点可以有多个Output

6.5 作业迁移

6.5.1 Hive SQL -> MaxCompute SQL 自动转换

6.5.1.1 使用 sql-checker 做语法检查

Agent 提供 SQL 语法检查的工具，可以帮助开发者自助的对 Hive SQL 做语法检查，并且对于不兼容的语法，sql-checker 会输出所有的语法和语义问题，并给出修改建议。

```
Usage:
  sql-checker -i <metadata dir> -p <default odps project> --sql <sql to check> [--settings

Options:
  -h,--help                     Print help information
  -i,--input-dir <input-dir>    Directory generated by meta carrier, which
                                  contains all the metadata
  -p,--project <project>        Default odps project
    --settings <settings>         Odps settings
    --sql <sql>                   Sql statements to check

Example:
  sql-checker -i meta -p odps_test -sql "select * from tbl_1"
                --settings odps.sql.type.system.odps2=true
Output:
  sql是否有语法或语义错误，错误的详细信息以及修改建议，如:
  Issues:
    0.
    Compatibility: ERROR
    Description: [line 1, col 15] table ODPS_DATA_CARRIER.string_types cannot be resolved
    Suggestion: null
```

6.5.1.2 在 Dataworks 上做检查和转换

1. 根据模板上传 Dataworks 项目描述文档，参见 6.4.2。
2. Dataworks 会自动批量将 Hive SQL 转换成 ODPS SQL，对于不能转换的 SQL，系统会给出错误提示，需要客户手动修改。

6.5.2 UDF、MR 迁移

支持相同逻辑的 UDF、MR 输入、输出参数的映射转换，但 UDF 和 MR 内部逻辑需要客户自己维护。【注意】：不支持在 UDF、MR 中直接访问文件系统、网络访问、外部数据源连接。

6.5.3 Spark 作业迁移

1. 【作业无需访问 MaxCompute 表和 OSS】用户 jar 包可直接运行，参照《MaxCompute Spark 开发指南》第二节准备开发环境和修改配置。注意，对于 spark 或 hadoop 的依赖必须设成 provided。

2. 【作业需要访问 MaxCompute 表】参考《MaxCompute Spark 开发指南》第三节编译 datasource 并安装到本地 maven 仓库，在 pom 中添加依赖后重新打包即可。
3. 【作业需要访问 OSS】参考《MaxCompute Spark 开发指南》第四节在 pom 中添加依赖后重新打包即可。

6.6 外表迁移

1. HDFS-> MaxCompute 的数据迁移，原则上全部迁到 MaxCompute 内部表。
2. 如果客户场景要求必须通过外表访问外部文件，需要先将文件迁移到 OSS 或者 OTS，在 MaxCompute 中创建外部表，实现对文件的访问。
3. 注意：MaxCompute 外部表支持的格式包括：ORC、PARQUET、SEQUENCEFILE、RCFILE、AVRO 和 TEXTFILE。

6.7 Pipeline 迁移

1. 根据模板上传 Dataworks 项目描述文档，参见 6.4.2。
2. 上传完成后，在 Dataworks 上做 SQL 语法转换，参见 6.5.1.2。
3. 转换后的 SQL 会根据 workflow.xml 中的配置，自动生成项目空间下开发环境的工作流节点。
4. 客户可以运行测试实例，验证后发布到生产环境。

7 经典用例

7.1 基本功能

7.1.1 准备工具和环境

预先下载好工具包：odps-data-carrier.zip

```
[root@node-1 test]# ll
total 231544
-rw-r--r-- 1 root root 237095263 Jul  3 12:19 odps-data-carrier.zip
```

此次我们要迁移的表为 database test 下名为 test 的表，该表的信息如下：

```
hive> desc test.test;
OK
col1                    string
col2                    bigint
col3                    string

# Partition Information
# col_name              data_type               comment

col3                    string
```

分区信息如下：

```
hive> show partitions test.test;
OK
col3=2019-06-25
col3=2019-06-26
```

这两个分区中，每个分区中有 100 条数据

本次数据迁移验证环节会用到 MaxCompute 客户端 odpscmd，下载页面

7.1.2 解压工具包，并配置 MaxCompute 连接信息

执行：unzip odps-data-carrier.zip

```
[root@node-1 test]# ll
total 231548
drwxr-xr-x 5 root root      4096 Jul  3 12:01 odps-data-carrier
-rw-r--r-- 1 root root 237095263 Jul  3 12:19 odps-data-carrier.zip
```

配置 odps 账号，其中 tunnel_endpoint 为可选项：vim odps-data-carrier/odps_config.ini

```
project_name=
access_id=
access_key=
end_point=
tunnel_endpoint=
```

7.1.3 运行 meta-carrier 收集 meta 信息

执行：sh odps-data-carrier/bin/meta-carrier -u thrift://127.0.0.1:9083 -o meta

```
[root@node-1 test]# sh odps-data-carrier/bin/meta-carrier -u thrift://127.0.0.1:9083 -o meta
ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.
log4j:WARN No appenders could be found for logger (org.apache.hadoop.util.Shell).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Working on default
 100% [==========] 3/3 (0:00:01 / 0:00:00) Working on default.staples
Working on test
 100% [==========] 1/1 (0:00:01 / 0:00:00) Working on test.test
Working on test_types
 100% [==========] 8/8 (0:00:01 / 0:00:00) Working on test_types.test_partition
```

7.1.4 修改 meta-carrier 的输出，调整 hive 与 odps 的映射

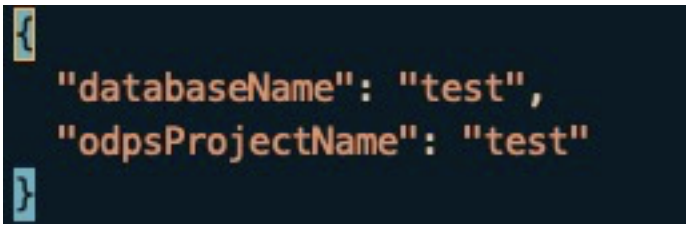
由于我们本次想迁移的表为 test.test，我们删除其他 db 和表，并修改 database test 到 MaxCompute project 的映射，删除之前的目录结构为：

```
[root@node-1 test]# tree meta
meta
├── default
│   ├── default.json
│   └── table_meta
│       ├── batters.json
│       ├── calcs.json
│       └── staples.json
├── global.json
├── test
│   ├── partition_meta
│   │   └── test.json
│   ├── table_meta
│   │   └── test.json
│   └── test.json
└── test_types
    ├── partition_meta
    │   └── test_partition.json
    ├── table_meta
    │   ├── complex_types_2.json
    │   ├── complex_types.json
    │   ├── date_and_time_types.json
    │   ├── dummy.json
    │   ├── misc_types.json
    │   ├── numeric_types.json
    │   ├── string_types.json
    │   └── test_partition.json
    └── test_types.json
```

删除之后为：

修改 database test 到 MaxCompute project 的映射，vim meta/test/test.json，

修改前为：

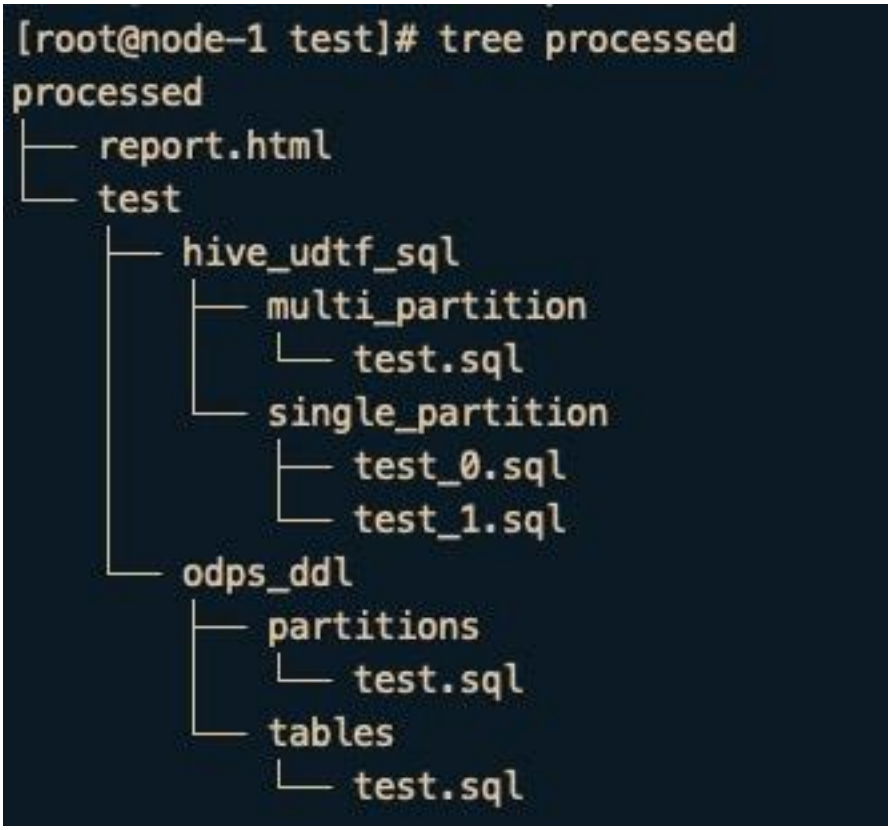


修改后为：

7.1.5 生成 ODPS DDL、Hive SQL 以及兼容性报告

执行：sh odps-data-carrier/bin/meta-processor -i meta -o processed

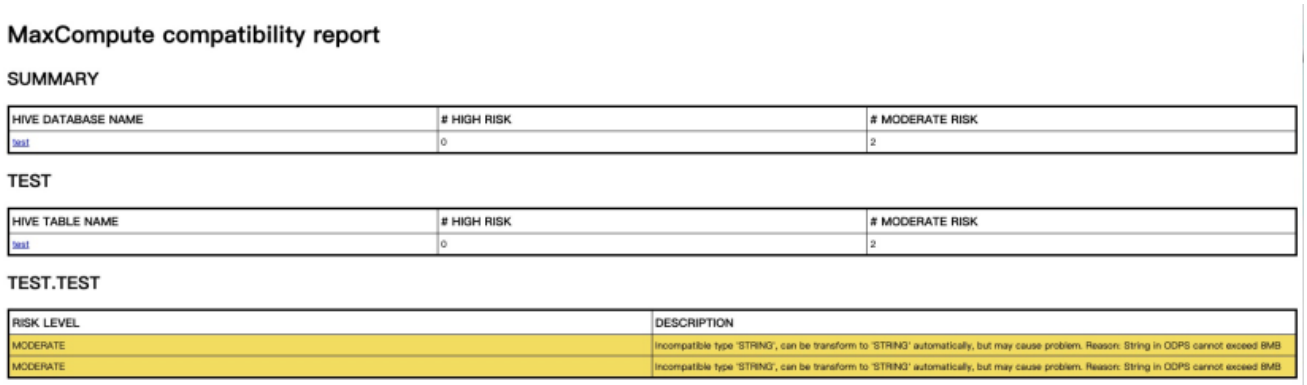
生成的目录结构为：



可以看到，生成的 MaxCompute ddl 和 hive sql 仅包含 test.test 这张表

7.1.6 查看兼容性报告，调整直到兼容性报告符合预期

为了方便查看，我们将 processed/report.html 下载到本地查看，效果如下：



MaxCompute compatibility report

SUMMARY

HIVE DATABASE NAME	# HIGH RISK	# MODERATE RISK
test	0	2

TEST

HIVE TABLE NAME	# HIGH RISK	# MODERATE RISK
test	0	2

TEST.TEST

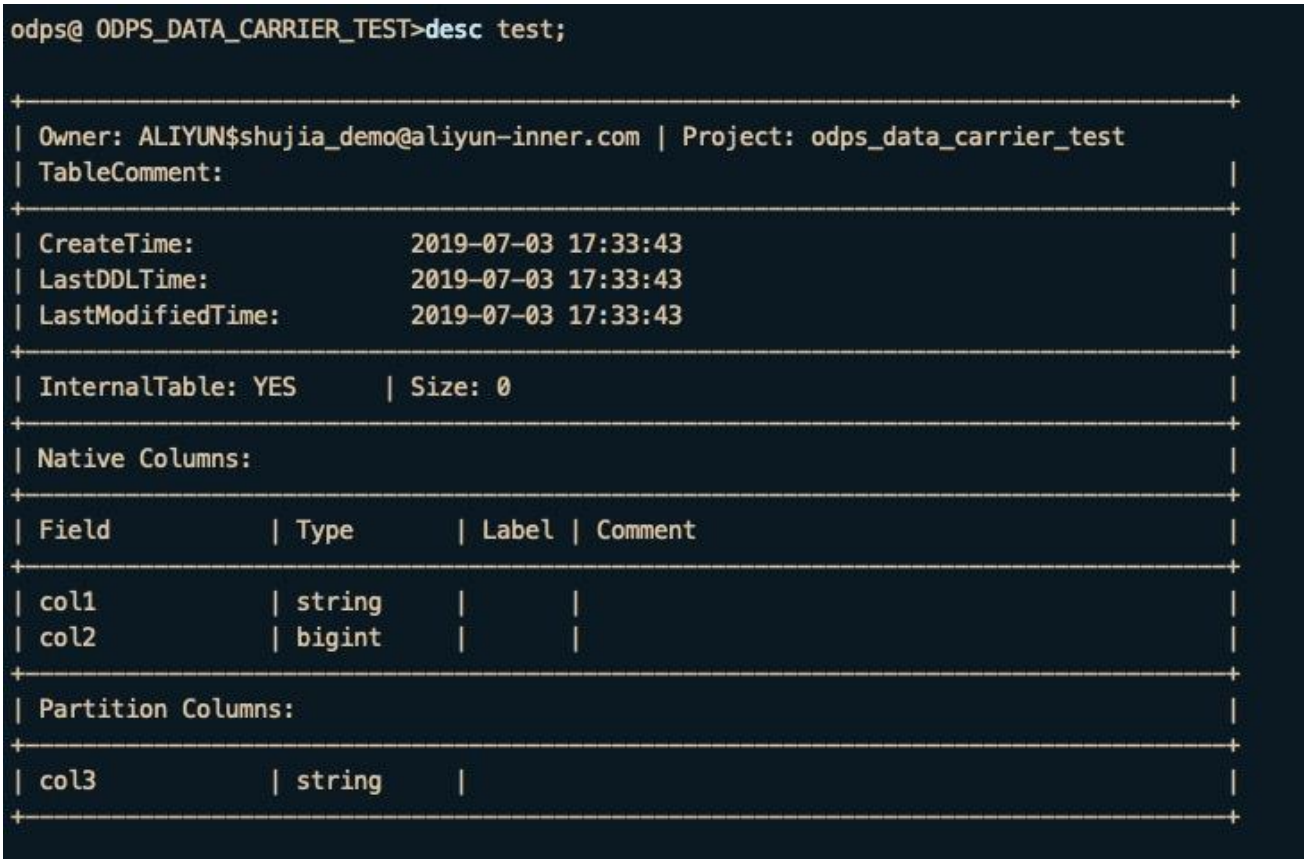
RISK LEVEL	DESCRIPTION
MODERATE	Incompatible type 'STRING', can be transform to 'STRING' automatically, but may cause problem. Reason: String in ODPS cannot exceed 8MB
MODERATE	Incompatible type 'STRING', can be transform to 'STRING' automatically, but may cause problem. Reason: String in ODPS cannot exceed 8MB

评估并解决报告中的风险点，之后如果有必要，重新执行第 4、5 步

7.1.7 运行 odps_ddl_runner.py 生成 odps 表和分区

执行：python3 odps-data-carrier/bin/odps_ddl_runner.py --input processed

可以看到，这个命令自动生成了对应的表和 partition，之后我们用 odpscmd 工具进行验证：



可以看到 MaxCompute 中的表已经建好了，之后我们在看下分区：

```
odps@ ODPS_DATA_CARRIER_TEST>list partitions test;

col3=2019-06-25
col3=2019-06-26
```

可以看到，分区也已经建好了

7.1.8 运行 hive_udtf_sql_runner.py，将 hive 的数据同步到 odps

执行：python3 odps-data-carrier/bin/hive_udtf_sql_runner.py --input_all processed/

用 odpscmd 工具验证数据是否被上传到 mc：

```
odps@ ODPS_DATA_CARRIER_TEST>desc test;

+------------------------------------------------------------------------------------+
| Owner: ALIYUN$shujia_demo@aliyun-inner.com | Project: odps_data_carrier_test
| TableComment:                                                                      |
+------------------------------------------------------------------------------------+
| CreateTime:                2019-07-03 17:33:43                                     |
| LastDDLTime:               2019-07-03 17:33:43                                     |
| LastModifiedTime:          2019-07-03 17:37:46                                     |
+------------------------------------------------------------------------------------+
| InternalTable: YES      | Size: 3512                                               |
+------------------------------------------------------------------------------------+
| Native Columns:                                                                    |
+------------------------------------------------------------------------------------+
| Field           | Type        | Label | Comment                                    |
+------------------------------------------------------------------------------------+
| col1            | string      |       |                                            |
| col2            | bigint      |       |                                            |
+------------------------------------------------------------------------------------+
| Partition Columns:                                                                 |
+------------------------------------------------------------------------------------+
| col3            | string      |                                                    |
+------------------------------------------------------------------------------------+
```

可以看到，数据已经成功上传，下面验证其中一个 partition：

```
odps@ ODPS_DATA_CARRIER_TEST>select count(*) from test where col3='2019-06-25';

ID = 20190703094046426g5hpz192
Log view:
http://logview.odps.aliyun.com/logview/?h=http://service.odps.aliyun.com/api&p=ODPS_DATA_CARRIER_TEST&i=20190703094046426g5hpz192&token=L1VrWG1oZUR0MWRXWitHVX
kzWEgvSzQ2aU5ZPSxPRFBTX09CTzoxMzI5MzgzMDA0NTQwNjUxLDE1NjQ3Mzg4NDYseyJTdGF0ZW1lbnQiOlt7IkFjdGlvbiI6WyJvZHBzOlJlYWQiXSwiRWZmZWN0IjoiQWxsb3ciLCJSZXNvdXJjZSI6WyJh
Y3M6b2RwczoqOnByb2plY3RzL29kcHNfZGF0YV9jYXJyaWVyX3Rlc3QvaW5zdGFuY2VzLzIwMTkwNzAzMDk0MDQ2NDI2ZzVocHoxOTIiXX1dLCJWZXJzaW9uIjoiMSJ9
Job Queueing.
Summary:
resource cost: cpu 0.00 Core * Min, memory 0.00 GB * Min
inputs:
        odps_data_carrier_test.test/col3=2019-06-25: 100 (1392 bytes)
outputs:
Job run time: 0.000
Job run mode: service job
Job run engine: execution engine
M1:
        instance count: 1
        run time: 0.000
        instance time:
                min: 0.000, max: 0.000, avg: 0.000
        input records:
                TableScan1: 100  (min: 100, max: 100, avg: 100)
        output records:
                StreamLineWrite1: 1  (min: 1, max: 1, avg: 1)
        metrics_output_count:
                HashAgg1: 1  (min: 1, max: 1, avg: 1)
                StreamLineWrite1: 1  (min: 1, max: 1, avg: 1)
                TableScan1: 100  (min: 100, max: 100, avg: 100)
        metrics_inner_time_ms:
                HashAgg1: 0  (min: 0, max: 0, avg: 0)   MaxInstance: 0
                StreamLineWrite1: 4  (min: 4, max: 4, avg: 4)   MaxInstance: 0
                TableScan1: 0  (min: 0, max: 0, avg: 0) MaxInstance: 0
R2_1:
        instance count: 1
        run time: 0.000
        instance time:
                min: 0.000, max: 0.000, avg: 0.000
        input records:
                StreamLineRead1: 1  (min: 1, max: 1, avg: 1)
        output records:
                AdhocSink1: 1  (min: 1, max: 1, avg: 1)
        metrics_output_count:
                AdhocSink1: 1  (min: 1, max: 1, avg: 1)
                Project1: 1  (min: 1, max: 1, avg: 1)
                SortedAgg1: 1  (min: 1, max: 1, avg: 1)
                StreamLineRead1: 1  (min: 1, max: 1, avg: 1)
        metrics_inner_time_ms:
                AdhocSink1: 63  (min: 63, max: 63, avg: 63)     MaxInstance: 0
                Project1: 0  (min: 0, max: 0, avg: 0)   MaxInstance: 0
                SortedAgg1: 0  (min: 0, max: 0, avg: 0) MaxInstance: 0
                StreamLineRead1: 0  (min: 0, max: 0, avg: 0)    MaxInstance: 0


+------------+
| _c0        |
+------------+
| 100        |
+------------+
```

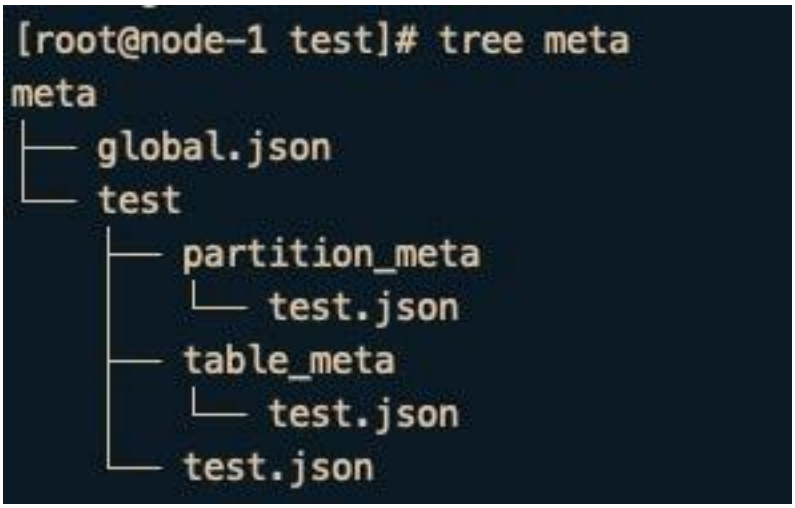
可以看到，partition 中 record 的数量符合预期。

7.2 进阶功能

7.2.1 仅生成指定 database 或 table 的 metadata

在上面的例子中，我们抓去了 hive 中所有 database 和表的 metadata，但在很多环境下，我们倾向于一次处理一个 database 或一张表，因此 meta-carrier 工具提供了抓去指定 database 或 table 的 metadata 的能力：

执行：sh odps-data-carrier/bin/meta-carrier -u thrift://127.0.0.1:9083 -d test -t test -o meta



可以看到，这里我们生成的 metadata 仅包含了 test.test 这张表

7.2.2 灵活的 hive 到 max compute 映射

在上面的例子中，我们将 hive 的 test.test 表映射到 mc 中 ODPS_DATA_CARRIER_TEST.test 这张表，然而，我们提供了更强大的能力，比如说修改 hive 表到 mc 的表明与列名映射，设置 mc 中表的 life cycle，增加 comment，等等。

用户可以编辑 meta-carrier 生成的 metadata 来做到上述的事情，可编辑的部分如下：

```
global.json:
{
    "datasourceType" : "HIVE", // 目前仅支持 HIVE
    "odpsVersion" : "1.0" , // 默认 ODPS 版本为 1.0
    "hiveCompatible" : false // 在 ODPS 2.0 下，是否打开 hive compatible 开关
 }

[database name].json:
 {
```

```
    "databaseName" : xxx, // Hive 数据库名
    "odpsProjectName" : xxx //  对应的 ODPS 项目名，默认与 Hive 数据库名相同
 }

[table name].json:
 {
     "tableName" : "xxx", // Hive 表名
     "odpsTableName" : "xxx", //  对应的 ODPS 表名，默认与 Hive 表名相同
     "lifeCycle" : 10, // ODPS 表的 life cycle，默认为空，即不启用 life cycle
     "comment" : "xxx", // ODPS 表的 comment，默认为空
     "ifNotExists" : true, //  创建 ODPS 表时是否加 if not exists，默认不加
     "columns" : [
         {
             "name" : "column_1", // Hive 列名
             "odpsColumnName" : "odps_column_1", //  对应的 ODPS 列名，默认与 Hive 列名相同
             "type" : "bigint", // ODPS 列的类型，用户暂时不可自行修改
             "comment" : "xxx" // ODPS 列的 comment
         },
         ...
     ],
     "partitionColumns" : [
         {
             "name" : "column_1", // Hive 分区列名
             "odpsColumnName" : "odps_column_1", //  对应的 ODPS 分区列名，默认与 Hive 分区
列名相同
             "type" : "bigint", // ODPS 分区列类型
             "comment" : "xxx" // ODPS 分区列的 comment
         },
         ...
     ]
 }
```

7.2.3 单表/单分区迁移

在运行 hive sql 进行数据迁移的时候，我们提供了两种模式，input_all 模式与 input_single_file 模式。

在 input_all 模式下，我们给一个 meta-processor 生成的目录，之后 odps_hive_udtf_runner 会自动遍历该目录下的文件，并串行执行里面的 hive sql，例如：

python3 odps-data-carrier/bin/hive_udtf_sql_runner.py --input_all processed/

在 input_single_file 模式下，我们给一个 hive sql 文件路径，odps_hive_udtf_runner 会从该文件中读取 hive sql 并执行。例如：

python3 odps-data-carrier/bin/hive_udtf_sql_runner.py --input_single_file processed/test/hive_udtf_sql/single_partition/test_0.sql

input_single_file 模式可以帮助我们熟悉工具，并且在数据量大的场景下可以控制迁移的进度。

8 最佳实践

8.1 【场景 1】Hive 数据和 Oozie 工作流任务如何迁移到 MaxCompute 和 Dataworks?

8.1.1 网络环境检查

1、 请确保您的 Hadoop 集群所在的 IDC 网络，与您选择的阿里云服务（MaxCompute、Dataworks)所在的 Region 能够正常建立网络连接。如果无法访问，建议您预先拉专线，保证网络可用。

2、 您可以使用 MMA Agent 的 network-measurement-tool 工具检查您的 Hadoop 集群到 MaxCompute 各 Region 网络的连接质量和网络上下行的传输速率评测。参见 6.3.1.3。

8.1.2 开通 MaxCompute 和 Dataworks 服务

1. 开通 MaxCompute 服务，参见文档：
   https://help.aliyun.com/document_detail/58226.html?spm=a2c4g.11174283.6.567.3836590evMSPuu
2. 开通 Dataworks 服务，参见文档：
   https://help.aliyun.com/document_detail/74293.html?spm=a2c4g.11186623.6.576.1e3f2f04QMK9Zi
3. 创建 Project，参见文档：
   https://help.aliyun.com/document_detail/27815.html?spm=a2c4g.11186623.6.568.628a417dnN9Dg3

8.1.3 安装 MMA Agent 客户端工具

1. 安装 MMA Agent 并使用 meta-carrier 获取 hive metadata，参见 6.3.1.1。
2. 使用 meta-processor 生成 ODPS DDL 和 Hive UDTF SQL，参见 6.4.1 的第 3、4 步。

8.1.4 批量迁移 Hive 的表和数据

8.1.4.1 方案一：使用客户端工具迁移数据

参见 6.4.1

8.1.4.2 方案二：使用 Dataworks 服务迁移数据

参见 6.4.2

8.1.5 单表迁移

8.1.5.1 指定表名获取 Meta

在使用 meta-carrier 工具获取 Hive Meta 数据的时候，可以通过参数指定需要迁移的 database 名称和表名，生成的 meta 目录就会只有这一个 table 的 json 文件。

```
Usage:
  ./meta-carrier -u <uri> -o <output dir> [-h] [-d <database>] [-t <table>]

Options:
    -d,--database <database>      Specify a database
    -t,--table <table>            Specify a table
    -o,--output-dir <output-dir>  Output directory
    -u,--uri <uri>                hive metastore thrift uri
    -o,--output-dir <output-dir>  Output directory
```

其余迁移操作步骤与 7.1.4.1 相同。

8.1.6 批量迁移 Oozie 工作流和节点任务

8.1.6.1 Oozie 工作流和节点任务迁移

1. 参见 6.4.2，您需要在 6.4.2.2 中配置 oozie 工作流模板，如下图：

```
.
├── project.xml # 项目描述文件，文件名固定为project.xml
└── src
    ├── assets # 用户已有的大数据平台任务
    │   ├── hive_meta # hive meta的数据目录，由铭宥的meta工具生成
    │   │   ├── meta
    │   │   │   ├── default
    │   │   │   ├── global.json
    │   │   │   └── test_types
    │   │   └── result
    │   │       └── test_types
    │   └── oozie # 此目录下存放oozie的调度任务
    │       ├── hive2_sample # oozie的某一个调度任务目录
    │       │   ├── README
    │       │   ├── job.properties
    │       │   ├── job.properties.security
    │       │   ├── script.q
    │       │   ├── workflow.xml
    │       │   └── workflow.xml.security
    │       ├── hive_sample # oozie的某一个调度任务目录
    │       │   ├── README
    │       │   ├── job.properties
    │       │   ├── script.q
    │       │   ├── workflow.xml
    │       │   └── workflow.xml.security
    └── workflows # DataWorks工作流描述目录，目录名固定为workflows
        ├── oozie_hive-wf # 工作流名称
        │   ├── nodes
        │   │   ├── hive-node # 节点名称
        │   │   │   └── script.q # 节点代码文件
        │   │   └── sqoop2_mysql2hive # 节点名称
        │   │       └── sqoop2_mysql2hive # 节点代码文件
```

2. 配置完成并上传打包文件后，Dataworks 服务会自动转换并生成 Dataworks 的工作流和节点任务。【注意】：仅支持发布到开发环境，需要客户自己测试验证后，发布到生产环境。

8.1.6.2 创建 Dataworks 标准工作流

1. 参见 6.4.2，如果您使用其他调度引擎，需要在 6.4.2.2 中按照 Dataworks 的标准模板配置您的工作流节点，如下图：

2. 配置完成并上传打包文件后，Dataworks 服务会自动转换并生成 Dataworks 的工作流和节点任务。【注意】：仅支持发布到开发环境，需要客户自己测试验证后，发布到生产环境。

8.1.6.3 Dataworks 服务支持 Oozie+Dataworks 混乱模式的工作流迁移

即：支持 7.5.1.1 和 7.5.1.2 两种混合配置模式，Dataworks 服务会为两种工作流配置做叠加处理。