尚硅谷大数据技术之Flink-CDC 3.0

 （作者：尚硅谷研究院）

版本：V3.0

# 第1章 CDC简介

## 1.1 什么是CDC

CDC是Change Data Capture(变更数据获取)的简称。核心思想是，监测并捕获数据库的变动（包括数据或数据表的插入、更新以及删除等），将这些变更按发生的顺序完整记录下来，写入到消息中间件中以供其他服务进行订阅及消费。

## 1.2 CDC的种类

CDC主要分为基于查询和基于Binlog两种方式，我们主要了解一下这两种之间的区别：

|  |  |  |
| --- | --- | --- |
|  | 基于查询的CDC | 基于Binlog的CDC |
| 开源产品 | Sqoop、DataX | Canal、Maxwell、Debezium |
| 执行模式 | Batch | Streaming |
| 是否可以捕获所有数据变化 | 否 | 是 |
| 延迟性 | 高延迟 | 低延迟 |
| 是否增加数据库压力 | 是 | 否 |

## 1.3 Flink-CDC

Flink社区开发了 flink-cdc-connectors 组件，这是一个可以直接从 MySQL、PostgreSQL 等数据库直接读取全量数据和增量变更数据的 source 组件。

目前也已开源，开源地址：<https://github.com/ververica/flink-cdc-connectors>



# 第2章 FlinkCDC案例实操

## 2.1 数据准备

### 2.1.1 在MySQL中创建数据库及表

[atguigu@hadoop103 flink-cdc-3.0.0]$ mysql -uroot -p000000

mysql: [Warning] Using a password on the command line interface can be insecure.

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 8

Server version: 8.0.31 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its

affiliates. Other names may be trademarks of their respective

owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database test;

Query OK, 1 row affected (0.00 sec)

mysql> create database test\_route;

Query OK, 1 row affected (0.00 sec)

mysql> use test;

Database changed

mysql> CREATE TABLE t1(

 -> `id` VARCHAR(255) PRIMARY KEY,

 -> `name` VARCHAR(255)

 -> );

Query OK, 0 rows affected (0.00 sec)

mysql> CREATE TABLE t2(

 -> `id` VARCHAR(255) PRIMARY KEY,

 -> `name` VARCHAR(255)

 -> );

Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE t3(

 -> `id` VARCHAR(255) PRIMARY KEY,

 -> `sex` VARCHAR(255)

 -> );

Query OK, 0 rows affected (0.01 sec)

mysql> use test\_route;

Database changed

mysql> CREATE TABLE t1(

 -> `id` VARCHAR(255) PRIMARY KEY,

 -> `name` VARCHAR(255)

 -> );

Query OK, 0 rows affected (0.00 sec)

mysql> CREATE TABLE t2(

 -> `id` VARCHAR(255) PRIMARY KEY,

 -> `name` VARCHAR(255)

 -> );

Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE t3(

 -> `id` VARCHAR(255) PRIMARY KEY,

 -> `sex` VARCHAR(255)

 -> );

Query OK, 0 rows affected (0.01 sec)

### 2.1.2 插入数据

**1）在test数据库中插入数据**

use test;

INSERT INTO t1 VALUES('1001','zhangsan');

INSERT INTO t1 VALUES('1002','lisi');

INSERT INTO t1 VALUES('1003','wangwu');

INSERT INTO t2 VALUES('1001','zhangsan');

INSERT INTO t2 VALUES('1002','lisi');

INSERT INTO t2 VALUES('1003','wangwu');

INSERT INTO t3 VALUES('1001','F');

INSERT INTO t3 VALUES('1002','F');

INSERT INTO t3 VALUES('1003','M');

**2）在test\_route数据库中插入数据**

use test\_route;

INSERT INTO t1 VALUES('1001','zhangsan');

INSERT INTO t1 VALUES('1002','lisi');

INSERT INTO t1 VALUES('1003','wangwu');

INSERT INTO t2 VALUES('1004','zhangsan');

INSERT INTO t2 VALUES('1005','lisi');

INSERT INTO t2 VALUES('1006','wangwu');

INSERT INTO t3 VALUES('1001','F');

INSERT INTO t3 VALUES('1002','F');

INSERT INTO t3 VALUES('1003','M');

### 2.1.3 开启MySQL Binlog并重启MySQL

[atguigu@hadoop103 ~]$ sudo vim /etc/my.cnf

#添加如下配置信息,开启`test`以及`test\_route`数据库的Binlog

#数据库id

server-id = 1

##启动binlog，该参数的值会作为binlog的文件名

log-bin=mysql-bin

##binlog类型，maxwell要求为row类型

binlog\_format=row

##启用binlog的数据库，需根据实际情况作出修改

binlog-do-db=test

binlog-do-db=test\_route

## 2.2 DataStream方式的应用

### 2.2.1 导入依赖

<properties>

 <maven.compiler.source>8</maven.compiler.source>

 <maven.compiler.target>8</maven.compiler.target>

 <flink-version>1.18.0</flink-version>

 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

 </properties>

 <dependencies>

 <dependency>

 <groupId>org.apache.flink</groupId>

 <artifactId>flink-java</artifactId>

 <version>${flink-version}</version>

 </dependency>

 <dependency>

 <groupId>org.apache.flink</groupId>

 <artifactId>flink-streaming-java</artifactId>

 <version>${flink-version}</version>

 </dependency>

 <dependency>

 <groupId>org.apache.flink</groupId>

 <artifactId>flink-clients</artifactId>

 <version>${flink-version}</version>

 </dependency>

 <dependency>

 <groupId>org.apache.flink</groupId>

 <artifactId>flink-table-planner\_2.12</artifactId>

 <version>${flink-version}</version>

 </dependency>

 <dependency>

 <groupId>org.apache.flink</groupId>

 <artifactId>flink-table-runtime</artifactId>

 <version>${flink-version}</version>

 </dependency>

 <dependency>

 <groupId>org.apache.flink</groupId>

 <artifactId>flink-table-api-java-bridge</artifactId>

 <version>${flink-version}</version>

 </dependency>

 <dependency>

 <groupId>org.apache.flink</groupId>

 <artifactId>flink-connector-base</artifactId>

 <version>${flink-version}</version>

 </dependency>

 <dependency>

 <groupId>com.ververica</groupId>

 <artifactId>flink-connector-mysql-cdc</artifactId>

 <version>3.0.0</version>

 </dependency>

 <dependency>

 <groupId>mysql</groupId>

 <artifactId>mysql-connector-java</artifactId>

 <version>8.0.31</version>

 </dependency>

 </dependencies>

 <build>

 <plugins>

 <plugin>

 <groupId>org.apache.maven.plugins</groupId>

 <artifactId>maven-assembly-plugin</artifactId>

 <version>3.0.0</version>

 <configuration>

 <descriptorRefs>

 <descriptorRef>jar-with-dependencies</descriptorRef>

 </descriptorRefs>

 </configuration>

 <executions>

 <execution>

 <id>make-assembly</id>

 <phase>package</phase>

 <goals>

 <goal>single</goal>

 </goals>

 </execution>

 </executions>

 </plugin>

 </plugins>

 </build>

### 2.2.2 编写代码

import com.ververica.cdc.connectors.mysql.source.MySqlSource;

import com.ververica.cdc.connectors.mysql.table.StartupOptions;

import com.ververica.cdc.debezium.JsonDebeziumDeserializationSchema;

import org.apache.flink.api.common.eventtime.WatermarkStrategy;

import org.apache.flink.api.common.restartstrategy.RestartStrategies;

import org.apache.flink.api.common.time.Time;

import org.apache.flink.runtime.state.hashmap.HashMapStateBackend;

import org.apache.flink.streaming.api.CheckpointingMode;

import org.apache.flink.streaming.api.datastream.DataStreamSource;

import org.apache.flink.streaming.api.environment.CheckpointConfig;

import org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;

import java.util.Properties;

public class FlinkCDCDataStreamTest {

 public static void main(String[] args) throws Exception {

 // TODO 1. 准备流处理环境

 StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();

 env.setParallelism(1);

 // TODO 2. 开启检查点 Flink-CDC将读取binlog的位置信息以状态的方式保存在CK,如果想要做到断点续传,

 // 需要从Checkpoint或者Savepoint启动程序

 // 2.1 开启Checkpoint,每隔5秒钟做一次CK ,并指定CK的一致性语义

 env.enableCheckpointing(3000L, CheckpointingMode.EXACTLY\_ONCE);

 // 2.2 设置超时时间为 1 分钟

 env.getCheckpointConfig().setCheckpointTimeout(60 \* 1000L);

 // 2.3 设置两次重启的最小时间间隔

 env.getCheckpointConfig().setMinPauseBetweenCheckpoints(3000L);

 // 2.4 设置任务关闭的时候保留最后一次 CK 数据

 env.getCheckpointConfig().enableExternalizedCheckpoints(

 CheckpointConfig.ExternalizedCheckpointCleanup.RETAIN\_ON\_CANCELLATION);

 // 2.5 指定从 CK 自动重启策略

 env.setRestartStrategy(RestartStrategies.failureRateRestart(

 3, Time.days(1L), Time.minutes(1L)

 ));

 // 2.6 设置状态后端

 env.setStateBackend(new HashMapStateBackend());

 env.getCheckpointConfig().setCheckpointStorage(

 "hdfs://hadoop102:8020/flinkCDC"

 );

 // 2.7 设置访问HDFS的用户名

 System.setProperty("HADOOP\_USER\_NAME", "atguigu");

 // TODO 3. 创建 Flink-MySQL-CDC 的 Source

 // initial:Performs an initial snapshot on the monitored database tables upon first startup, and continue to read the latest binlog.

// earliest:Never to perform snapshot on the monitored database tables upon first startup, just read from the beginning of the binlog. This should be used with care, as it is only valid when the binlog is guaranteed to contain the entire history of the database.

// latest:Never to perform snapshot on the monitored database tables upon first startup, just read from the end of the binlog which means only have the changes since the connector was started.

// specificOffset:Never to perform snapshot on the monitored database tables upon first startup, and directly read binlog from the specified offset.

// timestamp:Never to perform snapshot on the monitored database tables upon first startup, and directly read binlog from the specified timestamp.The consumer will traverse the binlog from the beginning and ignore change events whose timestamp is smaller than the specified timestamp.

 MySqlSource<String> mySqlSource = MySqlSource.<String>builder()

 .hostname("hadoop103")

 .port(3306)

 .databaseList("test ") // set captured database

 .tableList("test.t1") // set captured table

 .username("root")

 .password("000000")

 .deserializer(new JsonDebeziumDeserializationSchema()) // converts SourceRecord to JSON String

 .startupOptions(StartupOptions.initial())

 .build();

 // TODO 4.使用CDC Source从MySQL读取数据

 DataStreamSource<String> mysqlDS =

 env.fromSource(

 mySqlSource,

 WatermarkStrategy.noWatermarks(),

 "MysqlSource");

 // TODO 5.打印输出

 mysqlDS.print();

 // TODO 6.执行任务

 env.execute();

 }

}

### 案例测试

**1）打包并上传至Linux**



**2）启动HDFS集群**

[atguigu@hadoop102 flink-1.18.0]$ start-dfs.sh

**3）启动Flink集群**

[atguigu@hadoop102 flink-1.18.0]$ bin/start-cluster.sh

**4）启动程序**

[atguigu@hadoop102 flink-1.18.0]$ bin/flink run -m hadoop102:8081 -c com.atguigu.cdc.FlinkCDCDataStreamTest ./flink-cdc-test.jar

**5）观察TaskManager日志，会从头读取表数据**

**6）给当前的Flink程序创建Savepoint**

[atguigu@hadoop102 flink-local]$ bin/flink savepoint JobId hdfs://hadoop102:8020/flinkCDC/save

**在WebUI中cancelJob**

**在MySQL的test.t1表中添加、修改或者删除数据**

**从Savepoint重启程序**

[atguigu@hadoop102 flink-standalone]$ bin/flink run -s hdfs://hadoop102:8020/flinkCDC/save/savepoint-5dadae-02c69ee54885 -c com.atguigu.cdc. FlinkCDCDataStreamTest ./gmall-flink-cdc.jar

**观察TaskManager日志，会从检查点读取表数据**

## FlinkSQL方式的应用

### 2.3.1 编写代码

package com.atguigu;

import org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;

import org.apache.flink.table.api.Table;

import org.apache.flink.table.api.bridge.java.StreamTableEnvironment;

public class FlinkSQLTest {

 public static void main(String[] args) {

 StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();

 env.setParallelism(1);

 StreamTableEnvironment tableEnv = StreamTableEnvironment.create(env);

 tableEnv.executeSql("" +

 "create table t1(\n" +

 " id string primary key NOT ENFORCED,\n" +

 " name string

 ") WITH (\n" +

 " 'connector' = 'mysql-cdc',\n" +

 " 'hostname' = 'hadoop103',\n" +

 " 'port' = '3306',\n" +

 " 'username' = 'root',\n" +

 " 'password' = '000000',\n" +

 " 'database-name' = 'test',\n" +

 " 'table-name' = 't1'\n" +

 ")");

 Table table = tableEnv.sqlQuery("select \* from t1");

 table.execute().print();

 }

}

### 2.3.2 代码测试

**直接运行即可，打包到集群测试与DataStream相同！**

## MySQL到Doris的StreamingETL实现（3.0）

### 2.4.1 环境准备

**1）安装FlinkCDC**

[atguigu@hadoop102 software]$ tar -zxvf flink-cdc-3.0.0-bin.tar.gz -C /opt/module/

**2）拖入MySQL以及Doris依赖包**

将flink-cdc-pipeline-connector-doris-3.0.0.jar以及flink-cdc-pipeline-connector-mysql-3.0.0.jar防止在FlinkCDC的lib目录下

### 2.4.2 同步变更

**2）编写MySQL到Doris的同步变更配置文件**

在FlinkCDC目录下创建job文件夹，并在该目录下创建同步变更配置文件

[atguigu@hadoop102 flink-cdc-3.0.0]$ mkdir job/

[atguigu@hadoop102 flink-cdc-3.0.0]$ cd job

[atguigu@hadoop102 job]$ vim mysql-to-doris.yaml

source:

 type: mysql

 hostname: hadoop103

 port: 3306

 username: root

 password: "000000"

 tables: test.\.\*

 server-id: 5400-5404

 server-time-zone: UTC+8

sink:

 type: doris

 fenodes: hadoop102:7030

 username: root

 password: "000000"

 table.create.properties.light\_schema\_change: true

 table.create.properties.replication\_num: 1

pipeline:

 name: Sync MySQL Database to Doris

 parallelism: 1

**3）启动任务并测试**

**（1）开启Flink集群，注意：在Flink配置信息中打开CheckPoint**

[atguigu@hadoop103 flink-1.18.0]$ vim conf/flink-conf.yaml

添加如下配置信息

execution.checkpointing.interval: 5000

分发该文件至其他Flink机器并启动Flink集群

[atguigu@hadoop103 flink-1.18.0]$ bin/start-cluster.sh

**（2）开启Doris FE**

[atguigu@hadoop102 fe]$ bin/start\_fe.sh

**（3）开启Doris BE**

[atguigu@hadoop102 be]$ bin/start\_be.sh

**（4）在Doris中创建test数据库**

[atguigu@hadoop103 doris]$ mysql -uroot -p000000 -P9030 -hhadoop102

mysql: [Warning] Using a password on the command line interface can be insecure.

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 0

Server version: 5.7.99 Doris version doris-1.2.4-1-Unknown

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its

affiliates. Other names may be trademarks of their respective

owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database test;

Query OK, 1 row affected (0.00 sec)

mysql> create database doris\_test\_route;

Query OK, 1 row affected (0.00 sec)

**（5）启动FlinkCDC同步变更任务**

[atguigu@hadoop103 flink-cdc-3.0.0]$ bin/flink-cdc.sh job/mysql-to-doris.yaml

**（6）刷新Doris中test数据库观察结果**

**（7）在MySQL的test数据中对应的几张表进行新增、修改数据以及新增列操作，并刷新Doris中test数据库观察结果**

### 2.4.3 路由变更

**1）编写MySQL到Doris的路由变更配置文件**

source:

 type: mysql

 hostname: hadoop103

 port: 3306

 username: root

 password: "000000"

 tables: test\_route.\.\*

 server-id: 5400-5404

 server-time-zone: UTC+8

sink:

 type: doris

 fenodes: hadoop102:7030

 benodes: hadoop102:7040

 username: root

 password: "000000"

 table.create.properties.light\_schema\_change: true

 table.create.properties.replication\_num: 1

route:

 - source-table: test\_route.t1

 sink-table: doris\_test\_route.doris\_t1

 - source-table: test\_route.t2

 sink-table: doris\_test\_route.doris\_t1

 - source-table: test\_route.t3

 sink-table: doris\_test\_route.doris\_t3

pipeline:

 name: Sync MySQL Database to Doris

 parallelism: 1

**2）启动任务并测试**

[atguigu@hadoop103 flink-cdc-3.0.0]$ bin/flink-cdc.sh job/mysql-to-doris-route.yaml

**3）刷新Doris中test\_route数据库观察结果**

**4）在MySQL的test\_route数据中对应的几张表进行新增、修改数据操作，并刷新Doris中doris\_test\_route数据库观察结果**