尚硅谷大数据技术之Flink-CDC

（作者：尚硅谷研究院）

版本：V2.0.0

1. CDC简介

http://www.dreamwu.com/post-1594.html

* 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. 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：

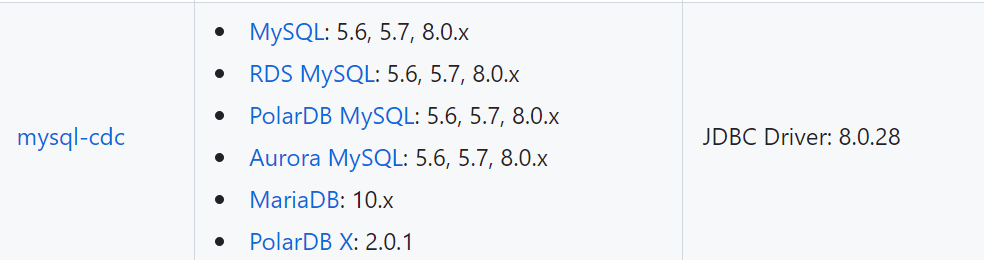
|  |  |  |
| --- | --- | --- |
|  | 基于查询的CDC | 基于Binlog的CDC |
| 开源产品 | Sqoop、Kafka JDBC Source | Canal、Maxwell、Debezium |
| 执行模式 | Batch | Streaming |
| 是否可以捕获所有数据变化 | 否 | 是 |
| 延迟性 | 高延迟 | 低延迟 |
| 是否增加数据库压力 | 是 | 否 |

* 1. 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，

<http://www.dreamwu.com/post-1594.html>

开源地址：<https://github.com/ververica/flink-cdc-connectors>



1. FlinkCDC案例实操
   1. 开启MySQL Binlog并重启MySQL

#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=gmall

binlog-do-db=gmall2023\_config

* 1. DataStream方式的应用
     1. 导入依赖

<properties>

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

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

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

**<flink.version>1.17.2</flink.version>**

**<flink-cdc.vesion>2.4.2</flink-cdc.vesion>**

**<hadoop.version>3.3.4</hadoop.version>**

</properties>

<dependencies>

<dependency>

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

<artifactId>flink-clients</artifactId>

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

<scope>provided</scope>

</dependency>

<dependency>

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

<artifactId>flink-connector-kafka</artifactId>

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

</dependency>

<dependency>

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

<artifactId>flink-json</artifactId>

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

<scope>provided</scope>

</dependency>

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>fastjson</artifactId>

<version>1.2.83</version>

</dependency>

<!--如果保存检查点到hdfs上，需要引入此依赖-->

<dependency>

<groupId>org.apache.hadoop</groupId>

<artifactId>hadoop-client-api</artifactId>

<version>${hadoop.version}</version>

<!--<scope>provided</scope>-->

</dependency>

<dependency>

<groupId>org.apache.hadoop</groupId>

<artifactId>hadoop-client-runtime</artifactId>

<version>${hadoop.version}</version>

<!--<scope>provided</scope>-->

</dependency>

<dependency>

<groupId>org.apache.logging.log4j</groupId>

<artifactId>log4j-to-slf4j</artifactId>

<version>2.14.0</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>org.apache.logging.log4j</groupId>

<artifactId>log4j-api</artifactId>

<version>2.14.0</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>1.7.36</version>

</dependency>

<!--cdc 依赖-->

<dependency>

<groupId>com.ververica</groupId>

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

<version>${flink-cdc.vesion}</version>

</dependency>

<!-- flink sql 相关的依赖: 使用 cdc 必须导入 sql 依赖-->

<dependency>

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

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

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

<scope>provided</scope>

</dependency>

<dependency>

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

<artifactId>flink-table-planner-loader</artifactId>

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

<scope>provided</scope>

</dependency>

<dependency>

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

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

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

<scope>provided</scope>

</dependency>

</dependencies>

* + 1. 编写代码

package com.atguigu;

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.runtime.state.hashmap.HashMapStateBackend;

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

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

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

public class FlinkCDC\_01\_DS {

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

// TODO 1. 初始化流处理环境

StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();

env.setParallelism(1);

// TODO 2. 检查点

// 2.1 启用检查点

env.enableCheckpointing(10 \* 1000L);

// 2.2 设置相邻两次检查点最小间隔

env.getCheckpointConfig().setMinPauseBetweenCheckpoints(30 \* 1000L);

// 2.3 设置取消 Job 时检查点的清理模式

env.getCheckpointConfig().setExternalizedCheckpointCleanup(

CheckpointConfig.ExternalizedCheckpointCleanup.RETAIN\_ON\_CANCELLATION

);

// 2.4 设置状态后端类型

env.setStateBackend(new HashMapStateBackend());

// 2.5 设置检查点存储路径

env.getCheckpointConfig().setCheckpointStorage("hdfs://hadoop102:8020/flinkCDC");

// 2.6 设置 HDFS 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("hadoop102")

.port(3306)

.databaseList("gmall2023\_config") // set captured database

.tableList("gmall2023\_config.test") // 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();

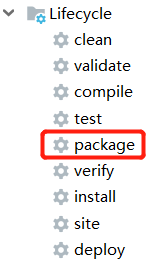
// TODO 6.执行任务

env.execute();

}

}

* + 1. 案例测试
       - 1. 打包并上传至Linux



* + - * 1. 启动HDFS集群

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

* + - * 1. 启动Flink集群

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

* + - * 1. 启动程序

[atguigu@hadoop102 flink-local]$ bin/flink run -m hadoop102:8081 -c com.atguigu.cdc.FlinkCDC\_01\_DS ./gmall-flink-cdc.jar

* + - * 1. 观察taskManager日志，会从头读取表数据
        2. 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gmall2023\_config.t\_user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/flink/save/... -c com.atguigu.cdc.FlinkCDC\_01\_DS ./gmall-flink-cdc.jar

观察taskManager日志，会从检查点读取表数据

* 1. FlinkSQL方式的应用

代码实现

package com.atguigu;

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

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

public class FlinkCDC\_02\_SQL {

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

// TODO 1. 准备环境

// 1.1 流处理环境

StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();

env.setParallelism(1);

// 1.2 表执行环境

StreamTableEnvironment tableEnv = StreamTableEnvironment.create(env);

// TODO 2. 创建动态表

tableEnv.executeSql("CREATE TABLE user\_info (\n" +

"id INT,\n" +

"name STRING,\n" +

"age INT,\n" +

"primary key(id) not enforced\n" +

") WITH (" +

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

"'hostname' = 'hadoop102'," +

"'port' = '3306'," +

"'username' = 'root'," +

"'password' = '000000'," +

"'database-name' = 'gmall2023\_config'," +

"'table-name' = 'test'," +

"'scan.startup.mode' = 'earliest-offset'" +

")");

tableEnv.executeSql("select \* from user\_info").print();

// TODO 3. 执行任务

env.execute();

}

}