尚硅谷大数据技术之 Maxwell

(作者：尚硅谷大数据研发部)

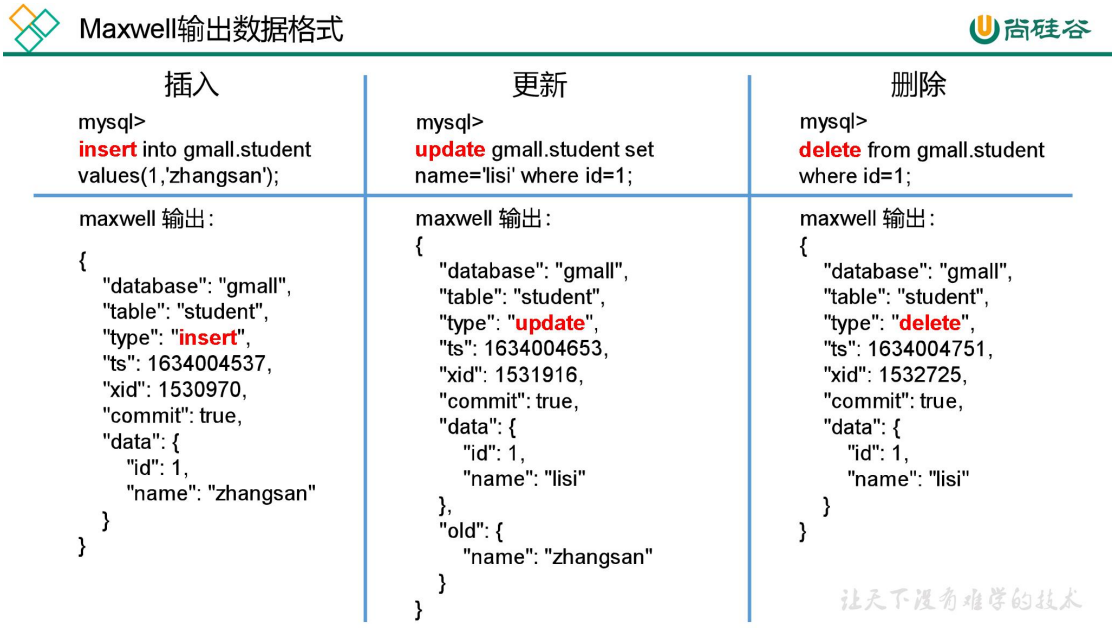
版本：V1.0

第 1 章 Maxwell 简介

1.1 Maxwell 概述

Maxwell 是由美国 Zendesk 公司开源，用 Java 编写的 MySQL 变更数据抓取软件。它会实时监控 Mysql 数据库的数据变更操作（包括 insert、update、delete），并将变更数据以 JSON 格式发送给 Kafka、Kinesi 等流数据处理平台。官网地址：http://maxwells-daemon.io/

1.2 Maxwell 输出数据格式



注：Maxwell 输出的 json 字段说明：

字段	解释
database	变更数据所属的数据库
table	表更数据所属的表
type	数据变更类型
ts	数据变更发生的时间
xid	事务 id
commit	事务提交标志，可用于重新组装事务
data	对于 insert 类型，表示插入的数据；对于 update 类型，标识修改之后的数据；对于 delete 类型，表示删除的数据
old	对于 update 类型，表示修改之前的数据，只包含变更字段

第 2 章 Maxwell 原理

Maxwell 的工作原理是实时读取 MySQL 数据库的二进制日志（Binlog），从中获取变更数据，再将变更数据以 JSON 格式发送至 Kafka 等流处理平台。

2.1 MySQL 二进制日志

二进制日志（Binlog）是 MySQL 服务端非常重要的一种日志，它会保存 MySQL 数据库的所有数据变更记录。Binlog 的主要作用包括主从复制和数据恢复。Maxwell 的工作原理和主从复制密切相关。

2.2 MySQL 主从复制

MySQL 的主从复制，就是用来建立一个和主数据库完全一样的数据库环境，这个数据库称为从数据库。

1）主从复制的应用场景如下：

（1）做数据库的热备：主数据库服务器故障后，可切换到从数据库继续工作。

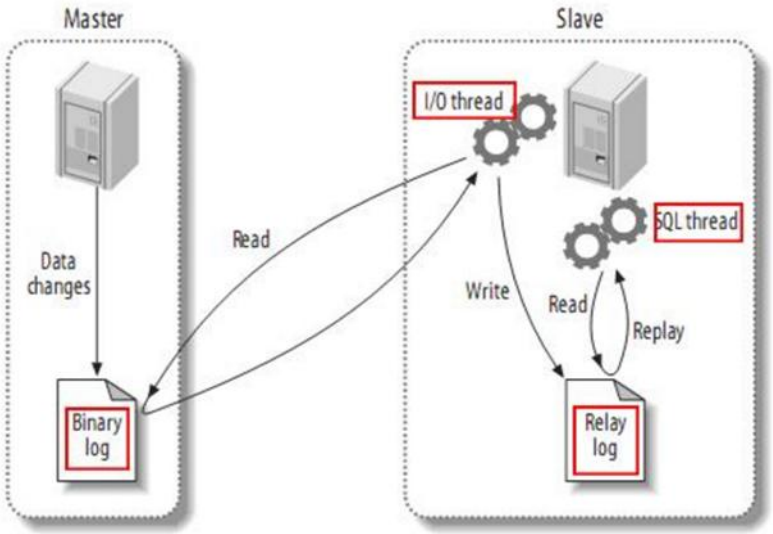
（2）读写分离：主数据库只负责业务数据的写入操作，而多个从数据库只负责业务数据的查询工作，在读多写少场景下，可以提高数据库工作效率。

2）主从复制的工作原理如下：

（1）Master 主库将数据变更记录，写到二进制日志(binary log)中

（2）Slave 从库向 mysql master 发送 dump 协议，将 master 主库的 binary log events 拷贝到它的中继日志(relay log)

（3）Slave 从库读取并回放中继日志中的事件，将改变的数据同步到自己的数据库。



2.3 Maxwell 原理

很简单，就是将自己伪装成 slave，并遵循 MySQL 主从复制的协议，从 master 同步数据。

第 3 章 Maxwell 部署

3.1 安装 Maxwell

1）下载安装包

（1）地址：https://github.com/zendesk/maxwell/releases/download/v1.29.2/maxwell-1.29.2.tar.gz

注：Maxwell-1.30.0 及以上版本不再支持 JDK1.8。

（2）将安装包上传到 hadoop102 节点的/opt/software 目录

注：此处使用教学版安装包，教学版对原版进行了改造，增加了自定义 Maxwell 输出数据中 ts 时间戳的参数，生产环境请使用原版。

2）将安装包解压至/opt/module

```
[atguigu@hadoop102 maxwell]$ tar -zxvf maxwell-1.29.2.tar.gz -C /opt/module/
```

3）修改名称

```
[atguigu@hadoop102 module]$ mv maxwell-1.29.2/ maxwell
```

3.2 配置 MySQL

3.2.1 启用 MySQL Binlog

MySQL 服务器的 Binlog 默认是未开启的，如需进行同步，需要先进行开启。

1）修改 MySQL 配置文件/etc/my.cnf

```
[atguigu@hadoop102 ~]$ sudo vim /etc/my.cnf
```

2）增加如下配置

```
[mysqld]

#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
```

注：MySQL Binlog 模式

Statement-based：基于语句，Binlog 会记录所有写操作的 SQL 语句，包括 insert、update、

delete 等。

优点： 节省空间

缺点： 有可能造成数据不一致，例如 insert 语句中包含 now()函数。

Row-based：基于行，Binlog 会记录每次写操作后被操作行记录的变化。

优点：保持数据的绝对一致性。

缺点：占用较大空间。

mixed：混合模式，默认是 Statement-based，如果 SQL 语句可能导致数据不一致，就自动切换到 Row-based。

Maxwell 要求 Binlog 采用 Row-based 模式。

3）重启 MySQL 服务

```
[atguigu@hadoop102 ~]$ sudo systemctl restart mysqld
```

3.2.2 创建 Maxwell 所需数据库和用户

Maxwell 需要在 MySQL 中存储其运行过程中的所需的一些数据，包括 binlog 同步的断点位置（Maxwell 支持断点续传）等等，故需要在 MySQL 为 Maxwell 创建数据库及用户。

1）创建数据库

```
msyql> CREATE DATABASE maxwell;
```

2）调整 MySQL 数据库密码级别

```
mysql> set global validate_password_policy=0;
mysql> set global validate_password_length=4;
```

3）创建 Maxwell 用户并赋予其必要权限

```
mysql> CREATE USER 'maxwell'@'%' IDENTIFIED BY 'maxwell';
mysql> GRANT ALL ON maxwell.* TO 'maxwell'@'%';
mysql> GRANT SELECT, REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'maxwell'@'%';
```

3.3 配置 Maxwell

1）修改 Maxwell 配置文件名称

```
[atguigu@hadoop102 maxwell]$ cd /opt/module/maxwell
[atguigu@hadoop102 maxwell]$ cp config.properties.example config.properties
```

2）修改 Maxwell 配置文件

```
[atguigu@hadoop102 maxwell]$ vim config.properties

#Maxwell 数据发送目的地，可选配置有 stdout|file|kafka|kinesis|pubsub|sqs|rabbitmq|redis
producer=kafka
#目标 Kafka 集群地址
kafka.bootstrap.servers=hadoop102:9092,hadoop103:9092,hadoop104:9092
```

```
#目标 Kafka topic，可静态配置，例如:maxwell，也可动态配置，例如：%{database}_%{table}
kafka_topic=maxwell

#MySQL 相关配置
host=hadoop102
user=maxwell
password=maxwell
jdbc_options=useSSL=false&serverTimezone=Asia/Shanghai
```

第 4 章 Maxwell 使用

4.1 启动 Kafka 集群

若 Maxwell 发送数据的目的地为 Kafka 集群，则需要先确保 Kafka 集群为启动状态。

4.2 Maxwell 启停

1）启动 Maxwell

```
[atguigu@hadoop102 ~]$ /opt/module/maxwell/bin/maxwell --config /opt/module/maxwell/config.properties --daemon
```

2）停止 Maxwell

```
[atguigu@hadoop102 ~]$ ps -ef | grep maxwell | grep -v grep | grep maxwell | awk '{print $2}' | xargs kill -9
```

3）Maxwell 启停脚本

（1）创建并编辑 Maxwell 启停脚本

```
[atguigu@hadoop102 bin]$ vim mxw.sh
```

（2）脚本内容如下

```
#!/bin/bash

MAXWELL_HOME=/opt/module/maxwell

status_maxwell(){
    result=`ps -ef | grep maxwell | grep -v grep | wc -l`
    return $result
}


start_maxwell(){
    status_maxwell
    if [[ $? -lt 1 ]]; then
        echo "启动 Maxwell"
        $MAXWELL_HOME/bin/maxwell --config $MAXWELL_HOME/config.properties --daemon
    else
        echo "Maxwell 正在运行"
    fi
```

```
}


stop_maxwell(){
    status_maxwell
    if [[ $? -gt 0 ]]; then
        echo "停止 Maxwell"
        ps -ef | grep maxwell | grep -v grep | awk '{print $2}' | xargs kill -9
    else
        echo "Maxwell 未在运行"
    fi
}


case $1 in
    start )
        start_maxwell
    ;;
    stop )
        stop_maxwell
    ;;
    restart )
       stop_maxwell
       start_maxwell
    ;;
esac
```

4.2 增量数据同步

1）启动 Kafka 消费者

```
[atguigu@hadoop102 kafka]$ bin/kafka-console-consumer.sh --bootstrap-server hadoop102:9092 --topic maxwell
```

2）模拟生成数据

```
[atguigu@hadoop102 db_log]$ java -jar gmall2020-mock-db-2021-01-22.jar
```

3）观察 Kafka 消费者

```
{"database":"gmall","table":"comment_info","type":"insert","ts":1634023510,"xid":1653373,"xoffset":11998,"data":{"id":1447825655672463369,"user_id":289,"nick_name":null,"head_img":null,"sku_id":11,"spu_id":3,"order_id":18440,"appraise":"1204","comment_txt":"评论内容：12897688728191593794966121429786132276125164551411","create_time":"2020-06-16 15:25:09","operate_time":null}}
{"database":"gmall","table":"comment_info","type":"insert","ts":1634023510,"xid":1653373,"xoffset":11999,"data":{"id":1447825655672463370,"user_id":774,"nick_name":null,"head_img":null,"sku_id":25,"spu_id":8,"order_id":18441,"appraise":"1204","comment_txt":"评论内容：67552221621263422568447438734865327666683661982185","create_time":"2020-06-16 15:25:09","operate_time":null}}
```

4.3 历史数据全量同步

上一节，我们已经实现了使用 Maxwell 实时增量同步 MySQL 变更数据的功能。但有时

只有增量数据是不够的，我们可能需要使用到 MySQL 数据库中从历史至今的一个完整的数据集。这就需要我们在进行增量同步之前，先进行一次历史数据的全量同步。这样就能保证得到一个完整的数据集。

4.3.1 Maxwell-bootstrap

Maxwell 提供了 bootstrap 功能来进行历史数据的全量同步，命令如下：

```
[atguigu@hadoop102 maxwell]$ /opt/module/maxwell/bin/maxwell-bootstrap --database gmall --table user_info --config /opt/module/maxwell/config.properties
```

4.3.2 boostrap 数据格式

采用 bootstrap 方式同步的输出数据格式如下：

```
{
    "database": "fooDB",
    "table": "barTable",
    "type": "bootstrap-start",
    "ts": 1450557744,
    "data": {}
}
{
    "database": "fooDB",
    "table": "barTable",
    "type": "bootstrap-insert",
    "ts": 1450557744,
    "data": {
        "txt": "hello"
    }
}
{
    "database": "fooDB",
    "table": "barTable",
    "type": "bootstrap-insert",
    "ts": 1450557744,
    "data": {
        "txt": "bootstrap!"
    }
}
{
    "database": "fooDB",
    "table": "barTable",
    "type": "bootstrap-complete",
    "ts": 1450557744,
    "data": {}
}
```

注意事项：

1）第一条 type 为 bootstrap-start 和最后一条 type 为 bootstrap-complete 的数据，是 bootstrap

开始和结束的标志，不包含数据，中间的 type 为 bootstrap-insert 的数据才包含数据。

2）一次 bootstrap 输出的所有记录的 ts 都相同，为 bootstrap 开始的时间。