ENGINEERING BLOG

MAXWELL

admin July 19, 2019 Leave a comment Edit

Change data capture (CDC) is a set of software design patterns used to determine (and track) the data that has changed so that action can be taken using the changed data. CDC is also an approach to data integration that is based on the identification, capture and delivery of the changes made to enterprise data sources.

CDC solutions occur most often in data-warehouse environments since capturing and preserving the state of data across time is one of the core functions of a data warehouse, but CDC can be utilised in any database or data repository system.

In a simplified CDC context, one computer system has data believed to have changed from a previous point in time, and a second computer system needs to take action based on that changed data. The former is the source, the latter is the target. It is possible that the source and target are the same system physically, but that would not change the design pattern logically.

USE CASE : Asynchronous inter-service communication

SETUP TRIAL AND TESTING :

- Invalid entries in MySQL table is evenly reproduced in maxwell.
- Maxwell replicator is single threaded; events are processed from binlog and pushed to Kafka one message at a time.
- Running Maxwell with –bootstrapper=sync, the same thread is used to do bootstrapping, meaning that all binlog events are blocked until bootstrapping is complete.
- Running Maxwell with –bootstrapper=async however, will make Maxwell spawn a separate thread for bootstrapping. In this async mode, non-bootstrapped tables are replicated as normal by the main thread, while the binlog events for bootstrapped

tables are queued and sent to the replication stream at the end of the bootstrap process.

TEST-1 :

- Maxwell-Down
- Kafka – Up
- Zookeeper – Up

MYSQL data is logged when maxwell is up

TEST-2 :

- Maxwell – Up
- Kafka – Down
- Zookeeper – Up

MYSQL data logged properly when maxwell restarted

TEST-3 :

- With ignore_producer_error = false
- Maxwell – Up
- Zookeeper – Up
- Kafka – Down
- MySQL – Up

No data loss when Kafka is up and maxwell restarted

TEST-4 :

- Maxwell – Down
- Kafka – Down

- Zookeeper – Up
- MySQL – Up

Tested with 3 lac records

- Records inserted into single table;
- Records inserted by importing csv
- Records inserted using script.
- Total failed logs (checked via metrics)=0

Testing with point/geometry data type, works perfectly with point/geometry data type; Testing with blob data type, works perfectly with blob data type

> Gives error for float8 data type entry in MySQL Reason: MySQL is smart enough to parse float8 as double, but maxwell is unable to do so. **Steps to make it work for float* data type**:

1. Check out maxwell code
2. Go to column_definitions.g4 in antlr4
3. Add float8 wherever required
4. Handle it in columnDef.java
5. Make package in terminal
6. Copy maxwell jar to required folder

> WORKING OF MAXWELL

1. Reading the maxwell config file :MaxwellConfig.java
2. Making MySQL connection :MaxwellMysqlConfig.java
3. Reading and updating schema in the `schema ` table
4. Monitoring
5. Kafka Producer **JSON from maxwell to kafka**

```
{
        "database": "NextServices",
        "table": "STUDENT",
        "type": "update",
        "ts": 1550470300,
        "xid": 52054 340,
        "commit": true,
        "data": {
                "STUDENT_ID": 10403420,
                "AADHAR_NUMBER": null,
                "ADMISSIO N_NO": "Baa10",
                "ADMISSION_STUDENT_ID": null,
                "ADMISSION_YEAR": null,
                "ATTENDANC E_CODE": "BBR164",
                "BIRTH_PLACE": null,
                "BRANCH_ID": 1369,
                "ACADEMIC_SESSION_ID": 1272,
                "CASTE_ID": null,
                "CATEGORY_ID": null,
                "CREATED_BY": 505443,
                "CREATED_ON": "20 18-09-28 21:46:15",
                "DATE_OF_BIRTH": "1986-02-03",
                "FIRST_LANG_ID": null,
                "FIRST_NAME": "Chaitri ",
                "GENDER": "Male",
                "IDENTIFICATION_ONE": null,
                "IMAGE_ID": null,
                "JOINING_DATE": "2018 -02-23 00:00:00",
                "LAST_NAME": "Y",
                "MIDDLE_NAME": "JAIN",
                "MODIFIED_BY": 505443,
                "MODIFIE D_ON": "2019-02-18 11:41:40",
                "MOTHER_TONGUE_ID": null,
                "NATIONALITY": null,
                "PRIMARY_CONTACT_ID": nu ll,
                "RELIGION_ID": null,
                "REMARKS": null,
                "SECOND_LANG_ID": null,
                "SUBCASTE_ID": null,
                "A DMISSION_CLASS_ID": null,
                "EVIDENCE_OF_DISABILITY": null,
                "SERIOUS_INCIDENT": null,
                "PREVIOUS_DISCIPLINARY_ISSUES": null,
                "USER_PROFILE_ID": 6147952,
                "STUDENT_U UID": "5efe8188-e63f-4919-b672-
9412c327b040",
                "DISPLAY_NAME": null
        },
        "old": {
                "MODIFIED _ON": "2019-02-18 10:47:24"
        }
}
```

PROCEDURE ADOPTED TO RUN MAXWELL ON DOCKERS :

1. Create maxwell configuration file in gitlab
2. Create Task definition with environment variables ***gitlabToken, giturl, maxwellConfigName***
3. In jenkins provide maven ***goal : clean install*** and the ***svn path : svn://192.168.10.7/iconcept/branches/coreInfra/Maxwell/Base/config/v1.22.0/maxw 1.22.0***
4. Create ECS Cluster, Service, ECR Repository

> Please configure main MAXWELL in t3 medium machine, micro can be risky

```
ERRORS AND SOLUTIONS :

ERROR :
Couldn't find table 'XXXXXXX' in database NextServices(mostly occur in local n

REASON :
We have allowed maxwell to read only on NextServices ,but it tries to scan tab

SOLUTION :
Since this table is not in NextServices, we don’t need to scan it, we can add
```

```
ERROR :
Unable to find partition leader for topic ‘XXXXXXX’

REASON :
It is not able to find leader for topic in kafka as atleast one amongst kafka

SOLUTION :
Make kafka node up
```

```
ERROR :
Overflowed in-memory buffer, spilling over into /tmp/maxwellXXXXevents (mostly

REASON :
RAM is not sufficient

SOLUTION :
Make space in memory (scale the machine) or add swap space
```

```
ERROR :
Packet size insufficient

REASON :
Large set of updates has done in single transaction and huge bin log file is g

SOLUTION :
Move the offset to next normal binlog file (get file number from DBA team) for
```

```
ERROR :
One amongst MySQL, Kafka is not up => maxwell will go down

SOLUTION :
Make dependant service up
```

```
ERROR :
Any reason which is not solvable/understandable due to schema.

SOLUTION :
Drop “schemas,databases,tables,columns” tables of maxwell DB. It will recreate
```

```
ERROR :
Any reason which are not solvable/understandable at particular positions for a

SOLUTION :
Set binlog offset to latest in `positions` table for that particular client or
```

> For restarting of maxwell , cron job is set which will run after every 10 mins and alert is sent via email

MONITORING MAXWELL

```
TELEGRAF --> PROMETHEUS --> GRAFANA
```

- GitHub link : https://github.com/zendesk/maxwell
- Official Docs : http://maxwells-daemon.io/
- SVN Link : svn://192.168.10.7/iconcept/branches/coreInfra/maxwell-New/new-maxwell
- SVN Maxwell Docker : svn://192.168.10.7/iconcept/branches/coreInfra/Maxwell

> **STEPS TO MOVE CUSTOMIZED DOCKER VERSION TO PRODUCTION**
>
> 1. Remove summary tables/ KK pipeline
> 2. Configure population of summary tables by a cron job that executes every 3 hours
> 3. Have two ecs clusters for maxwell(Medium) and maxwellSingleTopic(Micro)
> 4. Create properties in gitlab
> 5. Configure ecs task definition with maxwell config name and gitlabUrl
> 6. Create aws ecs services
> 7. Create jenkins jobs
> 8. Create ECR for maxwell image in production

> Authors : Pratik Asai, Peddi Rohith

Leave a comment

Comment

Post Comment