# scaled

Service to automate import / export of data between amazon-s3 and cassandra

Project Scale is built upon three packages

1. amazonS3 - Operations to perform with an S3 based storage

2. compress - Archive multiple files in a folder / directory to enable instant transmission of data

3. sstable - Create the required sstables from local data in a CSV file and transfer the received data into C\*

Setup Information:

Hardware:

Operating System - Mac OS X 10.0.3

Model Name - MacBook Pro

Processor Name - Intel Core i5

Processor Speed - 2.6 GHz

Number of Processors - 1

Total Number of Cores - 2

L2 Cache (per Core) - 256 KB

L3 Cache - 3 MB

RAM - 8 GB 1600 MHz

Secondary Memory - 120 GB SSD

Software:

Java - Java(TM) 1.8

Eclipse IDE - Luna

XCode IDE - 6.3.1

Cassandra - 2.1.4

Java packages information:

amazonS3:

Constants - Constants used globally in the package. [Will update to include all constants shortly]

DeleteS3 - Delete a bucket[s] or object[s] with key[s] from the S3 storage

DownloadS3 - Download the contents of a bucket[s] specified by the key

ListS3 - List the contents of S3 bucket[s]

UploadS3 - Upload a local file to S3 to the specified bucket using the specified key

compress:

Unzip - Create a folder locally from contents specified in the zip file, adhereing to directory structure[KeySpace/ColumnFamily]

zip - Archive the specified local folder to a zip file, with name specified in the program

sstable:

JmxBulkLoader - Loads local sstables specified Program Arguments of IDE to a live C\* Cluster

sstableImport - Create an sstable file set from a local CSV file as data input

Preparation:

Create keyspace and column family in C\* using CQLSH or cassandra-cli

Login to CQL in the node:

Execute cqlsh from terminal

When connected:

CREATE KEYSPACE userdata WITH replication = {'class': 'SimpleStrategy', 'replication\_factor': '3'} AND durable\_writes = true;

CREATE TABLE userdata.users (fname text PRIMARY KEY, age int, email text, lname text, password text, uuid int)

Exit shell

Execution steps:

1. Create an input data file

modify the "conf/dummy.csv" to input the data. Filename or Location maybe changed based on need

2. Create sstable files [\*.db, \*.index, \*.sha1, etc.]

From command line:

javac sstableImport.java [Create Java class file to run]

java sstableImport [Run the program to generate output]

From IDE:

Eclipse builds the files automatically. So, just hit run and the output is generated.

Output:

A new directory is created named "userdata" that represents the keyspace, containing the sstable files.

[Create a new folder named "users" inside "userdata" to adhere to "keyspace/columnfamily" convention and copy the files from userdata to users]

Zip:

A zip file is created if [userdata/users/[sstables]] exists. So if an error is thrown in the first run, its normal. Will fix it soon.

3. Amazon S3 Operations:

Assumptions:

Object Key - Change according to need. Currently it is "users".

Bucket - If you need a new bucket, let me know. Else the default is "scale-userdata"

Execution - Tested extensively on IDE only. So just hit run.

Upload / Download / List / Delete:

File to be uploaded is hard-coded in the program. Paths are relative. May change filename based on need but must change if zip file is different.

Just hit run from IDE.

4. Load the sstable files to C\*

Using sstableloader:

In terminal,

Navigate to the folder containing "keyspace/columnfamily/sstablefiles" after extraction from downloaded sstablefiles from S3,

Execute "sstableloader -d localhost userdata/users"

Note:

Cassandra needs to be running and the address needs to be localhost or local-IP-address of ring

sstableloader uploads files in the format "keyspace/columnfamily" to maintain the C\* DB

Using JMXBulkloader:

In IDE,

Add the "keyspace/columnfamily" as program arguments to Run Configuration

Save and execute

Note:

To debug the program, add the following to VM arguments in addition to program arguments above, to ensure consistency.

VM arguments:

-Dcom.sun.management.jmxremote

-Dcom.sun.management.jmxremote.port=9160

-Dcom.sun.management.jmxremote.authenticate=false

-Dcom.sun.management.jmxremote.ssl=false

Issues:

SSTABLELOADER:

sstableloader and JMXBulkLoader do not function as intended in local mode and throws the following error

"java.lang.IllegalArgumentException: Not enough bytes"

Error has been traced to "/cassandra/src/java/org/apache/cassandra/db/composites/AbstractCType.java" and "checkRemaining" method

Will fix soon

GC happens in unusual intervals and does not comply with configuration

AMAZON S3:

To comply with agreement to use S3, I should not reveal the ACCESS KEY online.

But would be happy to share if needed.

Work in progress:

Automation:

The above steps are being automated using Ruby and Shell script to enable commandline access with options

Architecture Flow:

An architectural flow diagram is created to show the program will execute as a service

GC Collection:

Better GC mechanism is in analysis

Dynamic Data:

Dynamic Data Streaming to C\* from S3/RDBMS/HDFS/NoSQL is being analyzed

Constants:

Better declaration and use of global constants is being analyzed

Scripts update:

Ruby / Shell scripts that perform the same actions as the Java programs is in analysis. Will upload a the Ruby version of service soon.

CQL:

Script to perform CRUD operations is being written to reduce time to create and manage keyspaces and column families, dynamically.