**|** Reviews Apps for Android

**|** README

**|** Karan Patel

**|** Eric Smith

**|** Venkata Bharat Varma Namburi

**|** Table of Contents

**|** Development

**|** Map Reduce Implementation

**|** Map Reduce Standalone Testing

**|** Database Design & Implementation

**|** Database Extraction to JSON

**|** Android Graphical User Interface Implementation

**|** Cluster Configuration

**|** Initial Configuration

**|** Initial Execution

**|** Reduced Instances Configuration

**|** Reduced Instances Execution

**|** Increased Instances Configuration

**|** Increased Instances Execution

**|** Modified Dataset Configuration

**|** Modified Dataset Execution

**|** Terminated Node Execution

**|** Development

**|** Map Reduce Implementation

The sample data used for the Map Reduce implementation was a subset of a dataset of product reviews and metadata from Amazon between 1996 – 2014. More specifically, the said subset of data pertains to application reviews for Android devices. The subset of data was provided by the University of California San Diego and can be obtained here: http://jmcauley.ucsd.edu/data/amazon/ (Apps for Android). This subset of data was selected because it was a considerably large sample set (752,967 total reviews for 87,271 unique users) that could effectively test the functionality of the configured cluster utilization.

The Map Reduce implementation was developed using Maven for build automation. Maven has the capability of bundling the dependencies an application may require at once. The following dependencies were used to successfully execute the Map Reduce jobs:

1. Hadoop Map Reduce 2.8.5: https://mvnrepository.com/artifact/org.apache.hadoop
2. Org.json (20180813): https://search.maven.org/artifact/org.json/json/20180813/bundle

Hadoop MapReduce is a software framework for writing applications which process excessive amounts of data in-parallel. Org.json is a package that implements JSON encoders / decoders in Java. The subset of data used for the Map Reduce implementation is of type JSON, so this package was also a required dependency for precise parsing. Additionally, JDK 11 was the Java version used for development.

There were three Map Reduce jobs configured. The first job determined the total number of unique reviews, which, to reiterate, was 752,967. The second job was to determine the total number of 5.0 (highest score possible) reviews that were accumulated (386,637). Finally, the third job was to determine the total breakdown of each review score (78,713 for 1.0, 44,385 for 2.0, 85,121 for 3.0, 158,081 for 4.0, and 386,637 for 5.0, as mentioned). We thought these three tests increased in complexity linearly and provided effective stress testing for our cluster. Figure 1 represents the Map and Reduce implementation for the third test.



*Figure 1*

**|** Development

**|** Map Reduce Implementation

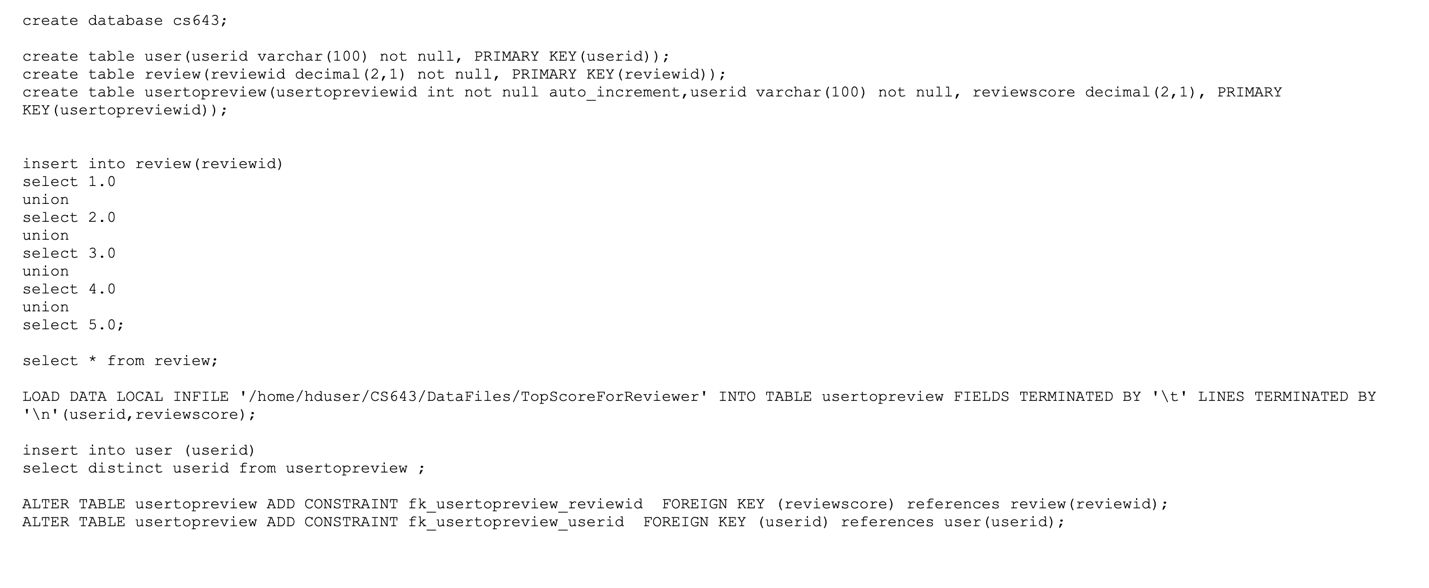
**|** Map Reduce Standalone Testing

The three aforementioned jobs were developed locally and tested on a single, standalone EC2 instance. The following reference was used for installing Hadoop in standalone mode on Ubuntu 18.04: https://www.digitalocean.com/community/tutorials/how-to-install-hadoop-in-stand-alone-mode-on-ubuntu-18-04. For consistency purposes, Hadoop Map Reduce 2.8.5, Org.json (20180813), and JDK 11 were used for installation.

**|** Development

**|** Database Design & Implementation

We determined that we wanted to provide a Graphical User Interface (GUI) for displaying our data. In order to achieve that, we determined that a database for storing and retrieving data would be most effective. The database would consist of three tables. The first table would store User information. The second table would store Review information. The third table would be the Relationship table between Users and Reviews. The database was created using MySQL (https://dev.mysql.com/downloads/mysql/). Figure 2 represents the script used to generate the database and the necessary constraints. In order to populate the database, each of the aforementioned Map Reduce jobs’ output were converted to .txt files. Figure 2 showcases the proper syntax to load data from a .txt file into a relational database.

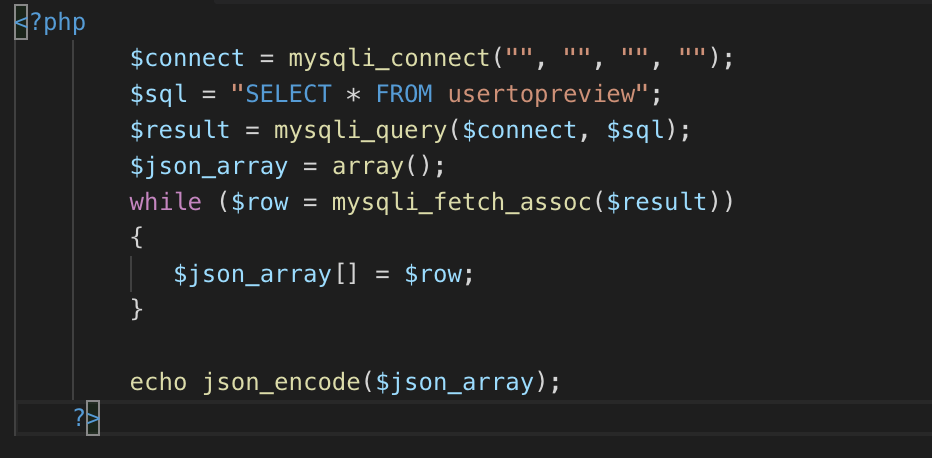


*Figure 2*

**|** Development

**|** Database Extraction to JSON

Once the database was created and functional, query extraction was required to create the GUI described earlier. The query extraction was done by writing a PHP script that translates a result set from a query to JSON. Figure 3 represents the said query extraction script.



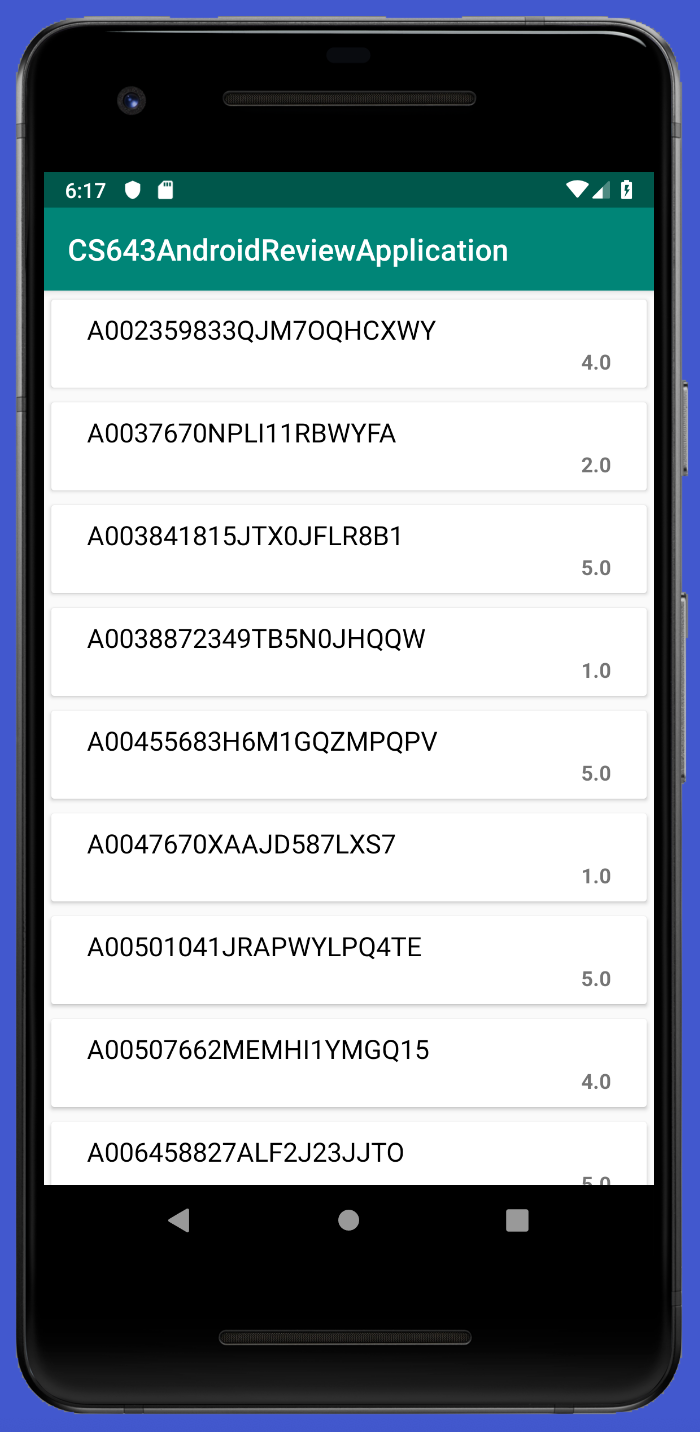
*Figure 3*

**|** Development

**|** Android Graphical User Interface Implementation

We thought it would be appropriate to use Android Studio to display our data on an Android Device Emulator with our own Android application, considering the working subset of data pertains to reviews for Android applications. To populate the said application, the JSON output generated from the query extraction procedure mentioned above was hosted in a configured on a locally deployed web server. In order to make effective calls to the said server, the application utilized the Volley (https://developer.android.com/training/volley

) dependency, which is an HTTP library used for Android networking. Figure 4 represents a snapshot view of the data retrieved (Id for each unique User and the highest score they provided for an Android application) from the server encapsulated in a RecyclerView on the Android Device Emulator.



*Figure 4*

**|** Cluster Configuration

**|** Initial Configuration

The cluster utilized was configured using Amazon Elastic Map Reduce (EMR). Amazon provides a managed Hadoop framework for convenient processing of data across dynamically scalable Amazon EC2 instances. The professor of the course allowed for us to provide instructions for how to set up the cluster using Amazon EMR. For reference, the official documentation provided b Amazon can be obtained here: https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-gs.html.

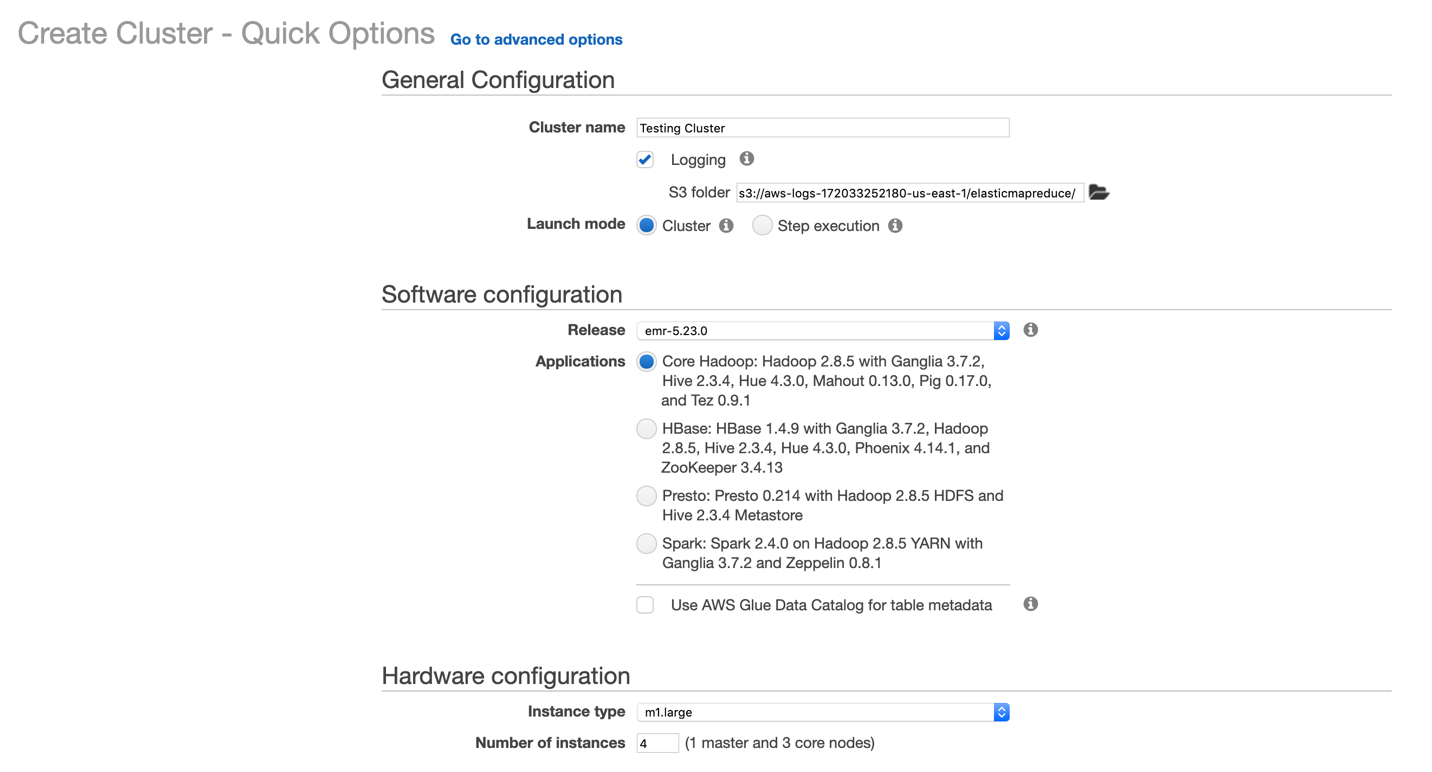
Step 1: Create an S3 bucket for a testing environment. Once the S3 bucket has been created, the subset of data as well as the JAR file that encapsulates all of the Map Reduce jobs can be found here:

Subset of data: https://s3.amazonaws.com/cs643-application-group9-testing-environment/reviews\_Apps\_for\_Android\_5.json

JAR file: <https://s3.amazonaws.com/cs643-application-group9-testing-environment/ApplicationProject-0.0.1-SNAPSHOT.jar>

Upload the two files to the created S3 bucket. Should the files be unavailable through the designated URL’s, they will be attached with this submission for convenience as well. If an Amazon EC2 Key Pair has not been created, please proceed with that. Otherwise, proceed with Step 2.

Step 2: Sign in to the AWS Management Console and open the Amazon EMR console (https://console.aws.amazon.com/elasticmapreduce/home?region=us-east-1). Select ‘Create Cluster.’ Figure 5 represents the initial configurations used for our cluster setup. Note: designate the ‘S3 folder’ input to a more appropriate personal directory, such as the S3 bucket created in Step 1. Moreover, for the ‘EC2 key pair’ input, assign your unique EC2 key pair.

**

*Figure 5*

**|** Cluster Configuration

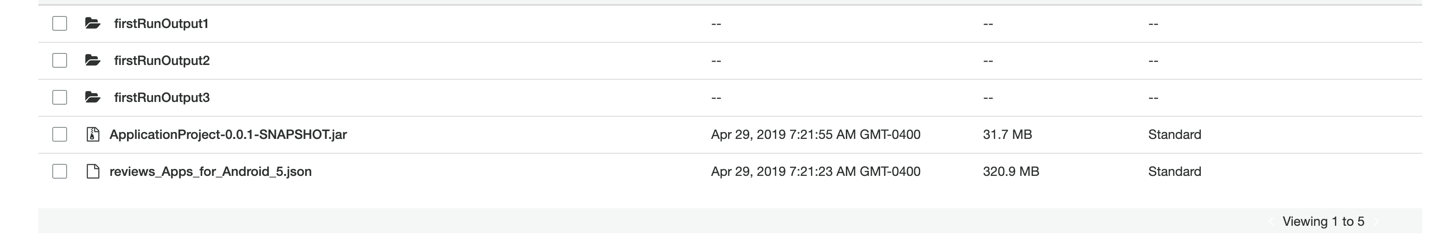
**|** Initial Configuration

**|** Initial Execution

The cluster will take several minutes to initialize. Once finished, navigate to the ‘Steps’ tab to verify that the ‘Setup hadoop debugging’ step has been successfully completed. Once verified, selected ‘Add Step’ from the ‘Steps’ tab. In the dialog prompt, provide the following input:

* Step type: Custom JAR
* Name\*: A unique name to identify step
* JAR Location: S3BucketCreated/ApplicationProject-0.0.1-SNAPSHOT.jar
* Arguments:
  + s3://S3BucketCreated/reviews\_Apps\_for\_Android\_5.json
  + s3://S3BucketCreated/firstRunOutput1
  + s3://S3BucketCreated/firstRunOutput2
  + s3://S3BucketCreated/firstRunOutput3
* Action on failure: Continue

Select ‘Add’ to initiate the step throughout the configured cluster. Once the step has been completed, the results will be uploaded to the S3 bucket created, as evident in Figure 6. The three Map Reduce jobs took 3.5 minutes to complete.



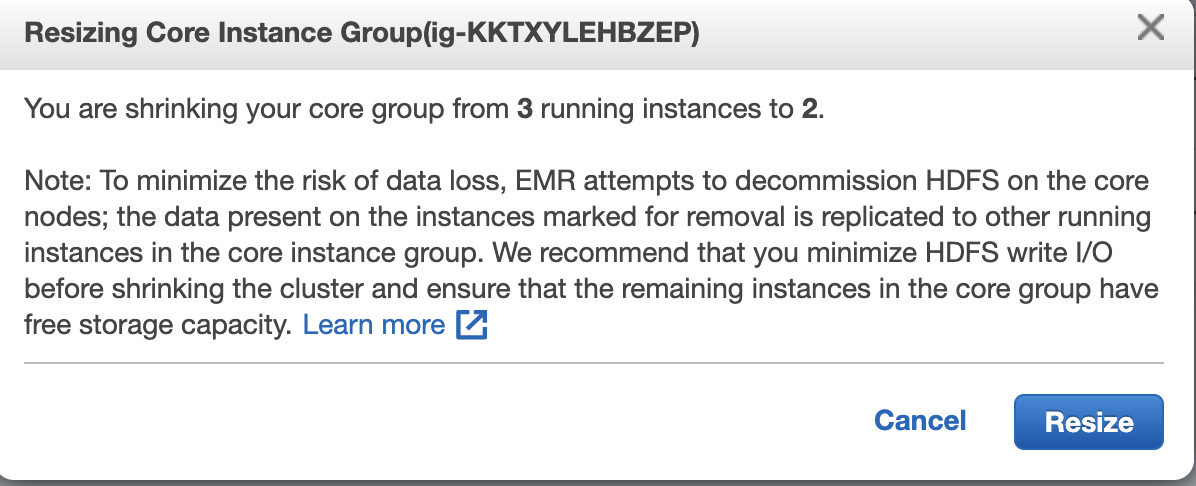
*Figure 6*

Navigating back to the ‘Steps’ tab of the EMR – AWS Console, the step is now classified as ‘Completed.’

**|** Cluster Configuration

**|** Reduced Instances Configuration

The three Map Reduce jobs will now be executed on a smaller-sized cluster. In the EMR – AWS Console, select the ‘Hardware’ tab and under the ‘Instance’ count’ column for the CORE nodes, select ‘Resize.’ Reduce the cluster size by 1, making the ‘Instance count’ parameter 2. Figure 7 represents the confirmation box for a reduced-sized cluster. Select ‘Resize’ to proceed.



*Figure 7*

**|** Cluster Configuration

**|** Reduced Instances Configuration

**|** Reduced Instances Execution

Once the cluster has finished resizing, navigate to the ‘Steps’ tab in the EMR – AWS Console and select ‘Add Step.’ In the dialog prompt, provide the following input:

* Step type: Custom JAR
* Name\*: A unique name to identify step
* JAR Location: S3BucketCreated/ApplicationProject-0.0.1-SNAPSHOT.jar
* Arguments:
  + s3://S3BucketCreated/reviews\_Apps\_for\_Android\_5.json
  + s3://S3BucketCreated/reducedRunOutput1
  + s3://S3BucketCreated/reducedRunOutput2
  + s3://S3BucketCreated/reducedRunOutput3
* Action on failure: Continue

Select ‘Add’ to initiate the step throughout the reduced-sized cluster. Once the step has been completed, the results will be uploaded to the S3 bucket created. The three Map Reduce jobs took 4.8 minutes to complete.

**|** Cluster Configuration

**|** Increased Instances Configuration

The three Map Reduce jobs will now be executed on a larger-sized cluster. In the EMR – AWS Console, select the ‘Hardware’ tab and under the ‘Instance’ count’ column for the CORE nodes, select ‘Resize.’ Increase the (already reduced) cluster size by 2, making the ‘Instance count’ parameter 4.

**|** Cluster Configuration

**|** Increased Instances Configuration

**|** Increased Instances Execution

Once the cluster has finished resizing, navigate to the ‘Steps’ tab in the EMR – AWS Console and select ‘Add Step.’ In the dialog prompt, provide the following input:

* Step type: Custom JAR
* Name\*: A unique name to identify step
* JAR Location: S3BucketCreated/ApplicationProject-0.0.1-SNAPSHOT.jar
* Arguments:
  + s3://S3BucketCreated/reviews\_Apps\_for\_Android\_5.json
  + s3://S3BucketCreated/increasedRunOutput1
  + s3://S3BucketCreated/increasedRunOutput2
  + s3://S3BucketCreated/increasedRunOutput3
* Action on failure: Continue

Select ‘Add’ to initiate the step throughout the reduced-sized cluster. Once the step has been completed, the results will be uploaded to the S3 bucket created. The three Map Reduce jobs took 2.99 minutes to complete.

**|** Cluster Configuration

**|** Modified Dataset Configuration

The three Map Reduce jobs will now be executed on a smaller data subset. The original size of the data subset was 752,967 lines. This reduced data subset has 326,833 lines, a roughly 50% decrease. Navigate to the S3 Management Console. Upload the following file to the working S3 bucket: <https://s3.amazonaws.com/cs643-application-group9-testing-environment/reviews2.json>. Similar to the preceding S3 URLS, a hard copy will be provided upon submission for convenience.

**|** Cluster Configuration

**|** Modified Dataset Configuration

**|** Modified Dataset Execution

Once the new data subset has been uploaded to the working S3 bucket, navigate to the ‘Steps’ tab in the EMR – AWS Console and select ‘Add Step.’ In the dialog prompt, provide the following input:

* Step type: Custom JAR
* Name\*: A unique name to identify step
* JAR Location: S3BucketCreated/ApplicationProject-0.0.1-SNAPSHOT.jar
* Arguments:
  + s3://S3BucketCreated/reviews2.json
  + s3://S3BucketCreated/smallFileRunOutput1
  + s3://S3BucketCreated/smallFileRunOutput2
  + s3://S3BucketCreated/smallFileRunOutput3
* Action on failure: Continue

Select ‘Add’ to initiate the step the cluster. Once the step has been completed, the results will be uploaded to the S3 bucket created. The three Map Reduce jobs took 2.60 minutes to complete on the smaller data subset.

**|** Cluster Configuration

**|** Terminated Node Execution

For the final test, a node will be ‘Stopped’ during a running step. Repeat the steps taken in Cluster Configuration | Modified Dataset Configuration | Modified Dataset Execution. While the step is running, navigate to the ‘Hardware’ tab and expand the Core group. Select an EC2 instance (of the 4 total) and a redirection to the Instances EC2 Management Console will occur. From here, right-click the EC2 instance and select ‘Stop’ to stop the instance while the step is running. Back in the EMR – AWS Console, select the ‘Events’ tab to identify the Cluster State Change, as evident in Figure 8. Once the step has been completed, the results will be uploaded to the S3 bucket created. The three Map Reduce jobs took 2.96 minutes to complete.



*Figure 8*

With this, the testing on the EMR cluster is complete. To properly close the cluster, select ‘Terminate’ on the EMR – AWS Console.