INFSCI 2750 - Mini Project 02

Yichi Zhang (yiz141@pitt.edu)
Quan Zhou (quz3@pitt.edu)

Part 1: Setting up Spark

In the previous project, we've already finished setting up the Hadoop cluster. In order to set up a Spark environment on top the Hadoop cluster, the following steps were taken:

Add new environment variables to the `~/.bashrc` file.

```
# Add Spark installation path to $PATH
export PATH="/usr/lib/jvm/java-8-oracle:/usr/local/spark/bin":$PATH
# Add three new environment variable
export HADOOP_CONF_DIR=/usr/local/hadoop/etc/hadoop
export SPARK_HOME=/usr/local/spark
export LD_LIBRARY_PATH=/usr/local/hadoop/lib/native:$LD_LIBRARY_PATH
```

Download and install the pre-built version of Spark and install it on the cluster.

```
cd /usr/local/
wget http://mirrors.gigenet.com/apache/spark/spark-2.3.0/spark-2.3.0-bin-hadoop2.7.tgz
tar -xzf spark-2.3.0-bin-hadoop2.7.tgz
ln -s spark-2.3.0-bin-hadoop2.7/ spark
```

Edit the configuration file `$SPARK_HOME/conf/spark-defaults.conf` for Spark

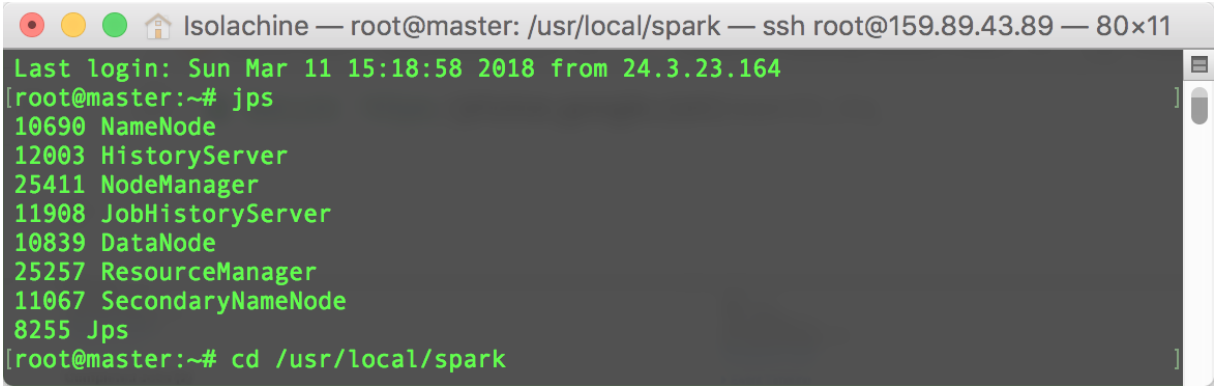
```
spark.master                      yarn
spark.driver.memory               1g
spark.yarn.am.memory              1g
spark.executor.memory             1g
spark.eventLog.enabled            true
spark.eventLog.dir                hdfs://master:9000/spark-logs
spark.history.provider            org.apache.spark.deploy.history.FsHistoryProvider
spark.history.fs.logDirectory     hdfs://master:9000/spark-logs
spark.history.fs.update.interval  10s
spark.history.ui.port             18080
spark.executor.cores              2
```

With all the preparation steps done, we can start the Hadoop cluster and Spark History Server by commands:

```
$HADOOP_HOME/sbin/start-all.sh
$HADOOP_HOME/bin/mapred --daemon start historyserver
```

```
$SPARK_HOME/sbin/start-history-server.sh
```

We can track all the Hadoop related processes with the `jps` command.



> Screenshot of running processes related to Hadoop

All the services on our cluster are running properly. We can monitor the cluster's status and performance through the following webpages.

Monitor	Link
HDFS NameNode	http://159.89.43.89:9870
ResourceManager of Yarn	http://159.89.43.89:8088
Spark History Server	http://159.89.43.89:18080
Hadoop Job History Server	http://159.89.43.89:19888

To start a Spark Shell on the cluster, simply run:

```
$SPARK_HOME/bin/spark-shell --master yarn --deploy-mode client
```

> Screenshot of Spark Shell running on VM

```
lsolachine — root@master: /usr/local/spark — ssh root@159.89.43.89 — 80×10
scala> val textFile = spark.read.textFile("input/hadoop-env.sh")
textFile: org.apache.spark.sql.Dataset[String] = [value: string]

scala> textFile.count()
res3: Long = 414

scala> textFile.first()
res4: String = export JAVA_HOME=/usr/lib/jvm/java-8-oracle

scala>
```

> Simple test run in Spark Shell

The whole project's source code was written in JAVA and used Maven as dependency management and build tool.
The JAVA code and `pom.xml` file for the project is in the `source_code` folder. The `mini-project-02-1.0.jar` was built locally with maven to include all the source code provided and was uploaded to the VM server for running.

```
mvn package
scp target/mini-project-02-1.0.jar root@159.89.43.89:/usr/local/spark
```

To track the log of Spark program run on the cluster, we added edited `yarn-site.xml` make logs all output to the HDFS in a aggregated manner.

```
    <property>
        <name>yarn.log-aggregation-enable</name>
        <value>true</value>
    </property>
```

So one can simply print the aggregated `stdout` logs with the following command:

```
$HADOOP_HOME/bin/yarn logs -log_files stdout -applicationId <AppID>
```

Part 2 : Last FM Data Analysis

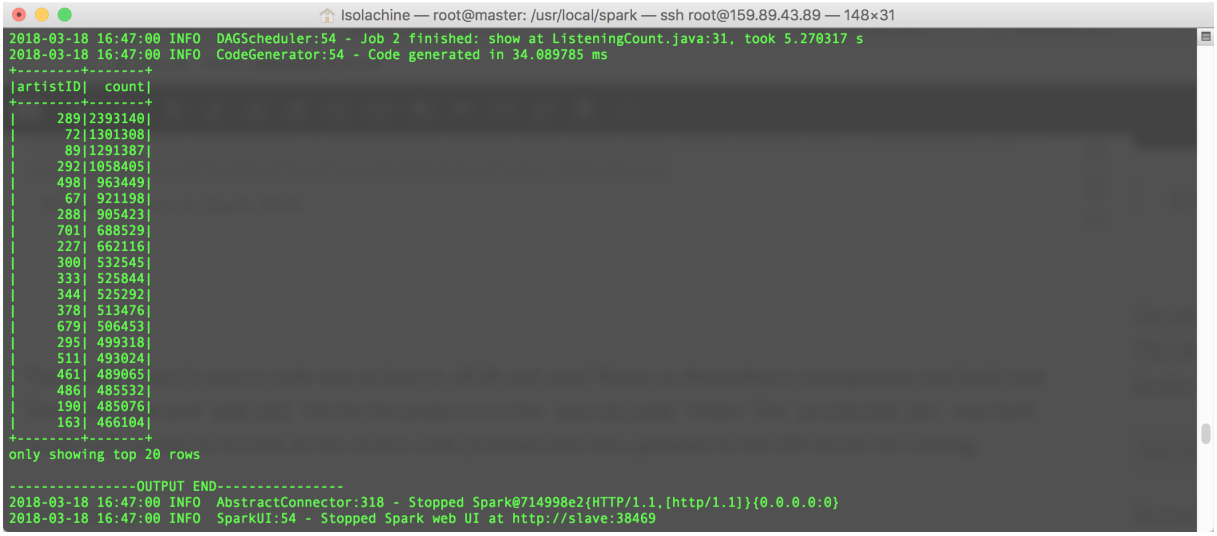
The `ListeningCount.java` file is the source code for counting total listening counts of each artist.
The program can be launched by:

```
$SPARK_HOME/bin/spark-submit \
```

```
 --deploy-mode cluster \
 --class ListeningCount \
 mini-project-02-1.0.jar
```

The logs can be viewed by:

```
$HADOOP_HOME/bin/yarn logs -log_files stdout -applicationId application_1521391514647_0001
```



> Screenshot of Top 20 Listened Artists and Count

The job summary can be viewed at:

```
http://159.89.43.89:18080/history/application_1521391514647_0001/1/jobs/
```

Part 3: Access Log Analysis

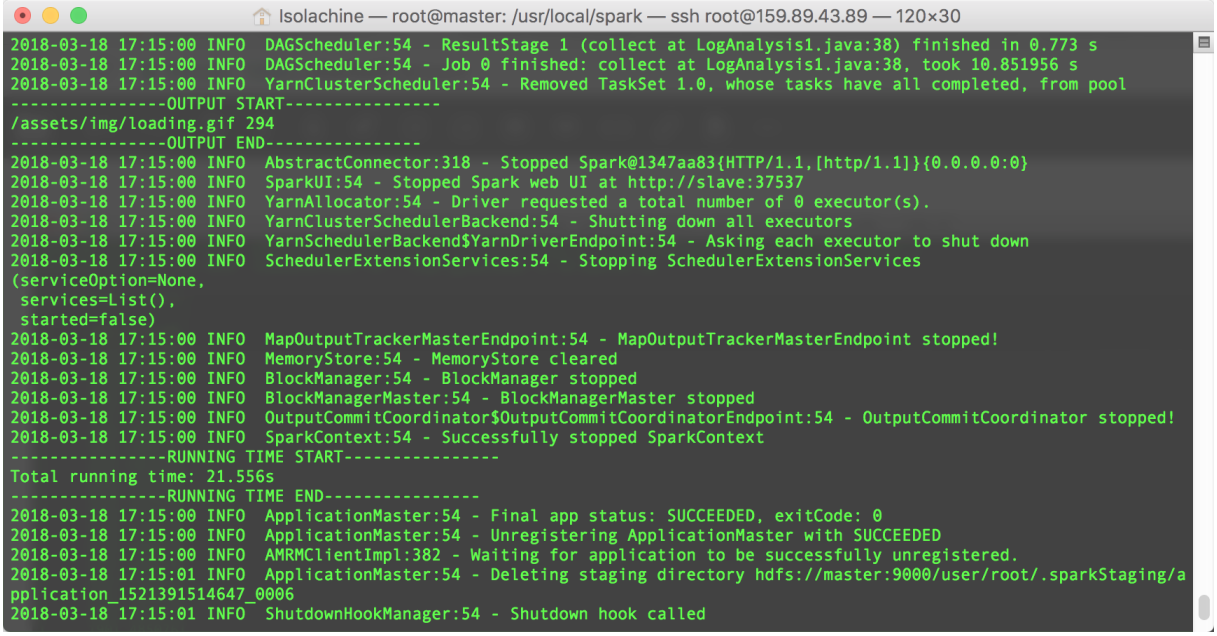
Problem 1

The `LogAnalysis1.java` file is the source code for problem 1.
The program can be launched by:

```
$SPARK_HOME/bin/spark-submit \
 --deploy-mode cluster \
 --class LogAnalysis1 \
 mini-project-02-1.0.jar
```

The logs can be viewed by:

```
$HADOOP_HOME/bin/yarn logs -log_files stdout -applicationId application_1521391514647_0006
```



> Screenshot of Results and Running Time

As the screen shot shows, `/assets/img/loading.gif` was accessed 294 times.
The job summary can be viewed at:

```
http://159.89.43.89:18080/history/application_1521391514647_0006/1/jobs/
```

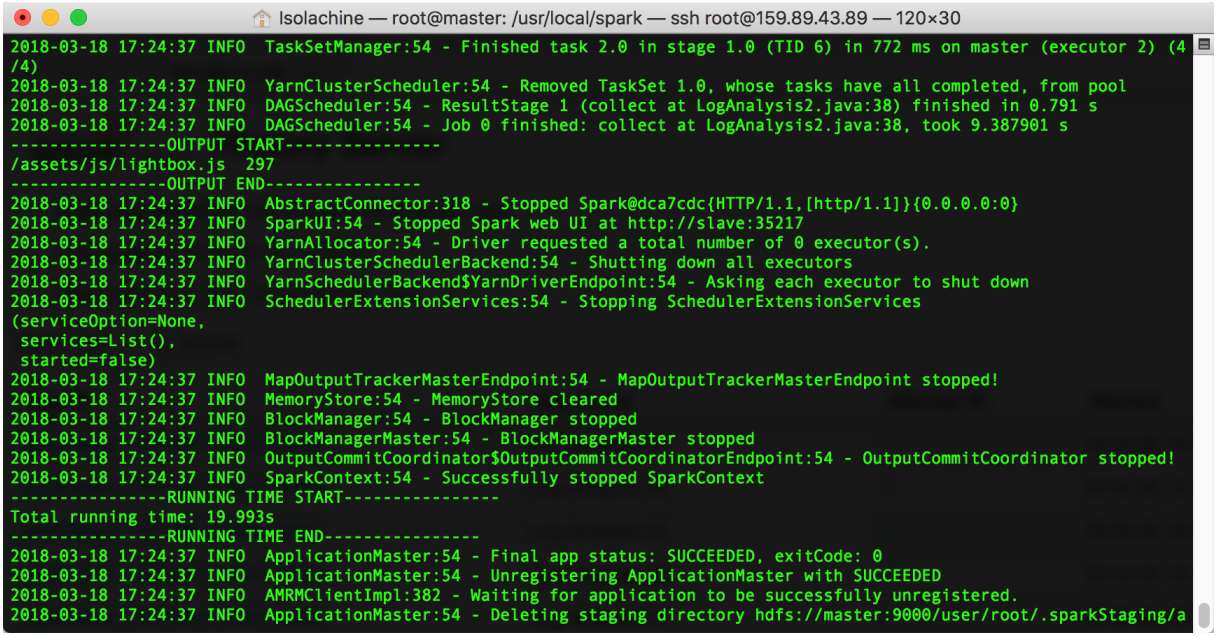
Problem 2

The `LogAnalysis2.java` file is the source code for problem 2.
The program can be launched by:

```
$SPARK_HOME/bin/spark-submit \
 --deploy-mode cluster \
 --class LogAnalysis2 \
 mini-project-02-1.0.jar
```

The logs can be viewed by:

```
$HADOOP_HOME/bin/yarn logs -log_files stdout -applicationId application_1521391514647_0009
```



> Screenshot of Results and Running Time

As the screen shot shows, `/assets/js/lightbox.js` was accessed 297 times.
The job summary can be viewed at:

```
http://159.89.43.89:18080/history/application_1521391514647_0009/1/jobs/
```

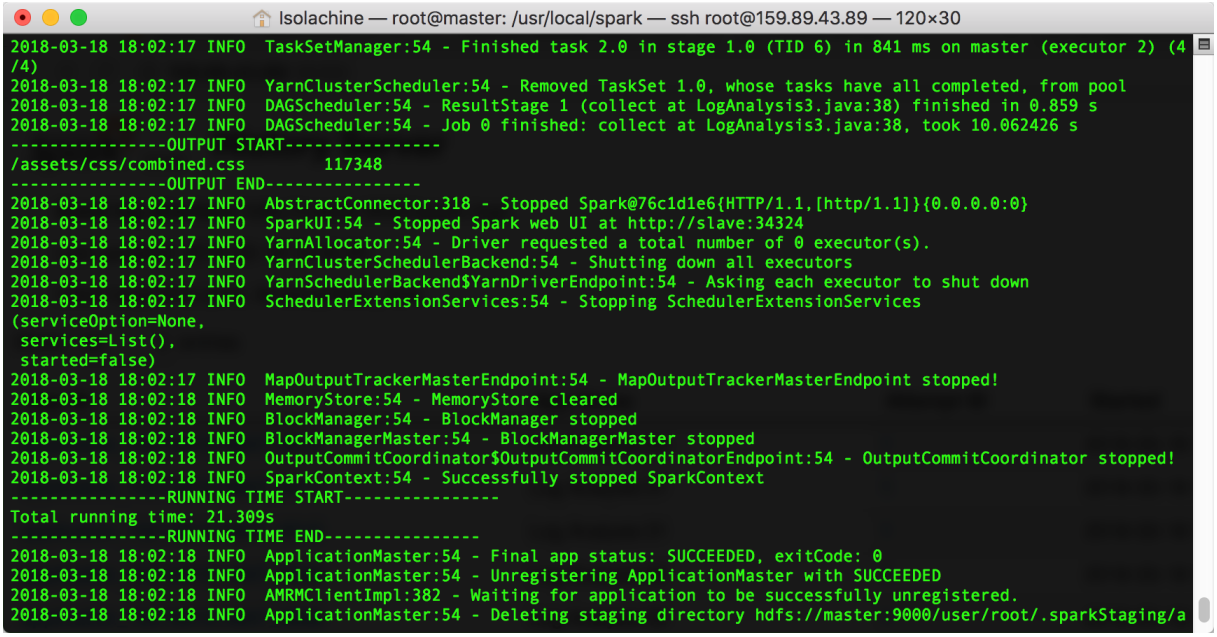
Problem 3

The `LogAnalysis3.java` file is the source code for problem 3.
The program can be launched by:

```
$SPARK_HOME/bin/spark-submit \
 --deploy-mode cluster \
 --class LogAnalysis3 \
 mini-project-02-1.0.jar
```

The logs can be viewed by:

```
$HADOOP_HOME/bin/yarn logs -log_files stdout -applicationId application_1521391514647_0012
```



```
lsolachine — root@master: /usr/local/spark — ssh root@159.89.43.89 — 120×30
2018-03-18 18:02:17 INFO  TaskSetManager:54 - Finished task 2.0 in stage 1.0 (TID 6) in 841 ms on master (executor 2) (4
/4)
2018-03-18 18:02:17 INFO  YarnClusterScheduler:54 - Removed TaskSet 1.0, whose tasks have all completed, from pool
2018-03-18 18:02:17 INFO  DAGScheduler:54 - ResultStage 1 (collect at LogAnalysis3.java:38) finished in 0.859 s
2018-03-18 18:02:17 INFO  DAGScheduler:54 - Job 0 finished: collect at LogAnalysis3.java:38, took 10.062426 s
----------------OUTPUT START----------------
/assets/css/combined.css        117348
----------------OUTPUT END----------------
2018-03-18 18:02:17 INFO  AbstractConnector:318 - Stopped Spark@76c1d1e6{HTTP/1.1,[http/1.1]}{0.0.0.0:0}
2018-03-18 18:02:17 INFO  SparkUI:54 - Stopped Spark web UI at http://slave:34324
2018-03-18 18:02:17 INFO  YarnAllocator:54 - Driver requested a total number of 0 executor(s).
2018-03-18 18:02:17 INFO  YarnClusterSchedulerBackend:54 - Shutting down all executors
2018-03-18 18:02:17 INFO  YarnSchedulerBackend$YarnDriverEndpoint:54 - Asking each executor to shut down
2018-03-18 18:02:17 INFO  SchedulerExtensionServices:54 - Stopping SchedulerExtensionServices
(serviceOption=None,
 services=List(),
 started=false)
2018-03-18 18:02:17 INFO  MapOutputTrackerMasterEndpoint:54 - MapOutputTrackerMasterEndpoint stopped!
2018-03-18 18:02:18 INFO  MemoryStore:54 - MemoryStore cleared
2018-03-18 18:02:18 INFO  BlockManager:54 - BlockManager stopped
2018-03-18 18:02:18 INFO  BlockManagerMaster:54 - BlockManagerMaster stopped
2018-03-18 18:02:18 INFO  OutputCommitCoordinator$OutputCommitCoordinatorEndpoint:54 - OutputCommitCoordinator stopped!
2018-03-18 18:02:18 INFO  SparkContext:54 - Successfully stopped SparkContext
----------------RUNNING TIME START----------------
Total running time: 21.309s
----------------RUNNING TIME END----------------
2018-03-18 18:02:18 INFO  ApplicationMaster:54 - Final app status: SUCCEEDED, exitCode: 0
2018-03-18 18:02:18 INFO  ApplicationMaster:54 - Unregistering ApplicationMaster with SUCCEEDED
2018-03-18 18:02:18 INFO  AMRMClientImpl:382 - Waiting for application to be successfully unregistered.
2018-03-18 18:02:18 INFO  ApplicationMaster:54 - Deleting staging directory hdfs://master:9000/user/root/.sparkStaging/a
```

> Screenshot of Results and Running Time

As the screen shot shows, `/assets/css/combined.css` was the most accessed resource, with 117348 times.
The job summary can be viewed at:

```
http://159.89.43.89:18080/history/application_1521391514647_0012/1/jobs/
```

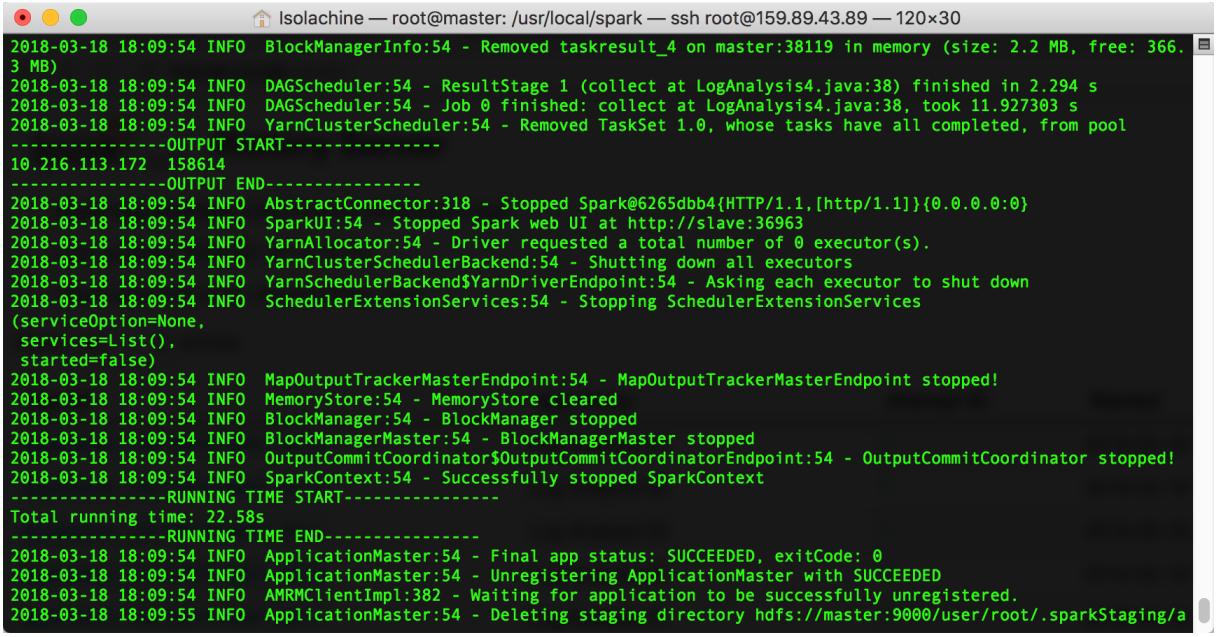
Problem 4

The `LogAnalysis4.java` file is the source code for problem 4.
The program can be launched by:

```
$SPARK_HOME/bin/spark-submit \
 --deploy-mode cluster \
 --class LogAnalysis4 \
 mini-project-02-1.0.jar
```

The logs can be viewed by:

```
$HADOOP_HOME/bin/yarn logs -log_files stdout -applicationId application_1521391514647_0015
```



> Screenshot of Results and Running Time

As the screen shot shows, `10.216.113.172` was the IP which accessed the website most frequently, with 158614 times.
The job summary can be viewed at:

```
http://159.89.43.89:18080/history/application_1521391514647_0015/1/jobs/
```

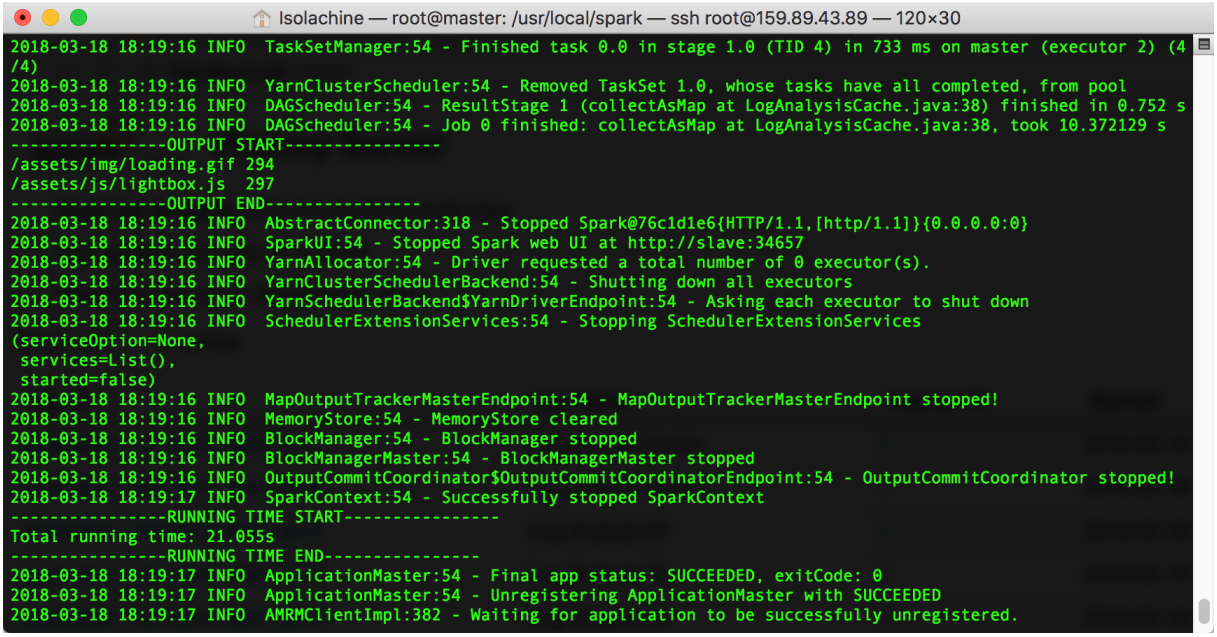
RDD Cache Used

The `LogAnalysisCache.java` file is the source code for problem with RDD cache.
The program can be launched by:

```
$SPARK_HOME/bin/spark-submit \
 --deploy-mode cluster \
 --class LogAnalysisCache \
 mini-project-02-1.0.jar
```

The logs can be viewed by:

```
$HADOOP_HOME/bin/yarn logs -log_files stdout -applicationId application_1521391514647_0017
```



```
lsolachine — root@master: /usr/local/spark — ssh root@159.89.43.89 — 120×30
2018-03-18 18:19:16 INFO  TaskSetManager:54 - Finished task 0.0 in stage 1.0 (TID 4) in 733 ms on master (executor 2) (4
/4)
2018-03-18 18:19:16 INFO  YarnClusterScheduler:54 - Removed TaskSet 1.0, whose tasks have all completed, from pool
2018-03-18 18:19:16 INFO  DAGScheduler:54 - ResultStage 1 (collectAsMap at LogAnalysisCache.java:38) finished in 0.752 s
2018-03-18 18:19:16 INFO  DAGScheduler:54 - Job 0 finished: collectAsMap at LogAnalysisCache.java:38, took 10.372129 s
----------------OUTPUT START----------------
/assets/img/loading.gif 294
/assets/js/lightbox.js  297
----------------OUTPUT END----------------
2018-03-18 18:19:16 INFO  AbstractConnector:318 - Stopped Spark@76c1d1e6{HTTP/1.1,[http/1.1]}{0.0.0.0:0}
2018-03-18 18:19:16 INFO  SparkUI:54 - Stopped Spark web UI at http://slave:34657
2018-03-18 18:19:16 INFO  YarnAllocator:54 - Driver requested a total number of 0 executor(s).
2018-03-18 18:19:16 INFO  YarnClusterSchedulerBackend:54 - Shutting down all executors
2018-03-18 18:19:16 INFO  YarnSchedulerBackend$YarnDriverEndpoint:54 - Asking each executor to shut down
2018-03-18 18:19:16 INFO  SchedulerExtensionServices:54 - Stopping SchedulerExtensionServices
(serviceOption=None,
 services=List(),
 started=false)
2018-03-18 18:19:16 INFO  MapOutputTrackerMasterEndpoint:54 - MapOutputTrackerMasterEndpoint stopped!
2018-03-18 18:19:16 INFO  MemoryStore:54 - MemoryStore cleared
2018-03-18 18:19:16 INFO  BlockManager:54 - BlockManager stopped
2018-03-18 18:19:16 INFO  BlockManagerMaster:54 - BlockManagerMaster stopped
2018-03-18 18:19:16 INFO  OutputCommitCoordinator$OutputCommitCoordinatorEndpoint:54 - OutputCommitCoordinator stopped!
2018-03-18 18:19:17 INFO  SparkContext:54 - Successfully stopped SparkContext
----------------RUNNING TIME START----------------
Total running time: 21.055s
----------------RUNNING TIME END----------------
2018-03-18 18:19:17 INFO  ApplicationMaster:54 - Final app status: SUCCEEDED, exitCode: 0
2018-03-18 18:19:17 INFO  ApplicationMaster:54 - Unregistering ApplicationMaster with SUCCEEDED
2018-03-18 18:19:17 INFO  AMRMClientImpl:382 - Waiting for application to be successfully unregistered.
```

> Screenshot of Results and Running Time

As the screen shot shows, when using RDD cache as intermediate results for further computation, the running time was 21.055 seconds.
The job summary can be viewed at:

```
http://159.89.43.89:18080/history/application_1521391514647_0017/1/jobs/
```

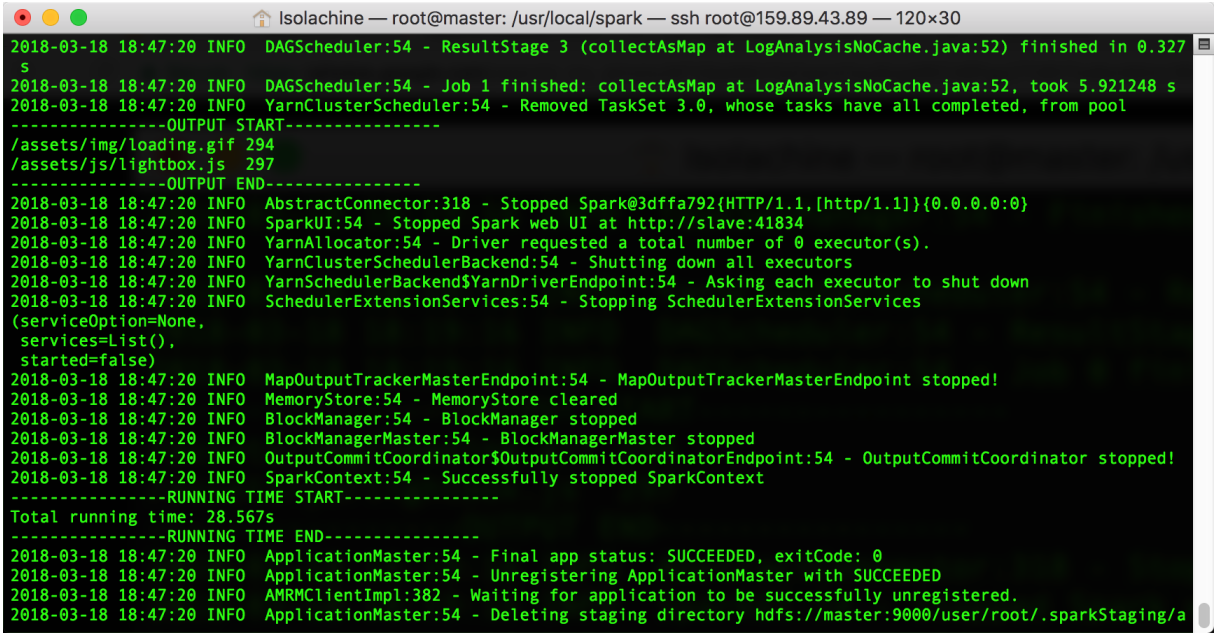
No Cache Used

The `LogAnalysisNoCache.java` file is the source code for problem with no cache used.
The program can be launched by:

```
$SPARK_HOME/bin/spark-submit \
 --deploy-mode cluster \
 --class LogAnalysisNoCache \
 mini-project-02-1.0.jar
```

The logs can be viewed by:

```
$HADOOP_HOME/bin/yarn logs -log_files stdout -applicationId application_1521391514647_0028
```



> Screenshot of Results and Running Time

As the screen shot shows, when using RDD cache as intermediate results for further computation, the running time was 28.567 seconds.
The job summary can be viewed at:

```
http://159.89.43.89:18080/history/application_1521391514647_0028/1/jobs/
```

RDD Cache Efficiency Analysis

From the previous sections, we were able to get the total running time of the program generating access counts on two different resource links. Both the mechanism of running with RDD cache and running without RDD cache was used.
The program with RDD cache was able to finish with 21.055 seconds, while the one without cache took 28.567 seconds. Caching made the whole program 25% faster than on demand loading.
We were able to see how time was consumed on each sub-task from the job summary web interface.

159.89.43.89:18080/history/application_1521391514647_0017/1/jobs/

Spark 2.3.0 — Log Analysis Cache application UI

Jobs | Stages | Storage | Environment | Executors

Spark Jobs (?)

User: root
Total Uptime: 21 s
Scheduling Mode: FIFO
Completed Jobs: 1

Event Timeline

Completed Jobs (1)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0	collectAsMap at LogAnalysisCache.java:38 collectAsMap at LogAnalysisCache.java:38	2018/03/18 18:19:06	10 s	2/2	8/8

> Job Summary of Program with RDD Cache

Despite the total running time of 21 seconds, the actual text processing time was about 10 seconds for caching the file and output the desired results.

159.89.43.89:18080/history/application_1521391514647_0028/1/jobs/

Spark 2.3.0 — Log Analysis No Cache application UI

Jobs | Stages | Storage | Environment | Executors

Spark Jobs (?)

User: root
Total Uptime: 28 s
Scheduling Mode: FIFO
Completed Jobs: 2

Event Timeline

Completed Jobs (2)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
1	collectAsMap at LogAnalysisNoCache.java:52 collectAsMap at LogAnalysisNoCache.java:52	2018/03/18 18:47:14	6 s	2/2	8/8
0	collectAsMap at LogAnalysisNoCache.java:37 collectAsMap at LogAnalysisNoCache.java:37	2018/03/18 18:47:02	11 s	2/2	8/8

> Job Summary of Program with No Cache Used

Despite the total running time of 28 seconds, the actual text processing time was about 11 seconds and 6 seconds for the two separate process.

This comparison shows caching has great advantage when dealing with data that can be used repeatedly. Although the second loading time is shorter than the first one (maybe due to loading mechanism of HDFS?), it could still save a lot of time with cache on a repetition of two jobs using the same intermediate result.

If the task was to do ad hoc search like database queries rather than ones we did above, caching would further save much more time with a load once, use many mechanism. This can be easily achieved by using the Spark Shell and Scala language to stream tasks.