

Summary

Map Reduce

The screenshot shows the AWS S3 console with the following details:

- Owner:** awslabs07026440t1705124199
- AWS Region:** US West (Oregon) us-west-2
- Last modified:** March 3, 2024, 14:57:05 (UTC+05:30)
- Size:** 99.6 KB
- Type:** csv
- S3 URI:** s3://plane-delay/input/DelayedFlights-updated.csv
- Amazon Resource Name (ARN):** arnaws:s3:::plane-delay/input/DelayedFlights-updated.csv
- Entity tag (Etag):** 5554fa56512fc5cc7e9cde4e52a6038
- Object URL:** https://plane-delay.s3.us-west-2.amazonaws.com/input/DelayedFlights-updated.csv

The screenshot shows the AWS EMR console with the following details:

- Your cluster "Plane Delay Hadoop V2" has been successfully created.**
- Cluster ID:** j-1885A2CE8X9UW
- Cluster configuration:** Instance groups
- Capacity:** 1 Primary | 1 Core | 1 Task
- Applications:** Amazon EMR version emr-7.0.0
- Installed applications:** Hadoop 3.3.6, Hive 3.1.3, Hue 4.11.0, Pig 0.17.0, Tez 0.10.2
- Cluster management:**
 - Log destination in Amazon S3: aws-logs-767397860534-us-west-2/elastictemapreduce
 - Persistent application UIs: YARN timeline server
 - Tez UI
 - Primary node public DNS: ec2-34-222-69-91.us-west-2.compute.amazonaws.com
 - Connect to the Primary node using SSH
 - Connect to the Primary node using SSM
- Status and time:**
 - Status: Waiting
 - Creation time: March 04, 2024, 13:38 (UTC+05:30)
 - Elapsed time: 7 minutes, 28 seconds

```
Last login: Mon Mar  4 13:54:39 on ttys001
upuladikario@Upul-M3 ~ % ssh -i ~/Documents/MSc/upul-test-key.pem hadoop@ec2-34-222-69-91.us-west-2.compute.amazonaws.com

      #_
~\_ #####          Amazon Linux 2023
~~ \#####\
~~  \###|
~~   \#/ ___  https://aws.amazon.com/linux/amazon-linux-2023
~~   V~' '-->
~~_/
~~_.-_-/
~/_-/
/_m/' 

Last login: Mon Mar  4 08:30:49 2024

EEEEEEEEEEEEEE MMMMMMMM           MMMMMMM RRRRRRRRRRRRRR
E:::::::::::E M:::::M       M:::::M R:::::::::::R
EE:::::EEEEEEE:::E M:::::M       M:::::M R:::::RRRRRR:::::R
E:::::E       EEEE M:::::M       M:::::M RR:::R     R:::::R
E:::::E       M:::::M:::::M     M:::::M:::::M R:::R     R:::::R
E:::::EEEEEEEEE M:::::M M::::M M:::::M M:::::M R:::RRRRR:::::R
E:::::::::::E M:::::M M::::M::::M M:::::M R:::::::::::RR
E:::::EEEEEEEEE M:::::M M:::::M M:::::M M:::::M R:::RRRRR:::::R
E:::::E       M:::::M M::::M M:::::M R:::::R     R:::::R
E:::::E       EEEE M:::::M     MMM M:::::M R:::::R     R:::::R
EE:::::EEEEEEE:::E M:::::M       M:::::M R:::::R     R:::::R
E:::::::::::E M:::::M       M:::::M RR:::::R     R:::::R
EEEEEEEEEEEEEEEEE MMMMMMM           MMMMMMM RRRRRRRR
```

```

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j2.properties Async: false
hive> CREATE EXTERNAL TABLE FlightDelays (rowNo INT,year STRING,month STRING,dayOfMonth STRING,dayOfWeek STRING,depTime STRING,crsDepTi
me STRING,arrTime STRING,crsArrTime STRING,unCarrier STRING,flightNo STRING,tailNo STRING,actualElapsedTime INT,crsElapsedTime INT,airT
ime INT,arrDelay INT,depDelay INT,origin STRING,dest STRING,distance INT,taxin INT,taxout INT,cancelled STRING,cancellationCode STRING
,diverted STRING,carrierDelay INT,weatherDelay INT,nasDelay INT,securityDelay INT,lateAirCraftDelay INT)ROW FORMAT DELIMITED FIELDS TER
MINATED BY ',' LOCATION 's3://plane-delay/input' TBLPROPERTIES ("skip.header.line.count""1");
OK
Time taken: 4.849 seconds

```

```

hive> SELECT year,avg((carrierDelay/arrDelay)*100)from FlightDelays GROUP BY year ORDER BY year;
Query ID = hadoop_20240304083238_7a735697-0aa5-4449-8b8b-5803751a1f68
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1709540004539_0001)

```

VERTICES	MODE	STATUS	TOTAL	COMPLETED	RUNNING	PENDING	FAILED	KILLED
Map 1	container	INITIALIZING	-1	0	0	-1	0	0
VERTICES	MODE	STATUS	TOTAL	COMPLETED	RUNNING	PENDING	FAILED	KILLED
Map 1	container	INITIALIZING	-1	0	0	-1	0	0
VERTICES	MODE	STATUS	TOTAL	COMPLETED	RUNNING	PENDING	FAILED	KILLED
Map 1	container	INITIALIZING	-1	0	0	-1	0	0
VERTICES	MODE	STATUS	TOTAL	COMPLETED	RUNNING	PENDING	FAILED	KILLED
Map 1	container	INITIALIZING	-1	0	0	-1	0	0

Map 1	container	SUCCEEDED	1	1	0	0	0	0
	Reducer 2	container	SUCCEEDED	2	2	0	0	0
Reducer 3	container	SUCCEEDED	1	1	0	0	0	0
VERTICES: 03/03 [=====>>>] 100% ELAPSED TIME: 11.72 s								
OK								
2003	24.557549755575373							
2004	43.64459443230066							
2005	28.01977637202288							
2006	30.453296261292596							
2007	19.850007017971283							
2008	28.88346981456985							
2009	28.33058554239575							
2010	21.89310246015957							

```

hive> SELECT year,avg((nasDelay/arrDelay)*100)from FlightDelays GROUP BY year ORDER BY year;
Query ID = hadoop_20240304083350_6a09f1e3-30db-4406-b73c-8b3ba75dca67
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1709540004539_0001)

```

VERTICES	MODE	STATUS	TOTAL	COMPLETED	RUNNING	PENDING	FAILED	KILLED
Map 1	container	INITIALIZING	-1	0	0	-1	0	0
VERTICES	MODE	STATUS	TOTAL	COMPLETED	RUNNING	PENDING	FAILED	KILLED
Map 1	container	INITED	1	0	0	1	0	0
VERTICES	MODE	STATUS	TOTAL	COMPLETED	RUNNING	PENDING	FAILED	KILLED
Map 1	container	SUCCEEDED	1	1	0	0	0	0
	Reducer 2	container	SUCCEEDED	2	2	0	0	0
Reducer 3	container	SUCCEEDED	1	1	0	0	0	0
VERTICES: 03/03 [=====>>>] 100% ELAPSED TIME: 6.23 s								
OK								
2003	29.686276314267346							
2004	18.24570061769958							
2005	16.63868805373129							
2006	18.119312329937703							
2007	30.625925917941924							
2008	30.16552562594132							
2009	37.63093330628511							
2010	33.87351363404217							

Time taken: 6.49 seconds, Fetched: 8 row(s)

```

hive> SELECT year,avg((weatherDelay/arrDelay)*100)from FlightDelays GROUP BY year ORDER BY year;
Query ID = hadoop_20240304083449_461d2645-4df1-4db2-9e54-070817e6f66a
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1709540004539_0001)

```

VERTICES	MODE	STATUS	TOTAL	COMPLETED	RUNNING	PENDING	FAILED	KILLED
Map 1	container	INITIALIZING	-1	0	0	-1	0	0
VERTICES	MODE	STATUS	TOTAL	COMPLETED	RUNNING	PENDING	FAILED	KILLED
Map 1	container	INITED	1	0	0	1	0	0
VERTICES	MODE	STATUS	TOTAL	COMPLETED	RUNNING	PENDING	FAILED	KILLED
Map 1	container	INITED	1	0	0	1	0	0
VERTICES	MODE	STATUS	TOTAL	COMPLETED	RUNNING	PENDING	FAILED	KILLED
Map 1	container	INITED	1	0	0	1	0	0

Map 1	container	SUCCEEDED	1	1	0	0	0	0
	Reducer 2	container	SUCCEEDED	0	2	0	2	0
Reducer 3	container	SUCCEEDED	1	1	0	0	0	0
VERTICES: 03/03 [=====>>>] 100% ELAPSED TIME: 6.46 s								
OK								
2003	7.831947966451205							
2004	6.4475279976916555							
2005	5.85869715149616							
2006	4.588604183967953							
2007	4.042975783210287							
2008	3.7254490054008955							
2009	0.4531661513792363							
2010	2.9023312955584664							
Time taken: 6.698 seconds, Fetched: 8 row(s)								

```

hive> SELECT year,avg((lateAirCraftDelay/arrDelay)*100)from FlightDelays GROUP BY year ORDER BY year;
Query ID = hadoop_20240304083540_0261c367-8161-4d0b-a126-5009933a993e
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1709540004539_0001)

```

VERTICES	MODE	STATUS	TOTAL	COMPLETED	RUNNING	PENDING	FAILED	KILLED
Map 1	container	INITIALIZING	-1	0	0	-1	0	0
VERTICES	MODE	STATUS	TOTAL	COMPLETED	RUNNING	PENDING	FAILED	KILLED
Map 1	container	INITED	1	0	0	1	0	0
VERTICES	MODE	STATUS	TOTAL	COMPLETED	RUNNING	PENDING	FAILED	KILLED
Map 1	container	INITED	1	0	0	1	0	0
VERTICES	MODE	STATUS	TOTAL	COMPLETED	RUNNING	PENDING	FAILED	KILLED
Map 1	container	INITED	1	0	0	1	0	0

Map 1	container	SUCCEEDED	1	1	0	0	0	0
	Reducer 2	container	SUCCEEDED	0	2	0	2	0
Reducer 3	container	SUCCEEDED	1	1	0	0	0	0
VERTICES: 03/03 [=====>>>] 100% ELAPSED TIME: 6.97 s								
OK								
2003	37.924225963706164							
2004	31.662176952308105							
2005	49.490838422749654							
2006	46.838787224801735							
2007	45.252432744291134							
2008	37.22555555488794							
2009	33.585314999939314							
2010	41.331052610239794							
Time taken: 7.194 seconds, Fetched: 8 row(s)								

Spark

Amazon EMR > EMR on EC2: Clusters > Plane Delay Spark V3

Plane Delay Spark V3

Updated less than a minute ago

[Terminate](#)

[Clone in AWS CLI](#)

[Clone](#)

▼ Summary

Cluster info

Cluster ID
j-1MPSIW6RFAN9

Cluster configuration
Instance groups

Capacity
1 Primary 1 Core 1 Task

Applications

Amazon EMR version
emr-7.0.0

Installed applications
Hadoop 3.3.6, Hive 3.1.3,
JupyterEnterpriseGateway 2.6.0, Livy 0.7.1,
Spark 3.5.0

Cluster management

Log destination in Amazon S3
aws-logs-767397860534-us-west-2/elasticmapreduce

Persistent application UIs
[Spark History Server](#)
[YARN timeline server](#)
[Tez UI](#)

Status and time

Status
 Waiting

Creation time
March 04, 2024, 14:14 (UTC+05:30)

Elapsed time
5 minutes, 3 seconds

Primary node public DNS
 ec2-52-37-116-5.us-west-2.compute.amazonaws.com

[Connect to the Primary node using SSH](#)
[Connect to the Primary node using SSM](#)

```
[hadoop@ip-172-31-5-140 ~]$ spark-shell
[Mar 04, 2024 8:51:13 AM org.apache.spark.launcher.Log4jHotPatchOption staticJavaAgentOption
WARNING: spark.log4jHotPatch.enabled is set to true, but /usr/share/log4j-cve-2021-44228-hotpatch/jdk17/Log4jHotPatchFat.jar does not exist at
the configured location

Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
24/03/04 08:51:23 WARN Client: Neither spark.yarn.jars nor spark.yarn.archive is set, falling back to uploading libraries under SPARK_HOME.
24/03/04 08:51:38 WARN YarnSchedulerBackend$YarnSchedulerEndpoint: Attempted to request executors before the AM has registered!
Spark context Web UI available at http://ip-172-31-5-140.us-west-2.compute.internal:4040
Spark context available as 'sc' (master = yarn, app id = application_1709542095932_0001).
Spark session available as 'spark'.
Welcome to

    /---/---/---/---/---/
   / \ / \ - \ / - \ / \ / \
  /---/ .--/ \--/ / \ / \ \ \
     /_/
version 3.5.0-amzn-0
```

```
scala> import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.SparkSession

scala> val spark = SparkSession.builder.appName("PlaneDelayApp").getOrCreate()
24/03/04 08:51:47 WARN SparkSession: Using an existing Spark session; only runtime SQL configurations will take effect.
spark: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@516cb8e1

scala> val df = spark.read.option("header", "true").option("inferSchema", "true").csv("s3://plane-delay/input/DelayedFlights-updated.csv")
df: org.apache.spark.sql.DataFrame = [c0: int, Year: int ... 28 more fields]

scala> df.createOrReplaceTempView("plane_delay")
24/03/04 08:52:25 WARN SparkStringUtils: Truncated the string representation of a plan since it was too large. This behavior can be adjusted by
y setting 'spark.sql.debug.maxToStringFields'.
```

```
scala> val resultCarrierDelay = spark.sql("SELECT Year, avg((CarrierDelay / ArrDelay)*100) from plane_delay GROUP BY Year")
resultCarrierDelay: org.apache.spark.sql.DataFrame = [Year: int, avg((CarrierDelay / ArrDelay) * 100)): double]

scala> spark.time(resultCarrierDelay .show())
+-----+
|Year|avg((CarrierDelay / ArrDelay) * 100))|
+----+-----+
|2003|      24.557549755575373|
|2004|      43.64459443230066|
|2005|      28.01977637202288|
|2006|      30.453296261292596|
|2007|      19.850007017971283|
|2008|      28.88346981456985|
|2009|      28.33058554239575|
|2010|      21.89310246015987|
+----+-----+
```

```

scala> val resultNASDelay = spark.sql("SELECT Year, avg((NASDelay /ArrDelay)*100) from plane_delay GROUP BY Year")
resultNASDelay: org.apache.spark.sql.DataFrame = [Year: int, avg((NASDelay / ArrDelay) * 100)): double]

scala> spark.time(resultNASDelay .show())
+---+
|Year|avg(((NASDelay / ArrDelay) * 100))|
+---+
|2003|          29.686276314267346|
|2004|          18.24570061769958|
|2005|          16.63868865373129|
|2006|          18.119312329937703|
|2007|          30.625925917941924|
|2008|          30.16552562594132|
|2009|          37.63093330628511|
|2010|          33.87351363484217|
+---+-----+

```

Time taken: 419 ms

```

scala> val resultWeatherDelay = spark.sql("SELECT Year, avg((WeatherDelay /ArrDelay)*100) from plane_delay GROUP BY Year")
resultWeatherDelay: org.apache.spark.sql.DataFrame = [Year: int, avg((WeatherDelay / ArrDelay) * 100)): double]

scala> spark.time(resultWeatherDelay .show())
+---+
|Year|avg(((WeatherDelay / ArrDelay) * 100))|
+---+
|2003|          7.8319479664511285|
|2004|          6.4475279976916555|
|2005|          5.85869715149616|
|2006|          4.588604183967953|
|2007|          4.042975783210287|
|2008|          3.7254490054008955|
|2009|          0.45316615137982363|
|2010|          2.9023312955584664|
+---+-----+

```

Time taken: 1759 ms

```

scala> val resultLateAircraftDelay = spark.sql("SELECT Year, avg((LateAircraftDelay /ArrDelay)*100) from plane_delay GROUP BY Year")
resultLateAircraftDelay: org.apache.spark.sql.DataFrame = [Year: int, avg((LateAircraftDelay / ArrDelay) * 100)): double]

scala> spark.time(resultLateAircraftDelay .show())
+---+
|Year|avg(((LateAircraftDelay / ArrDelay) * 100))|
+---+
|2003|          37.924225963786164|
|2004|          31.662176952308105|
|2005|          49.490838422749654|
|2006|          46.838787224881735|
|2007|          45.252432744291134|
|2008|          37.2255555408794|
|2009|          33.58531499939314|
|2010|          41.331052610239794|
+---+-----+

```

Time taken: 514 ms

```

scala> val resultSecurityDelay = spark.sql("SELECT Year, avg((SecurityDelay /ArrDelay)*100) from plane_delay GROUP BY Year")
resultSecurityDelay: org.apache.spark.sql.DataFrame = [Year: int, avg((SecurityDelay / ArrDelay) * 100)): double]

scala> spark.time(resultSecurityDelay .show())
+---+
|Year|avg(((SecurityDelay / ArrDelay) * 100))|
+---+
|2003|          0.0|
|2004|          0.0|
|2005|          0.0|
|2006|          0.0|
|2007|          0.22865853658536586|
|2008|          0.0|
|2009|          0.0|
|2010|          0.0|
+---+-----+

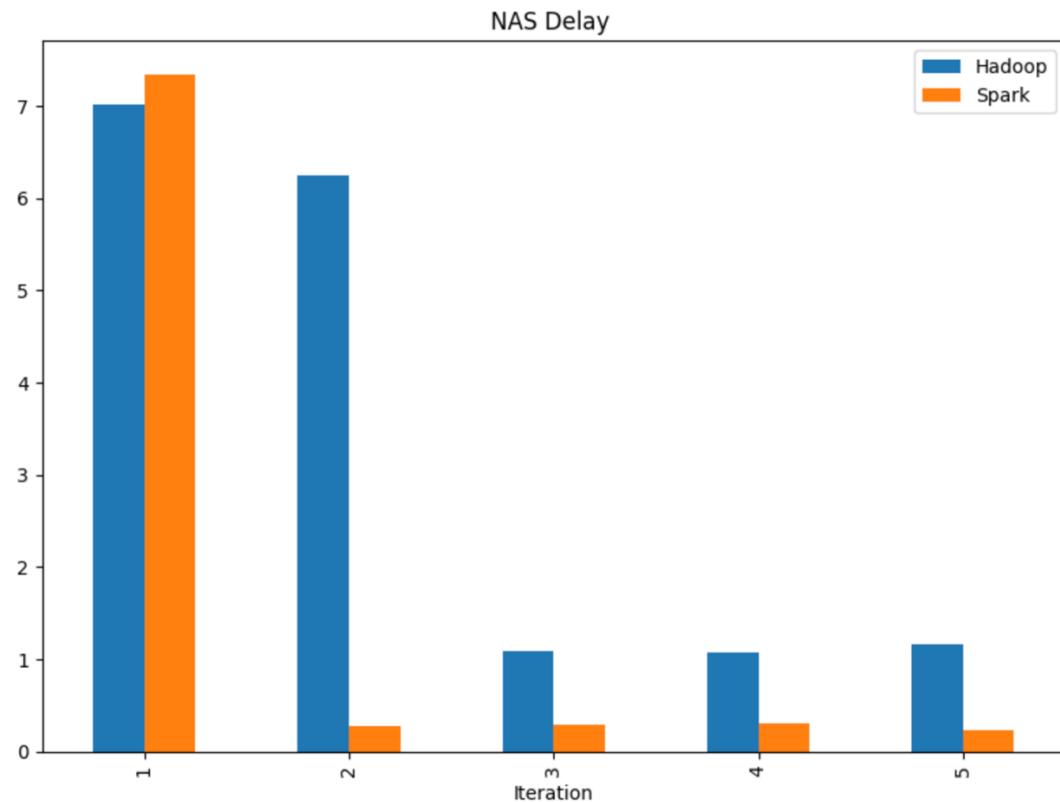
```

Time taken: 332 ms

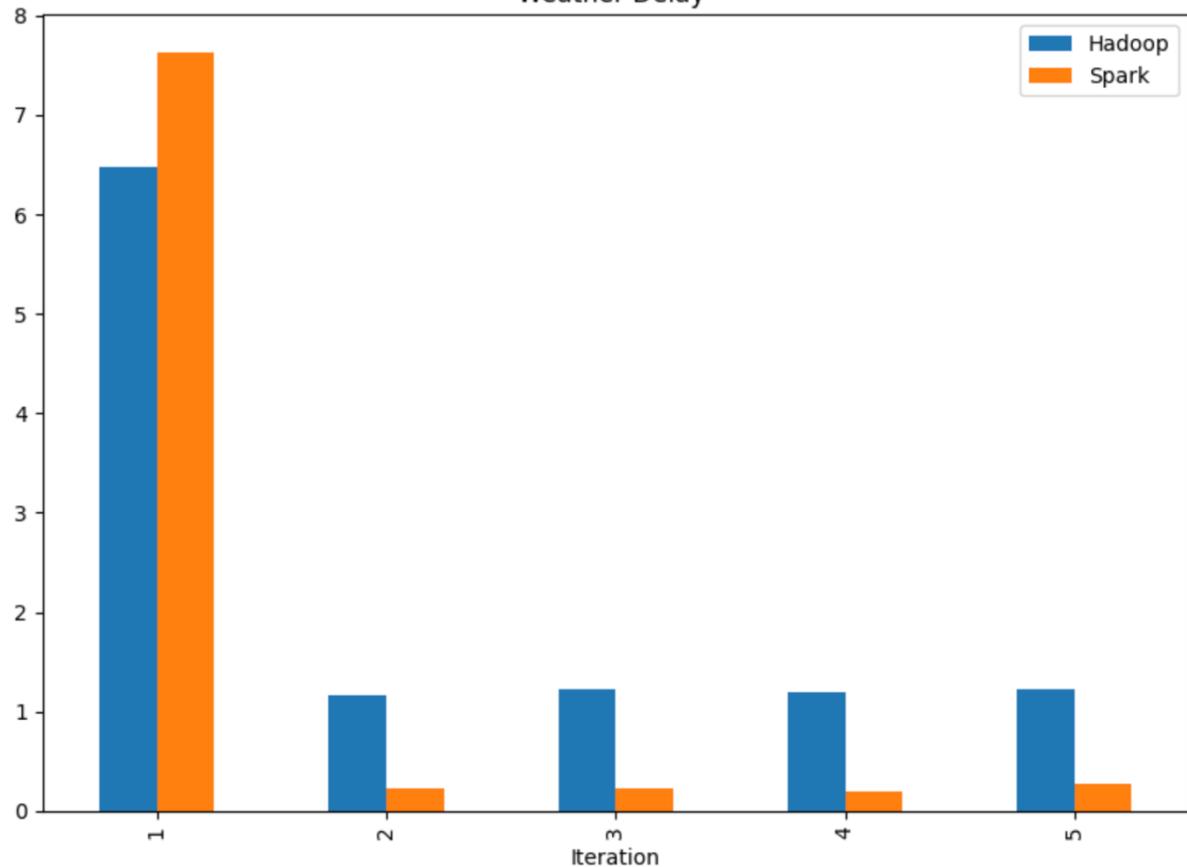
Results

One Iteration Summary (5th Iteration)

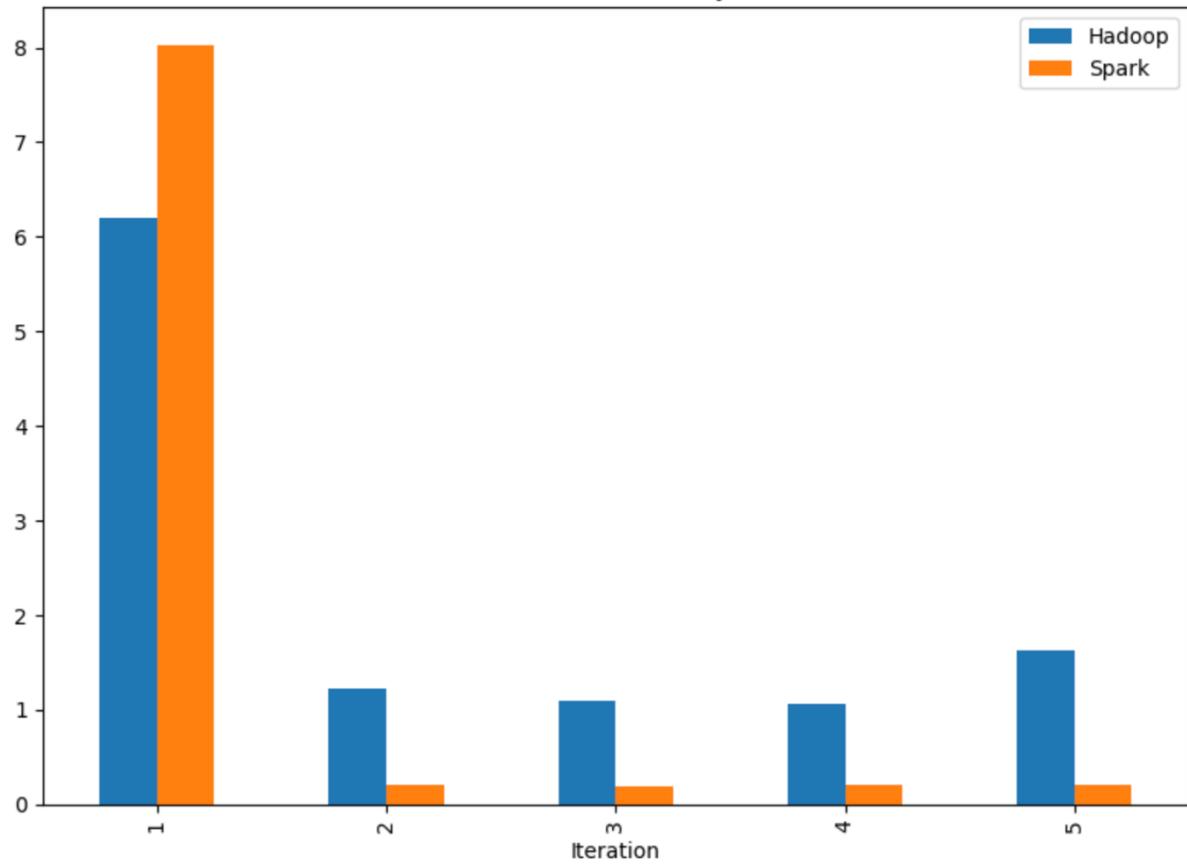
Time Taken By Query (in Sec) In One Iteration	HIVE QL	SPARK SQL
Carrier Delay Query	6.36	0.269
NAS Delay Query	1.16	1.16
Weather Delay Query	1.22	1.22
Late Aircraft Delay Query	1.63	0.213
Security Delay Query	1.09	0.229



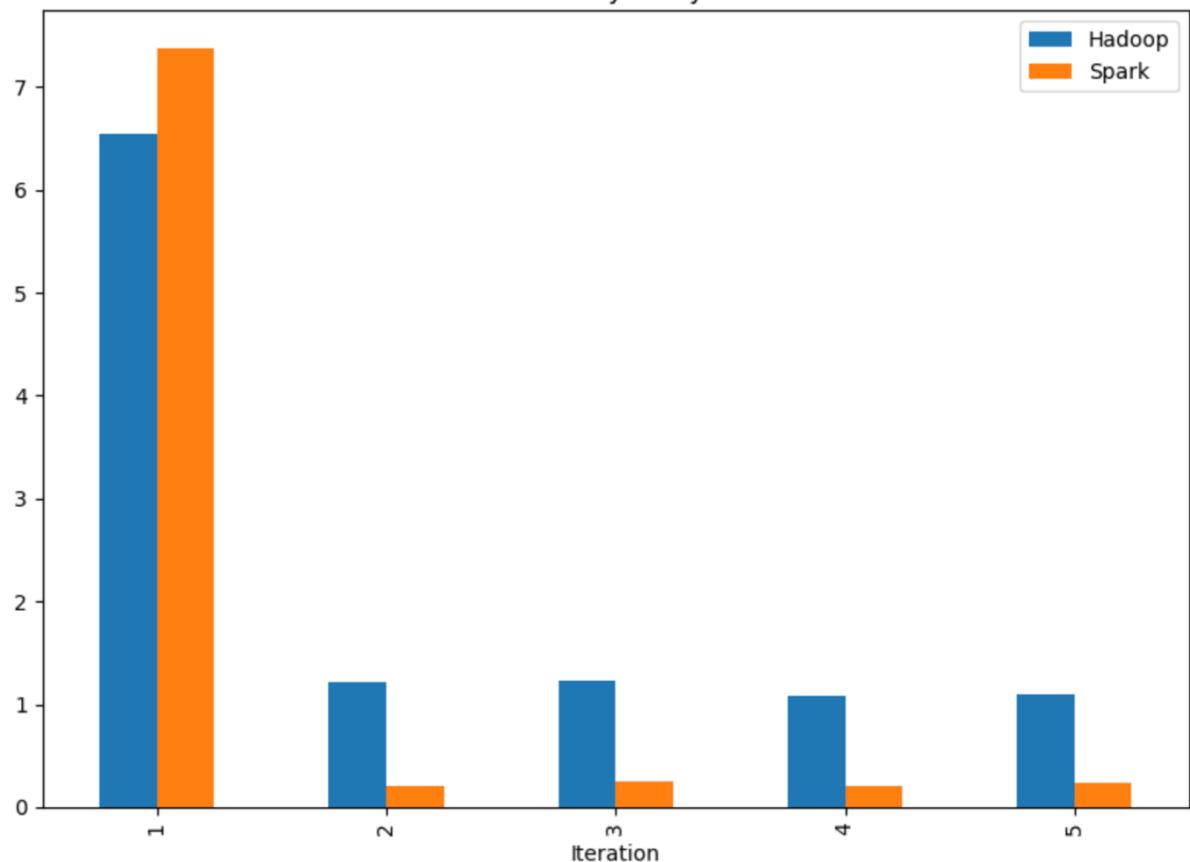
Weather Delay



Late Aircraft Delay



Security Delay



Carrier Delay

