# Data Extraction, Batch Processing with Spark Architecture for HW1

Wyatt Melin: wmelin2@illinois.ede

#### I. ABSTRACT

Big Data is rapidly growing and cloud computing model is the only go to fit to process all of it in a timely mannder. Cloud computing service provides the perfect framework for traditional for main stream open source Big Data software like Spark, Hadoop and HDFS to operate. Since the EBS transportation snapshot is restricted to the us-east-1 (N. Virgina) region, cleaning the data and running spark streaming jobs were developed accordingly in the same region for performance and cost. Cleaning the data with custom python software pandas ahead of time in an EC2 instance will reduce the amount of steps in N time spark will take for analysis.

#### II. CLEANING THE DATA

To get started with analyzing the data, first step is to attach the EBS volume to an EC2 instance. In the best interest of performance, the EC2 instance should be launched in the same region as the volume.

#### III. EXPLORE THE DATA

Once the EC2 is running and attached, a second writable storage media is required to explore the aviation transport data. In my case, I decided to create a python to loop through the aviation directory and upload only csv files (excluding the noise from other file formats) to HDFS. Once in HDFS, Spark would take over and pull the files to the EMR cluster. It was to my surprise that unzipping the data was about 50GB! Because of this, I made sure I added 250 GB writeable storage to my EMR cluster.

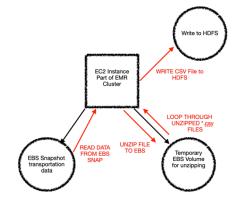


Figure 1 Demonstrates how files were both read and cleaned and finally forwarded onto HDFS to setup download for further processing by Spark and HDFS.

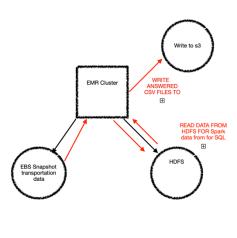
#### IV. HDFS+Spark

Using Spark is known for its in memory performance. Because of this software implementation, it offers lower application performances. However, not everything can fit into one machine. Especially when it comes to loading, storing and processing this large amount of data. There's a very limited amount of space in the L1, L2 cache of processors, not all data can fit inside this cache. Most instructions will be fed out to RAM once read from disk. At worst case scenario, the RAM will run out of space and revert to page memory/swap space. This can significant implications for overall read/write performance.

### V. AWS ARCHITECTURE AND DESGIN

To avoid the single machine solution, EMR is one solution AWS offers that is an easy end to end solution for

Big data workloads. It's very easy to setup a Spark cluster, configure master nodes and worker nodes for scaling. In the advance configuration, AWS gives the ability to configure the worker and master nodes on spot instances. In order to harness multiple worker nodes using the spot instances at once, I had to file a support ticket with AWS customer support to increase my concurrent spot instance machines quota. They increased my spot instance quota to 250 machines. [1]



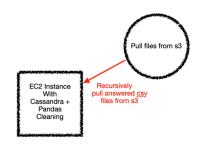


Figure 2 Demonstrates the bulk query analysis in AWS EMR and Casandra data ingestion from results posted from s3

Once processing is done on EMR cluster, I then normalized the results into one CSV per each question answered. In other words, once I have answered all the questions I then upload the results to s3. Once in s3, I then spined up an EC2 instance with Cassandra installed. Cassandra's csql has a feature that can import CSV files into the db. However, there's one caveat to this approach.

Importing via csql infers that the data is already formatted with a primary key. Therefore, there's some preprocessing required in pandas to insert a unique identifier in each record before it can be handed over to csql for file import.

# VI. HOME WORK QUESTIONS – DATA RETRVAL SPARK SQL TASK 1

- 1.2 Rank the top 10 airline by on-time arrival performance
  - Query: sqlContext.sql("SELECT COUNT(Carrier) as count,Carrier,ArrDelay FROM airports WHERE Cancelled != 1.00 AND ArrDelay IS NOT NULL GROUP BY Carrier,ArrDelay ORDER BY ArrDelay,COUNT(Carrier) DESC LIMIT 10")
- 1.3 Rank the days of the week by on-time arrival performance.
  - Query:SqlContext.sql("SELECT COUNT(DayOfWeek) as count,DayOfWeek,ArrDelay FROM airports WHERE Cancelled != 1.00 AND ArrDelay IS NOT NULL GROUP BY DayOfWeek,ArrDelay ORDER BY ArrDelay,COUNT(DayOfWeek) DESC")
- 2.1 For each airport X, rank the top-10 carriers in decreasing order of on-time departure performance from X.
  - sqlContext.sql("SELECT COUNT(Carrier) as count,DepDelay,ORIGIN FROM airports WHERE Cancelled!= 1.00 AND DepDelay IS NOT NULL AND ORIGIN IN ('CMI','BWI','MIA','LAX','IAH','SFO') GROUP BY Carrier,DepDelay,ORIGIN ORDER BY DepDelay ASC LIMIT 10")
- 2.2 For each source airport X, rank the top-10 destination airports in decreasing order of on-time departure performance from X.
  - sqlContext.sql("SELECT COUNT(Dest) as count,DepDelay,ORIGIN FROM airports WHERE Cancelled != 1.00 AND DepDelay IS NOT NULL AND ORIGIN IN ('CMI','BWI','MIA','LAX','IAH','SFO') GROUP BY Dest,DepDelay,ORIGIN ORDER BY DepDelay ASC LIMIT 10")
- 2.3 For each source-destination pair X-Y, rank the top-10 carries in decreasing order of on-time arrival performance at Y from X
  - teenagers = sqlContext.sql("WITH CTE\_L AS (SELECT Carrier, DepDelay FROM airports WHERE Origin='CMI' AND Dest='ORD' AND Cancelled != 1.00 AND DepDelay IS NOT NULL GROUP BY Carrier, Dep Delay UNION ALL SELECT Carrier, Dep Delay **FROM** airports Origin='IND' AND Dest='CMH' AND Cancelled != 1.00 AND DepDelay IS NOT NULL GROUP BY Carrier, Dep Delay **UNION** ALL SELECT Carrier, Dep Delay **FROM** airports WHERE Origin='DFW' AND Dest='IAH' AND Cancelled != 1.00 AND DepDelay IS NOT NULL GROUP BY Carrier, Dep Delay **UNION** ALL SELECT Carrier, Dep Delay **FROM** airports WHERE Origin='LAX' AND Dest='SFO' AND Cancelled != 1.00 AND DepDelay IS NOT NULL GROUP BY Carrier, Dep Delay **UNION** ALL SELECT

Carrier, Dep Delay **FROM** airports WHERE Origin='JFK' AND Dest='LAX' AND Cancelled != 1.00 AND DepDelay IS NOT NULL GROUP BY **UNION** Carrier, Dep Delay ALLSELECT Carrier, Dep Delay FROM WHERE airports Origin='ATL' AND Dest='PHX' AND Cancelled != 1.00 AND DepDelay IS NOT NULL GROUP BY Carrier, Dep Delay) SELECT \* FROM CTE\_L ORDER BY CTE\_L.DepDelay LIMIT 10")

#### Question 3.2

#### $CMI \rightarrow ORD \rightarrow LAX, 04/03/2008$

sqlContext.sql("WITH CTELAS (SELECT CMI.Carrier AS Carrier.CMI.DepDelay AS DepDelay,CMI.FlightDate AS FlightDate,CMI.DayOfWeek AS DayOfWeek,CMI.DayofMonth AS DayofMonth,CMI.Month AS Month, CMI. Year AS Year, CMI. Dest as STOP OVER, CMI. Origin AS Origin, ORD. Dest AS ORIGIN FROM airports AS ORD JOIN (SELECT Carrier, Dep Delay, Flight Date, Day Of Week, DayofMonth, Month, Year, Dest, Origin FROM airports WHERE Origin='CMI' AND Cancelled != 1.00 AND DepDelay IS NOT NULL AND FlightDate = '2008-04-03') AS CMI ON CMI.Dest = ORD.Origin WHERE ORD.Origin='ORD' ORD.Dest='LAX' AND ORD.Cancelled != 1.00 AND ORD.DepDelay IS NOT NULL AND ORD.FlightDate = '2008-04-03')SELECT \* FROM CTE L")

#### $JAX \rightarrow DFW \rightarrow CRP$ , 09/09/2008

sqlContext.sql("WITH AS CTE L (SELECT JAX.Carrier,JAX.DepDelay,JAX.FlightDat e,JAX.DayOfWeek,JAX.DayofMonth,JAX. Month,JAX. Year,JAX. Dest as STOP\_OVER,JAX.Origin,DFW.Dest FINAL\_DEST FROM airports AS DFW (SELECT IOIN Carrier, Dep Delay, Flight Date, Day Of Week, DayofMonth, Month, Year, Dest, Origin FROM airports WHERE Origin='JAX' AND Cancelled != 1.00 AND DepDelay IS NOT NULL AND FlightDate = '2008-09-09') AS JAX ON JAX.Dest = DFW.Origin DFW.Origin='DFW' WHERE DFW.Dest='CRP' AND DFW.Cancelled != 1.00 AND DFW.DepDelay IS NOT NULL AND DFW.FlightDate = '2008-09-09')SELECT \* FROM CTE L")

#### $SLC \rightarrow BFL \rightarrow LAX, 01/04/2008$

sqlContext.sql("WITH CTELAS (SELECT SLC.Carrier,SLC.DepDelay,SLC.FlightDat e,SLC.DayOfWeek,SLC.DayofMonth,SLC. Month,SLC.Year,SLC.Dest as STOP OVER, SLC. Origin, BFL. Dest FINAL DEST FROM airports AS BFL Carrier, Dep Delay, Flight Date, Day Of Week, DayofMonth, Month, Year, Dest, Origin FROM airports WHERE Origin='SLC' AND Cancelled != 1.00 AND DepDelay IS NOT NULL AND FlightDate = '2008-01-04') AS SLC ON SLC.Dest = BFL.Origin WHERE BFL.Origin='BFL' AND BFL.Dest='LAX' AND BFL.Cancelled != 1.00 AND BFL.DepDelay IS NOT NULL AND BFL.FlightDate = '2008-01-04') SELECT \* FROM CTE L")

#### $LAX \rightarrow SFO \rightarrow PHX$ , 12/07/2008

SqlContext.sql("WITH CTELAS (SELECT LAX.Carrier,LAX.DepDelay,LAX.FlightDa te,LAX.DayOfWeek,LAX.DayofMonth,LAX .Month,LAX.Year,LAX.Dest STOP\_OVER,LAX.Origin,SFO.Dest FINAL DEST FROM airports AS SFO (SELECT Carrier, DepDelay, Flight Date, Day Of Week, DayofMonth, Month, Year, Dest, Origin FROM airports WHERE Origin='LAX' AND Cancelled != 1.00 AND DepDelay IS NOT NULL AND FlightDate = '2008-12-08') AS LAX ON LAX.Dest = SFO.Origin SFO.Origin='SFO' WHERE SFO.Dest='PHX' AND SFO.Cancelled != 1.00 AND SFO.DepDelay IS NOT NULL AND SFO.FlightDate = '2008-12-08') SELECT \* FROM CTE L")

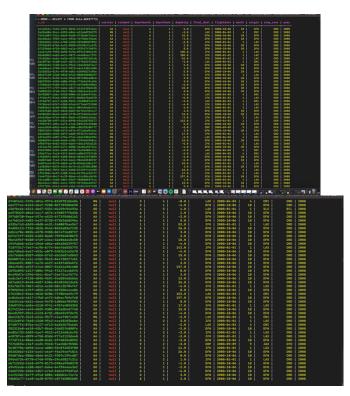
#### DFW $\rightarrow$ ORD $\rightarrow$ DFW, 10/06/2008

SqlContext.sql("WITH CTE L AS (SELECT DFW.Carrier,DFW.DepDelay,DFW.Flight Date, DFW. Day Of Week, DFW. Day of Month, DFW.Month,DFW.Year,DFW.Dest STOP OVER, DFW. Origin, ORD. Dest FINAL\_DEST FROM airports AS ORD (SELECT Carrier, Dep Delay, Flight Date, Day Of Week, DayofMonth, Month, Year, Dest, Origin FROM airports WHERE Origin='DFW' AND Cancelled != 1.00 AND DepDelay IS NOT NULL AND FlightDate = '2008-10-06') AS DFW ON DFW.Dest = ORD.Origin WHERE ORD.Origin='ORD' ORD.Dest='DFW' AND ORD.Cancelled != 1.00 AND ORD.DepDelay IS NOT NULL AND ORD.FlightDate = '2008-10-06') SELECT \* FROM CTE L")

#### LAX $\rightarrow$ ORD $\rightarrow$ JFK, 01/01/2008

sqlContext.sql("WITH CTE L AS (SELECT LAX.Carrier,LAX.DepDelay,LAX.FlightDa te,LAX.DayOfWeek,LAX.DayofMonth,LAX .Month,LAX,Year,LAX,Dest STOP\_OVER,LAX.Origin,ORD.Dest FINAL DEST FROM airports AS ORD (SELECT Carrier, DepDelay, Flight Date, Day Of Week, DayofMonth, Month, Year, Dest, Origin FROM airports WHERE Origin='LAX' AND Cancelled != 1.00 AND DepDelay IS NOT NULL AND FlightDate = '2008-01-01') AS LAX ON LAX.Dest = ORD.Origin ORD.Origin='ORD' WHERE ORD.Dest='JFK' AND ORD.Cancelled != 1.00 AND ORD.DepDelay IS NOT NULL AND ORD.FlightDate = '2008-01-01')SELECT \* FROM CTE\_L")

## VII. HOME WORK QUESTIONS – CASSANDRA, TASK 1 QUESTIONS 3.2,2.1,2.2,2.3



f40fe41-fdfb-401a-9976-01d9f81bb484	MQ	null	3	4.1	-8.0	LAX	2008-04-03	4	CHI	ORD	2868
Arfffac-bcb3-4baf-9288-86f789d86b98	ÃÃ		š i		-4.0	DEW	2888-18-86	10	DFW	080	2888
26fbc6e-6241-4bd7-9321-d6459c96693a	00	null	¥ 1	- 61	1.0	LAX	2888-91-84	~~~	SLC	BFL	2888
e978329-d84d-4cc7-a57e-af685fffbd3b	00	null	31	- 11	1.0	DEW	2008-10-06	10	DFW	ORD	2868
9768f28-9eae-497e-a533-6f73f568d1b2	AA I	null	š i	- 11	-2.0	DEW	2008-10-06	10	DEW	080	2998
e76b2e3-4d82-4s2f-8728-573b2d6829be	AA I	null	: 1		5.0	DEN	2008-10-00	10	DFW	ORD	2998
335e7a3-65d9-48d8-a431-24000fbae567	ÃÃ.	null	- 1	2	52.0	2FK	2008-01-01	1	LAX	ORD	2998
4d55c13-ffd2-482b-9442-862da83ef135	20	null	: 1	- 11	26.0	DEW		10	DFW	080	2008
AScaf0e-883b-45f8-9358-867af1Ad8f2f	AA .	null	: :	- 11	3.0	DEN	2008-10-05	10	DFW	ORD	2000
2fa71ff-b325-4183-8ab5-7599d4828bf5	20	null	8	- 11	2.0	DEW	2008-10-00	10	DFW	ORD	2000
278/177-0320-4183-8800-/09904828070 be4696f-8488-4fa9-a3ca-1be86da48c58	AA I	null	: 1	- 11	15.0	DFW	2005-10-05	10	DFW	ORD	2008
919a8eh-zA3a-45Ad-a8Aa-a81a5h529f57	**	null		- 11		DEW	2868-18-86	18	DFW	ORD	2888
949a8eb-a63a-456d-a86a-a64a5b529f57 1dff3c7-7647-4c9b-b774-564f665557f5	**		: 1	1 2	142.0	JFK	2005-10-00		LAX	ORD	2868
de7d5ed-ba67-453c-a9f9-3481b2c62679						DEN	2008-10-06		DEW		2998
5cf6d66-828f-45b5-bfe2-abcbdd7a5b63	AA I				15.0	DEM	2008-10-06	10	DFW		2008
6b8871d-ca1c-418c-9bc5-84a738277a51						DFW	2008-10-06		DFW		2008
0saa691-0887-4s76-a42f-a433fs03sbfe					3.0	DFW	2008-10-06		DFW		2998
84c9484-a53f-47ce-8857-5e344a9dce88					197.0	DFW	2008-10-06		DFW		2008
b93e893-1d17-488c-99a1-f3127aceb5fb	AA I				0.0	DFW	2008-10-06		DFW		2008
						DFW	2008-10-06		DFW		2008
edc3efc-9130-489b-9691-9c65846d6b66					13.0	DFW	2008-10-06		DFW		2008
87ad619-8448-4b87-b38e-81681561de6b					26.0	DFW	2008-10-06		DFW		2008
	UA				-4.0	JFK	2008-01-01		LAX		2868
							2008-10-06		DFW		2998
76238c2-5dse-482d-828b-73f8b4669c79	MQ				153.0	LAX	2008-04-03		CHI		2998
					197.0		2008-10-06		DFW		2008
eb341ee-b621-4aea-9af8-1d06daf03983						DFW	2008-10-06		DFW		2008
3b0f3fe-ab5e-4218-9f41-4f07a489d355						3FK			LAX		2008
1f42164-ce49-48d9-938b-853adb4d7748	AA				15.0	DFW	2008-10-06	10	DFW		2008
e4d199f-d9e1-4123-b7d1-d8e6949fdefb					-4.8	DFW	2888-18-86	10	DFW		2008
be11b75-32e3-42a4-957f-c1eaf387ca35	MQ				-2.0	LAX	2008-04-03		CHI	ORD	2008
212e8fd-76e8-4260-95c2-ccs48103bebe	AA				3.0	JFK	2008-01-01		LAX		2998
f48f7fe-87ba-4c27-af13-6a2b2b75e6d4	MQ				-2.0	LAX	2008-04-03		CHI		2008
82315e0-a610-45b7-86ab-24b82f6d00fe	AA				-2.0	DFW	2008-10-06	10	DFW		2998
e85a733-3689-4aef-9515-a912468cbc95	AA I				-2.0	JFK	2008-01-01		LAX		2998
c8f8fc4-d899-4750-83dd-977e2123dae0	AA	null			-2.0	3FK	2008-01-01		LAX	ORD	2008
77df116-88±6-44d0-b4b1-2976025e8592	AA İ				-2.0 j	DEM	2008-10-06	10	DFW	ORD	2008
2763b9a-c34f-41d1-9318-faacb8c95586	AA I	null	9 i		-3.0	CRP	2008-09-09		JAX	DFW	2008
e387f8a-dd9d-44ce-a88d-55c8f2d53f88	AA I				41.0	3FK	2008-01-01		LAX	ORD	2008
53d2bb2-6226-4c61-a6af-fda29cef4814	AA				26.0	DFW	2888-18-86	10	DFW	ORD	2008
9687d6a-58bb-48de-b421-f58fc199cd87	AA				0.0	DFW	2888-18-86	10	DFW	ORD	2008
964d734-9778-4748-9f8e-29c45827c31c	AA I	null i			1.0	3FK	2898-91-91		LAY	ORD	2998
b712d2d-1dc8-49f9-8173-b98sa99b8378	AA		ē i		-1.0	DFW	2008-10-06	10	DEW	080	2998
9492eab-e189-4b6f-bd4e-64f59e4ee362	00		- i		-2.0	DEW	2008-10-06	10	DFW	ORD	2008
3ebf22b-3d2d-43b7-a7ed-63dc2f93dfad	ÄÄ	null	- i		-4.0	DEW	2008-10-06	10	DEW	080	2008
c913dc-5e7f-491d-b994-e94bb1c74c7a	ÃÃ.	null	* 1		3.0	DEW	2008-10-06	10	DFW	ORD	2998
db83a77-1es0-4a20-8793-c07765002b85	ÄÄ	null			2.0		2008-10-06	10	DFW	ORD	2000
88 rows)											
1sh>											
1sh>											
1sh>											

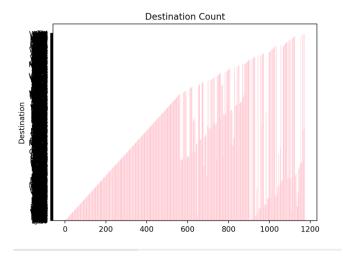
cqlsh> select * from data.quest2;			
		depdelay	origin
91be768c-8e88-4455-ae6f-0235096081f4	+ I 5	-1.0	CMI
60199c79-4081-43da-8cd5-46e25fa2912f	5091	-1.0	
41fdaa92-0561-47d3-8922-da17d527937f		-1.0	
c76d0703-56c4-4cb1-963f-e5b02008f6db		-1.0	
595e5244-6f49-4b6d-a661-eec057c5a3e8		-1.0	
e4a1ef8f-2ffe-4548-a56c-c00b4d91cbe3		-1.0	
70989d27-2979-49d9-af60-209b08c324cb	61048	-1.0	
68cf36d2-704a-44d1-a67a-32a57e241c04	86	-1.0	
a1a201cb-21f8-4f08-ab74-8e4c99127036		-1.0	
49bcc02d-8961-4a79-b975-e3d5c93b24f2		-1.0	
f2b2af11-f338-45b5-ad52-b6d059016e03		-1.0	
5c433d1f-3703-4cb3-9042-ed4e68b8d280		-1.0	
5d10cc90-c700-4a8d-9489-85acdbd207dd	3469	-1.0	
a0436fc3-aef0-4b8a-9720-1780fc4a0fde	1696	-1.0	
487a6337-7603-48b1-a64c-3430c71123a9		-1.0	
151a2243-92ea-4a44-8ef9-f63c6999b142	2782	-1.0	
fd5d6764-c466-4c5e-b3dd-2e435b75b28f	2288	-1.0	
6f8a6e0b-016e-410a-be81-b8c530db7361	8	-1.0	BWI
08564134-aa8d-4073-bded-dc9ba7b2f81e	344	-1.0	
3ddcc478-4b47-4262-b67c-5e5b35ca539f	20638	-1.0	IAH

# 

#### VIII. HOME WORK QUESTIONS – TASK 1 QUESTIONS 1.2, 1.3

1.2 Re	esults – Top 10							
Count	Carrier	ArrDelay						
536220	DL	-1						
464211	WN	-1						
445166	US	-1						
420117	AA	-1						
396823	UA	-1						
282005	NW	-1						
221618	CO	-1						
123695	HP	-1						
110624	TW	-1						
91606	00	-1						
1.3 Results – Top 14								
Count	DayOfWeek	ArrDelay						
538950	1	-1						
537849	2	-1						
529760	3	-1						
515826	4	-1						
509155	5	-1						
493765	7	-1						
461557	6	-1						
471416	2	-10						
464469	1	-10						
449941	6	-10						
449026	3	-10						
444732	7	-10						
409823	4	-10						
399515	5	-10						

As per my analysis of distribution, I ran a spark sql query to group by destination and count the number of times a particular destination was selected across the whole data set. Since there were so many different destinations, I had a tough time labeling in matplotlib on the y axis. Please find below results of the distribution of popular destinations:



## REFERENCES

 $[1]\ https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-spot-limits.html$