HYBRID CLOUD SERVICES

Mentors:

-Xu (Simon) Chen

Team Members:

- Akash Singh
- Jaison Babu
- Surekha Jadhwani
- Vignesh Shanmuganathan

-

1. Introduction:

1.1 Problem Statement:

Imagine a case where data is regarding flight journeys, flight launch etc. Some of the data also belong to military airbases. The data of military airbases is something that must not be compromised, however, other data which could be treated public can be worked upon on a public cloud. One can even take the case of credit card no. of credit cards to not be compromised or there are so many cases where some portion of data based on any text like company name etc. or some regex is sensitive.

Our idea is to come up with a text based mechanism where once user specifies different text string to be treated as sensitive, the data containing those text tags won't be compromised. The sensitive information will remain on private cloud whereas all public data can be sent to the public cloud, thus making it possible for private firms to be able to scale to public cloud without compromising any of their sensitive information.

For our case, we are running the text based segregation on live twitter feed. Our reasons for doing this is to demonstrate that the idea can work on a very large scale (we worked with about 20 GB of data) and something which is applicable to live streaming given that there are several applications which have live streaming of data as their backbone. Given these two aspects of twitter feed, we built our framework around it and our framework could easily be adopted to any data which could be either from a warehouse or from some live feed applicable to enumerable scenarios enabling private firms and users to be able to exploit the public cloud. The overall goal is to be able to attract more users to the public cloud along with giving them an assurance that none of their sensitive information will be compromised.

1.1.1 Post Segregation:

Data segregation is not enough, if analytics job cannot be run on this data. Our scope is to even run different analytics jobs like Geo-Location wise data distribution using time-zone tag, distribution of tweets based upon language, total counts of tweet in public and private cloud etc on this hybrid setup.

We even added an auto-scaling mechanism which makes assumption that public cloud capacity can be increased as and when needed easily and that by setting space limitations on the usage of private cloud, data can be shifted to public cloud automatically and securely, once space usage is exceeded. This should ideally happen in a VPN and our project makes an assumption that VPN could be setup between the private and public cloud.

2. Architecture of our implementation:

Note: We have considered MOC as public cloud and AWS as private cloud given the cost constraints.

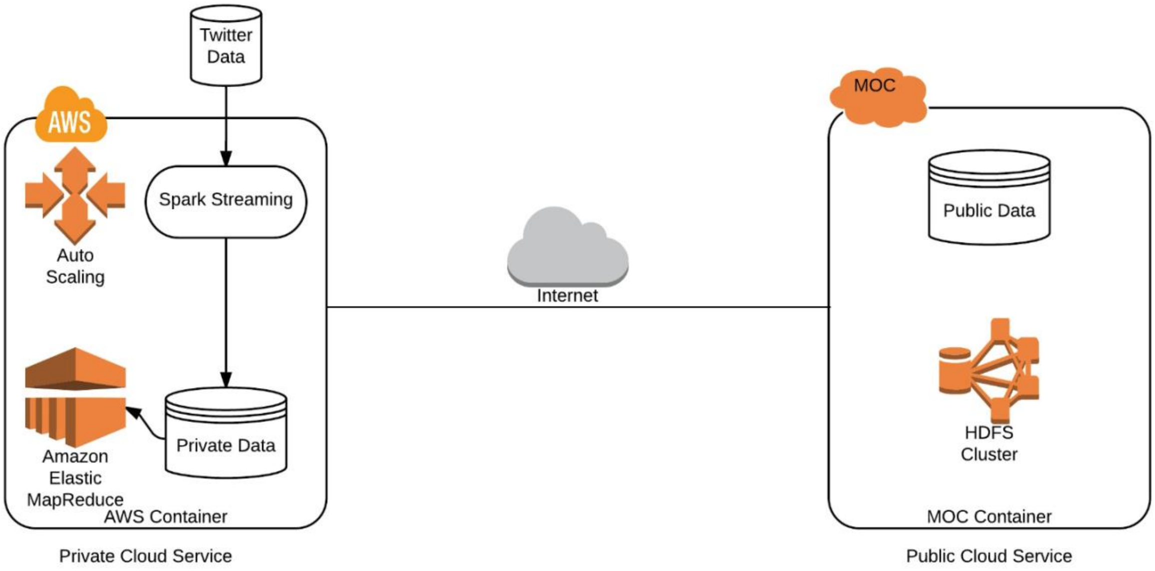


Figure 1: Hybrid cloud architecture

2.1 Scope:

a. Creating a public cloud in MOC that contains non-sensitive data.
b. Creating a private cloud in AWS that contains sensitive data.
c. Set-up of Hadoop clusters in AWS and MOC.
d. Establishing communication between AWS and MOC for data transfers.
e. A spark streaming service running on AWS will gather real time data from twitter and partition data according to user configuration for classifying data as sensitive/non-sensitive. The proposed data classifiers include hash tag mechanism and linguistics.
f. Based on the hash tags, the created classifier will distribute data across AWS and MOC.
g. Running jobs in the following setups:
    i. Running data analysis job on public cloud spanning the complete dataset (sensitive and non-sensitive data).
    ii. Running data analysis job on private cloud spanning the complete dataset.
    iii. Running data analysis job on non-sensitive data on public cloud and on sensitive data on private cloud, in parallel.
h. Performing benchmark analysis on the above proposed configurations.

2.2 Features:

a. Easy user configuration by using configuration file.
b. Establishing communication between the cloud clusters.
c. Ability to run data analysis jobs on either/both clouds.
d. Analyzing real world data (twitter feed) on real world load on the hybrid setup.
e. Configuring the setup to sustain load of such load without major failures spanning both clouds.

3. Design & Implementation:

The project fundamentally compromises of 3 parts that involves:

a) Private cloud - Amazon Web Services (AWS)
b) Private cloud - Massachusetts Open Cloud (MOC)
c) Data - Live twitter feed streaming using Twitter4j API

1.1 Cluster Configurations:

AWS Configuration		MC Configurations	
No of Machine:	2	No of Machine:	5
Ram:	1GB/machine	Ram:	8GB/machine
Storage:	15GB/machine	Storage:	80GB/machine
Sensitive Data:	4GB	Non-sensitive data:	15gb

1.2 Challenges faced:

a) Setting up spark streaming environment in AWS and Hadoop cluster in MOC.
b) Establishing secure communication between clouds.
c) Data segregation using user configuration for twitter data and auto scaling.

1.3 New Technology Learnt:

a) Private cloud and public cloud set-up in AWS and MOC.
b) Configuring HDFS in cluster mode.
c) Spark streaming job.
d) Hadoop analytical job.
e) Task tracking tool such as Trello.
f) Twitter 4j API in Java.

1.4 Implementation:

1.4.1 Data tagging mechanism:

Based on user twitter configuration file, the spark streaming job classifies the inflow of the twitter data into sensitive and non-sensitive data. A tweet that contains a hash tag mentioned in twitter configuration file is considered as sensitive data and is stored in HDFS on AWS (private cloud) and any tweet that does not contain the tag mentioned in twitter configuration file is considered as non-sensitive data and is stored in MOC (public cloud) HDFS.

1.4.2 Streaming job:

A spark streaming job be an in memory continuous operation is opted for running a data gathering task that continuously retrieves twitter data from twitter server and based on data tagging mechanism the data is segregated into private and public data to gather enough sample data set that can be used to run analytical job to perform performance evaluation of different cloud processing jobs.

1.4.3 Analytical job:

Different analytical jobs such as classifying tweets based on location, language and time zone is run that combines execution of the job either in standalone mode or clubbed mode to evaluate performance of these jobs to showcase the benefits of the hybrid cloud service over standalone cloud models.

1.4.4 Performance benchmarking:

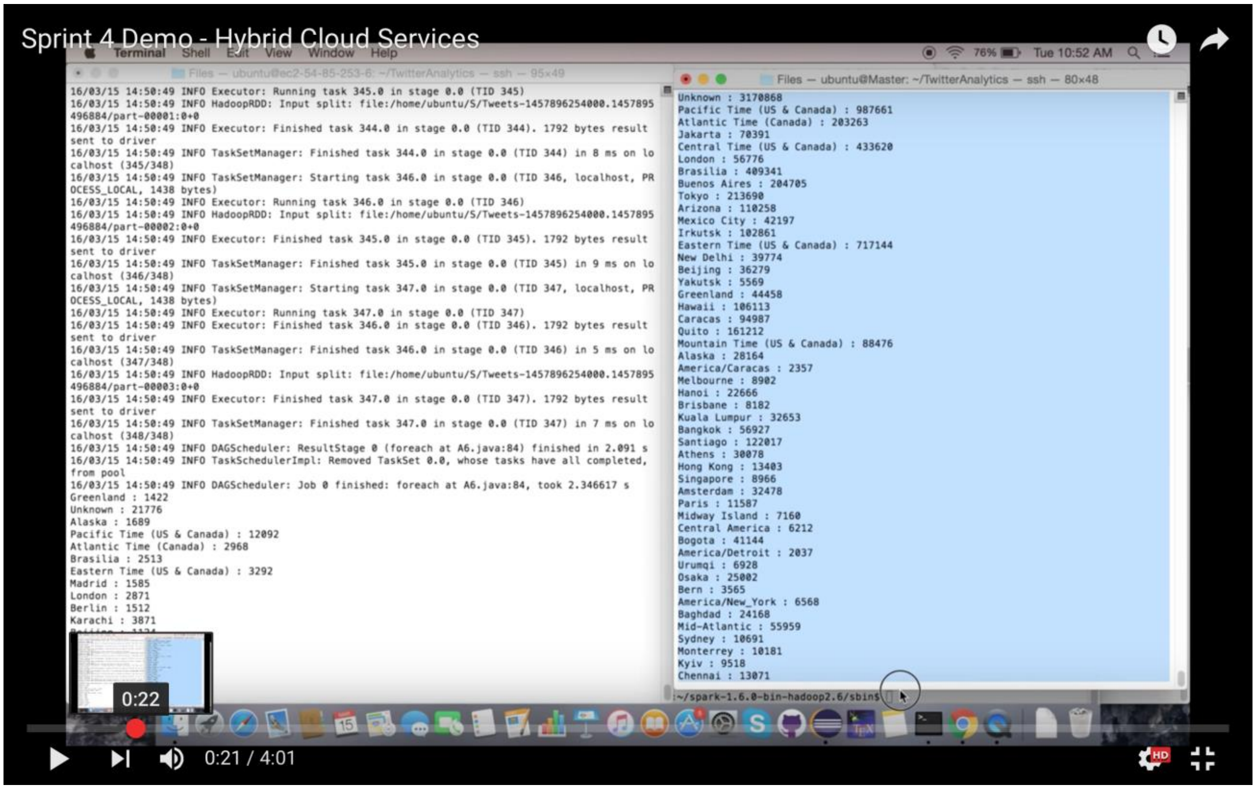
Based on different jobs run in public, private and hybrid model, we evaluate the options of running jobs in the above different models.

1.4.5 Auto-scaling:

Based on the user scaling configuration file, that contains 2 parameters (threshold of data in bytes, time in 10 sec intervals) [refer readme for more information properties], the job continuously monitors the private cloud in the mentioned time span if the current sensitive data size has exceeded the threshold mentioned and if true, the entire sensitive data is moved to public cloud to accommodate more data space in private cloud.

4. Results:

4.1 Segments of result of location wise tweet count : Left portion is result on private cloud and Right portion is result on public cloud.



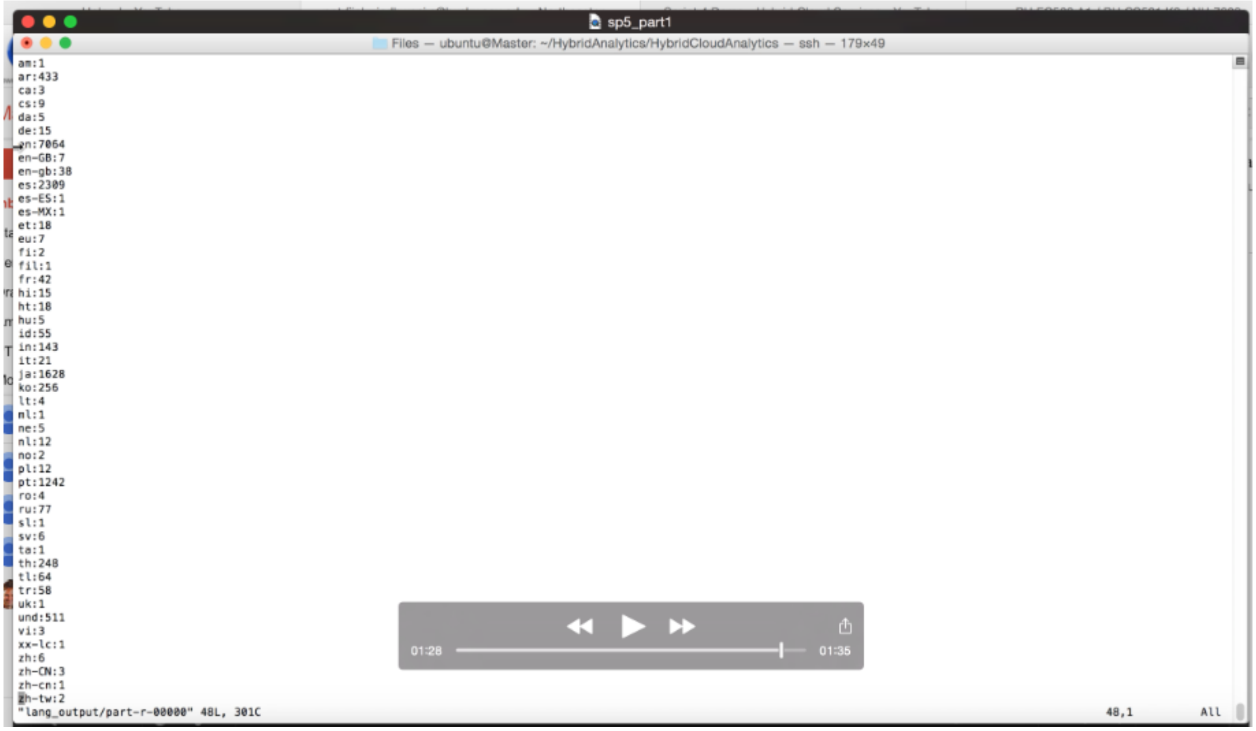
Files — ubuntu@ec2-54-85-253-6: ~/TwitterAnalytics — ssh — 95×49

```
16/03/15 14:50:49 INFO Executor: Running task 345.0 in stage 0.0 (TID 345)
16/03/15 14:50:49 INFO HadoopRDD: Input split: file:/home/ubuntu/S/Tweets-1457896254000.1457895
496884/part-00001:0+0
16/03/15 14:50:49 INFO Executor: Finished task 344.0 in stage 0.0 (TID 344). 1792 bytes result
sent to driver
16/03/15 14:50:49 INFO TaskSetManager: Finished task 344.0 in stage 0.0 (TID 344) in 8 ms on lo
calhost (345/348)
16/03/15 14:50:49 INFO TaskSetManager: Starting task 346.0 in stage 0.0 (TID 346, localhost, PR
OCESS_LOCAL, 1438 bytes)
16/03/15 14:50:49 INFO Executor: Running task 346.0 in stage 0.0 (TID 346)
16/03/15 14:50:49 INFO HadoopRDD: Input split: file:/home/ubuntu/S/Tweets-1457896254000.1457895
496884/part-00002:0+0
16/03/15 14:50:49 INFO Executor: Finished task 345.0 in stage 0.0 (TID 345). 1792 bytes result
sent to driver
16/03/15 14:50:49 INFO TaskSetManager: Finished task 345.0 in stage 0.0 (TID 345) in 9 ms on lo
calhost (346/348)
16/03/15 14:50:49 INFO TaskSetManager: Starting task 347.0 in stage 0.0 (TID 347, localhost, PR
OCESS_LOCAL, 1438 bytes)
16/03/15 14:50:49 INFO Executor: Running task 347.0 in stage 0.0 (TID 347)
16/03/15 14:50:49 INFO Executor: Finished task 346.0 in stage 0.0 (TID 346). 1792 bytes result
sent to driver
16/03/15 14:50:49 INFO TaskSetManager: Finished task 346.0 in stage 0.0 (TID 346) in 5 ms on lo
calhost (347/348)
16/03/15 14:50:49 INFO HadoopRDD: Input split: file:/home/ubuntu/S/Tweets-1457896254000.1457895
496884/part-00003:0+0
16/03/15 14:50:49 INFO Executor: Finished task 347.0 in stage 0.0 (TID 347). 1792 bytes result
sent to driver
16/03/15 14:50:49 INFO TaskSetManager: Finished task 347.0 in stage 0.0 (TID 347) in 7 ms on lo
calhost (348/348)
16/03/15 14:50:49 INFO DAGScheduler: ResultStage 0 (foreach at A6.java:84) finished in 2.091 s
16/03/15 14:50:49 INFO TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed,
from pool
16/03/15 14:50:49 INFO DAGScheduler: Job 0 finished: foreach at A6.java:84, took 2.346617 s
Greenland : 1422
Unknown : 21776
Alaska : 1689
Pacific Time (US & Canada) : 12092
Atlantic Time (Canada) : 2968
Brasilia : 2513
Eastern Time (US & Canada) : 3292
Madrid : 1585
London : 2871
Berlin : 1512
Karachi : 3871
```

Files — ubuntu@Master: ~/TwitterAnalytics — ssh — 80×48

```
Unknown : 3170868
Pacific Time (US & Canada) : 987661
Atlantic Time (Canada) : 203263
Jakarta : 70391
Central Time (US & Canada) : 433620
London : 56776
Brasilia : 409341
Buenos Aires : 204705
Tokyo : 213690
Arizona : 110258
Mexico City : 42197
Irkutsk : 102861
Eastern Time (US & Canada) : 717144
New Delhi : 39774
Beijing : 36279
Yakutsk : 5569
Greenland : 44458
Hawaii : 106113
Caracas : 94987
Quito : 161212
Mountain Time (US & Canada) : 88476
Alaska : 28164
America/Caracas : 2357
Melbourne : 8902
Hanoi : 22666
Brisbane : 8182
Kuala Lumpur : 32653
Bangkok : 56927
Santiago : 122017
Athens : 30078
Hong Kong : 13403
Singapore : 8966
Amsterdam : 32478
Paris : 11587
Midway Island : 7160
Central America : 6212
Bogota : 41144
America/Detroit : 2037
Urumqi : 6928
Osaka : 25002
Bern : 3565
America/New_York : 6568
Baghdad : 24168
Mid-Atlantic : 55959
Sydney : 10691
Monterrey : 10181
Kyiv : 9518
Chennai : 13071
:~/spark-1.6.0-bin-hadoop2.6/sbin$
```

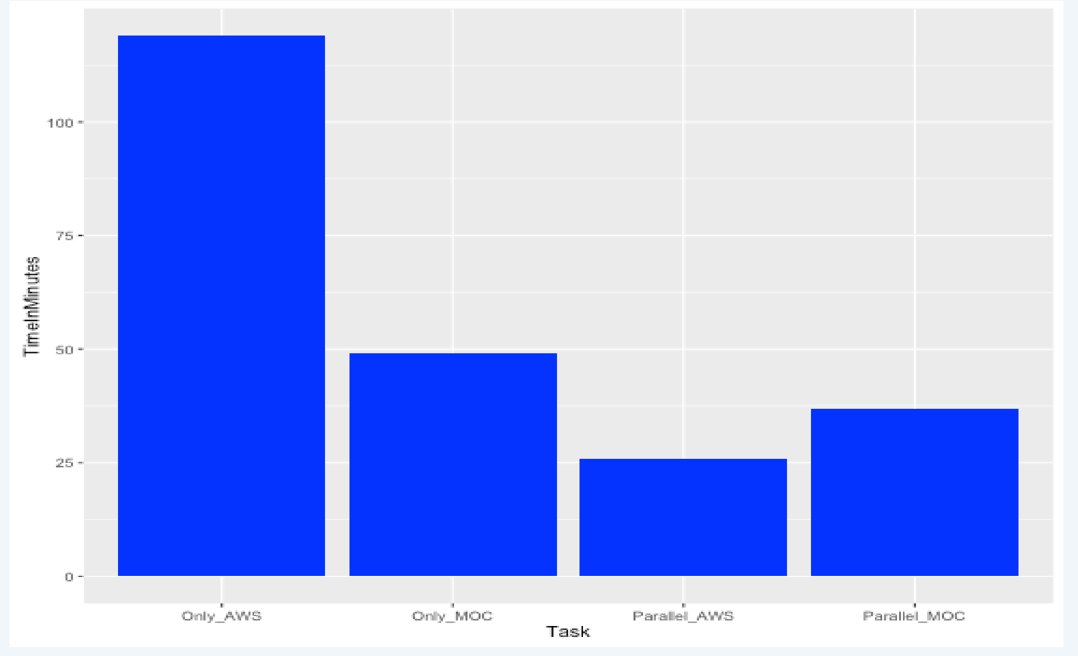
4.2 Portion of result of running language based distribution on data on small dataset.



sp5_part1
Files — ubuntu@Master: ~/HybridAnalytics/HybridCloudAnalytics — ssh — 179×49
01:28
01:35

```
am:1
ar:433
ca:3
cs:9
da:5
de:15
en:7064
en-GB:7
en-gb:38
es:2309
es-ES:1
es-MX:1
et:18
eu:7
fi:2
fil:1
fr:42
hi:15
ht:18
hu:5
id:55
in:143
it:21
ja:1628
ko:256
lt:4
ml:1
ne:5
nl:12
no:2
pl:12
pt:1242
ro:4
ru:77
sl:1
sv:6
ta:1
th:248
tl:64
tr:58
uk:1
und:511
vi:3
xx-lc:1
zh:6
zh-CN:3
zh-cn:1
zh-tw:2
"lang_output/part-r-00000" 48L, 301C                                                    48,1          All
```

4.3 Following is the result of the performance benchmarking we did on analytics job of geo-location wise tweet count.



5. Limitations:

a) Currently the hybrid service has been modelled to work for data from twitter based on provided tweet tags.
b) Type of analytical job depends on the meta-data provided by twitter.
c) Communication security depends on Linux security layer.
d) Makes an assumption that a VPN can be created between MOC (public) and AWS (private) cloud.

6. Conclusion:

We were successful in coming up with a framework which can segregate data on the presence of text configuration between private and public cloud to enable use of public cloud to private firm. It overcomes the hurdle of sensitive information compromise, which otherwise is difficult to tackle. We did this on live data feed to demonstrate its abilities to work not just on data in a warehouse, but even on live feed on which several analytics are done these days. Our model can help in giving assurance to firms that they will be able to use public cloud without compromising any sensitive data and attract more people to the public cloud.

7. Future Scope:

e) Extend the security layer between private and public cloud by implementing an VPN communication layer
f) Current data segregation method relies on preprocessed text inputs, extend the data segregation for generic data by preprocessing using a spark job.
g) Extend auto-scaling to modify number of private and public cluster nodes based on requirement of analytical job.
h) Provide cluster backup in case of any fatal issue with the cluster.