HYBRID CLOUD SERVICES

Mentors:

-Xu (Simon) Chen

Team Members:

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

-

1. Introduction:

1.1 Problem Statement

Virtualization technology has drastically re-shaped the enterprise IT landscape. Clouds of different flavors are becoming the dominant environment for most or sometimes all of the enterprise workload. Public cloud, for example Amazon AWS, Google Compute Engine, Microsoft Azure, can offer resource on-demand at very high capacity with varying quality of service guarantees; private cloud, on the other hand, offers on-premise, secure, and yet dynamic compute environment that can be easily integrated into existing in-house compute infrastructure.

Due to security concerns, not all companies are going all-in with public cloud. Certain workloads, such as credit card transactions, cannot be executed on public cloud due to regulatory concerns. That said public cloud, especially due to its elasticity, is an ideal platform for large-scale data processing. The ability to instantly spin up many instances just for one processing job and tear them down after job completion is the ideal use case. Private cloud, on the other hand, is less scalable, requires an investment up-front (although cheaper). As such, a hybrid cloud setup is gaining traction. The typical architecture is to combine private cloud and public cloud to create the abstraction of a single compute infrastructure to allow application owners to deploy and run applications on top. At the same time, a policy layer must be designed to smartly place workload onto different clouds for various concerns:

1) Data locality: processing jobs should be scheduled close to where the data is;
2) Data sensitivity: certain sensitive data must be placed in-house;
3) Cost: private cloud overall is cheaper than public cloud, although less capable of handling bursty workload.

In this project, we make a modest attempt to realize such a hybrid cloud setup, by placing a data processing pipeline spreading across both types of clouds. We designed and implemented a real-time streaming application that smartly places data based on sensitiveness onto different clouds. In the end, we run several processing workload on each cloud and cross clouds to evaluate the performance difference. Overall, we hope to provide insights into building and running such a hybrid cloud setup.

1.2 Concrete Example

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, number 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.

1.3 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 based 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 when 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

In our architecture, we have two different clouds, one public and one private. Each cloud has its control plane for dynamically add or remove compute resources. We assume that the two clouds are connected to each other via either the Internet or a secure connectivity, such as IPSEC VPN, such that network traffic can flow among VMs on the same cloud and across clouds. Additional care must be taken to make sure network configuration is done properly, such that we do not expose VMs wide-open to the general public – for example, HDFS clusters should not be exposed to the Internet at all times.

Certain clouds might have more features than others. For example, Amazon has a dynamic scaling capability, such that it will automatically spin up more VMs when the workload for existing VMs are high. We can choose to leverage this capability if needed in real world setup. In our setup, we run a data streaming application on the private cloud, simulating a traditional IT workload that processes incoming data in a real-time fashion. Both clouds have a storage layer for long-term data keeping. The difference is that our data pipeline tries to perform real-time

analysis to classify the data into sensitive and non-sensitive information: sensitive information is kept on-premise on the private cloud, while non-sensitive information is shipped via the network to the public cloud setup.

When there is a need for data processing, we can either spin up worker VMs on each cloud to process the data stored there, or run the workers to access data across clouds. The benefit work running processing jobs on different clouds is that we maximize data locality. In theory, only aggregated/processed data is traversing through the network links in between clouds. The downside is that we have to change the existing workflow, for example, a simple map-reduce job has to be split into two map-reduce jobs and another aggregation job.

The setup that we are using is as follows:

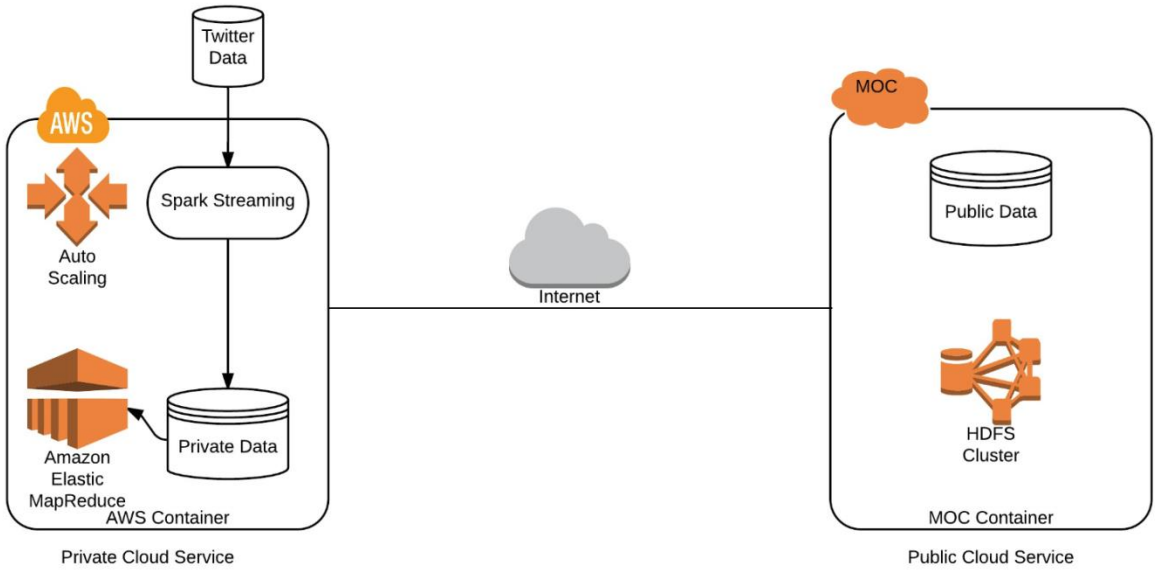


Figure 1: Hybrid cloud architecture

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. 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

3.1 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

3.2 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.

3.3 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.

3.4 Implementation:

3.4.1 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.

3.4.2 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.

3.4.3 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.

3.4.4 Performance benchmarking:

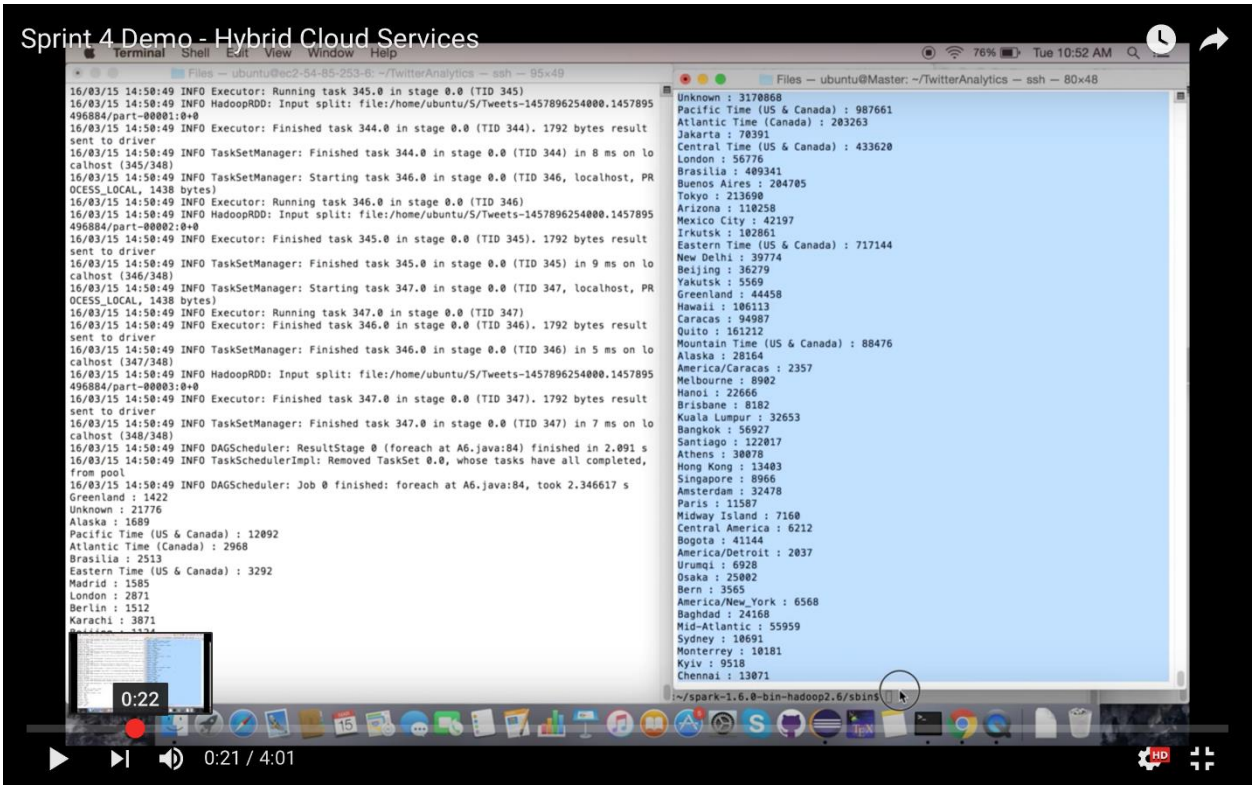
Based on different jobs run in public, private and hybrid model, we evaluate the options of running jobs in the above different models.

3.4.5 Auto-scaling:

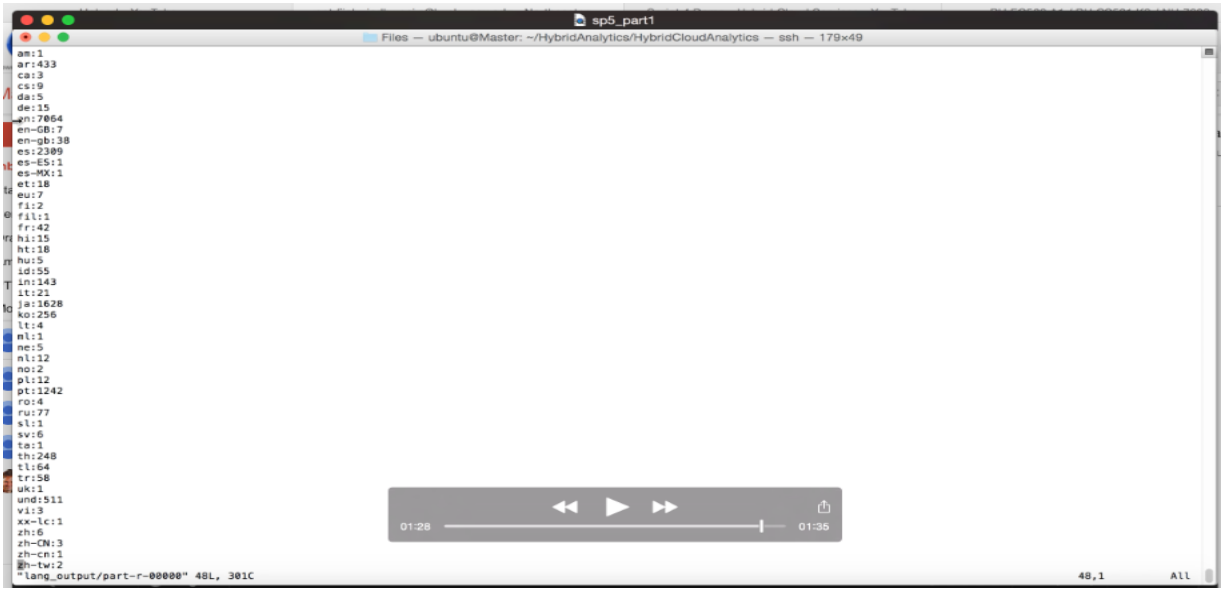
Based on the user scaling configuration file, that contains 2 parameters (threshold of data in bytes, time in 10 sec intervals) [refer readme for more information properties], the job continuously monitors the private cloud in the mentioned time span if the current sensitive data size has exceeded the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

4. Results:

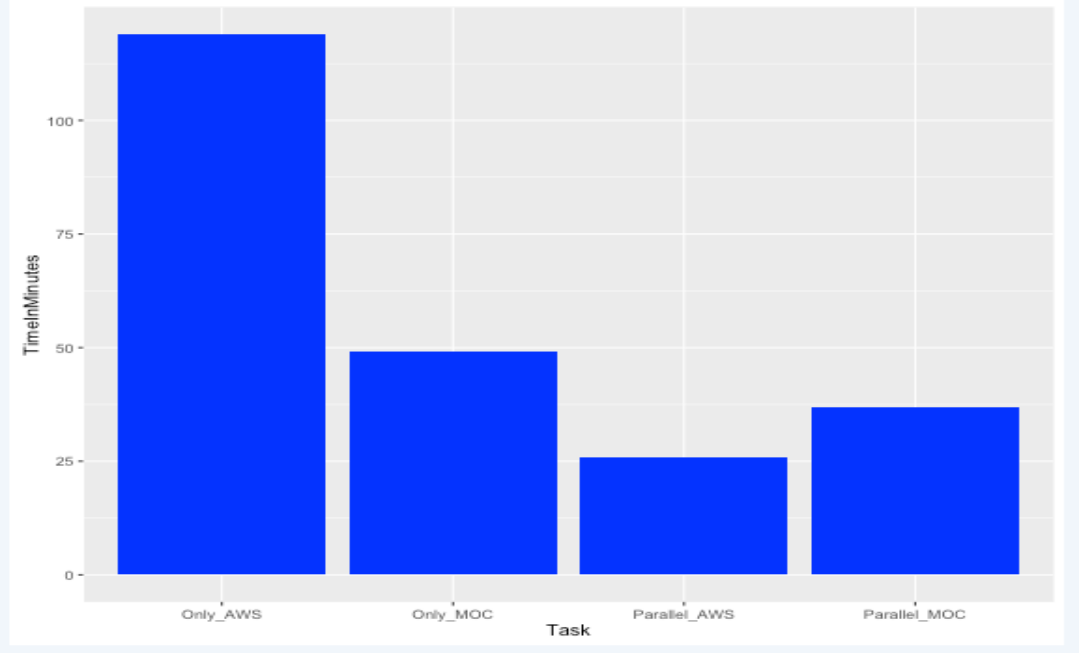
4.1 Segments of result of location wise tweet count: Left portion is result on private cloud and Right portion is result on public cloud.



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



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



Results indicating the feasibility of the above proposed solution:

For our setup, we used a private cloud with limited capacity and a public cloud with very high capacity in our configuration, which is a case true for several of small firms. The results we are comparing here our mainly the differences between running complete dataset on private cloud and running it in a hybrid way such that no sensitive information is compromised. We took a ratio of 1:10 for sensitive: non sensitive data ratio for our benchmarking. The results we obtain indicate significant performance gains by using hybrid model of execution. We even have added results of just utilizing the public cloud, which ran the fastest, but had some compromises in data security and privacy. This was done on complex time consuming jobs. Below is our result:

Public cloud used is MOC and private cloud is AWS.

Record 1:
11 mb moc -> 12.91
11 mb aws -> 2234.52 seconds
10 mb on MOC & 1 mb on AWS ->
Max (11.8, 171.82) + 4 seconds transfer time + aggregation 23 seconds = total 198.82 seconds

Record 2:
22 mb moc -> 21.32 seconds
22 mb aws -> 4752.42 seconds
20 mb on MOC & 2 mb on AWS ->
Max (18.45, 366.66) + 3 seconds transfer time + aggregation 25 seconds = total 394.66 seconds

Record 3:
44 mb moc -> 32.93 seconds
44 mb aws -> 9611.87 seconds
40 mb on MOC & 4 mb on AWS ->
Max (30.62, 779.53) + 4 seconds transfer + aggregation 24 seconds = total 807.53 seconds

Record 4:
88 mb moc -> 74.50 seconds
88 mb aws -> 17652.74 seconds
80 mb on MOC & 8 mb on AWS ->
Max (71.42, 1587.53) + 6 seconds transfer time + aggregation 26 seconds = total 1619.83 seconds

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.