The City College of New York | Department of Computer Science

THE GRADUATE CENTER CITY UNIVERSITY OF NEW YORK

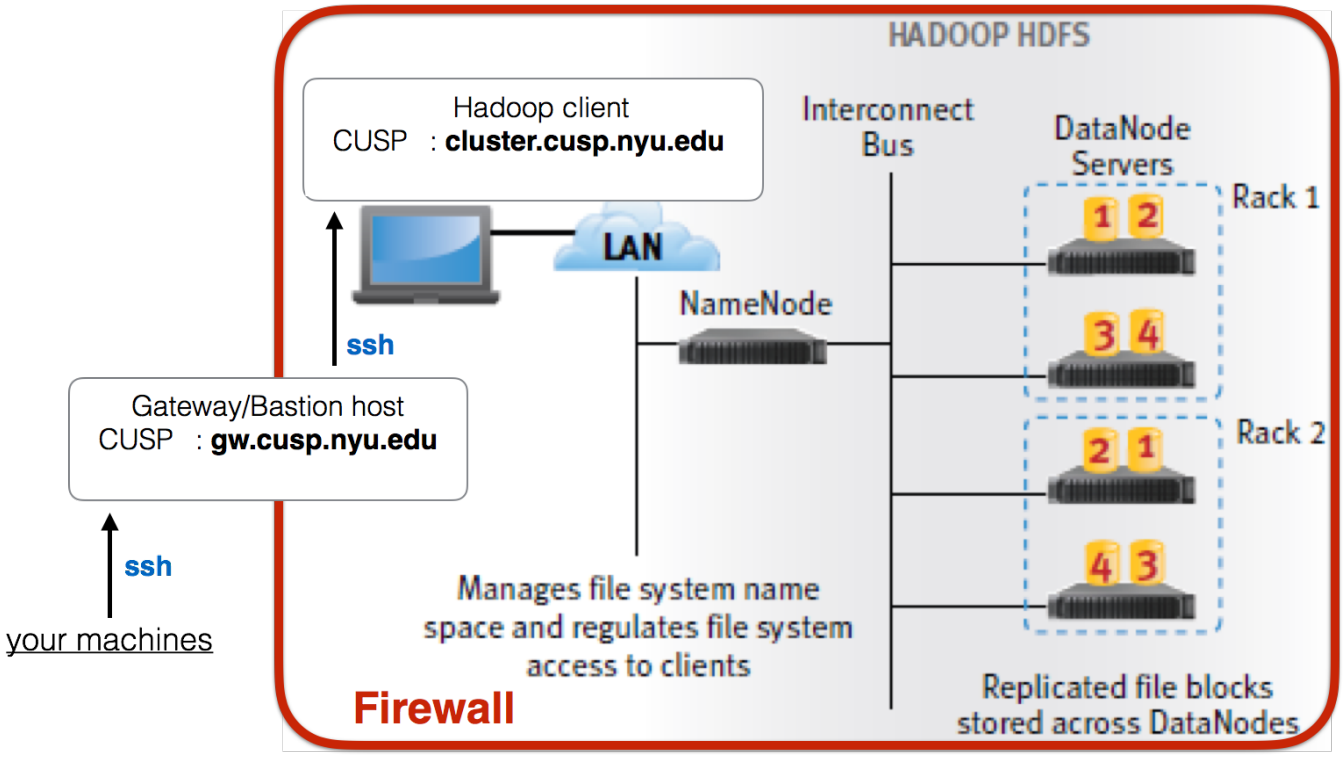
DSE I2450/CSc 84030: Big Data Analytics/Scalable Computation

SPRING 2021

# Lab 5 – Hadoop @ NYU/CUSP

In this lab, we're going to get familiar with the Hadoop computing environment by running Hadoop Streaming jobs on the clusters at NYU-HPC/CUSP. You should already have access to NYU CUSP cluster. Please make sure that you have a Terminal with SSH capability on your machine (e.g. Terminal on Linux/Mac OS X or Putty on Windows).

## General Access Diagram for the Hadoop Clusters @ NYU




In general, both CUSP and HPC clusters are protected by a firewall, thus, no direct SSH access are allowed. In order to access the Hadoop clusters, we must first login to a gateway or a bastion host, i.e. **gw.cusp.nyu.edu**. From this gateway, we can then access the Hadoop client machine, **cluster.cusp.nyu.edu**. This machine is equipped with a Hadoop environment for us to develop and run Hadoop jobs.

# TASK 1 – Access the Hadoop client node

Before we can perform any Hadoop work, we need to be on the Hadoop client node. In this case, we'll need to be on **cluster.cusp.nyu.edu**. The rest of lab will assume that we're using the CUSP cluster (the "$" sign indicates the bash prompt, only the contents after "$" are part of the commands). First, let's login into the CUSP cluster by running the following commands in your Terminal.

```
local$ ssh <USERNAME>@gw.cusp.nyu.edu

Last login: ….
[hvo@gw ~]$ ssh cluster

[hvo@cluster ~]$ which hadoop
/usr/bin/hadoop
```

Before running any Hadoop relate tasks, including Spark, we will need to authenticate again by running "kinit". This is for the cluster to know who is submitting a job. This credential could be different than the logged in user. You can check your current credential by using "klist".

```
[hvo@cluster ~]$ kinit
Password for hvo@CUSP.NYU.EDU:

[hvo@cluster~ ]$ klist
Ticket cache: FILE:/tmp/krb5cc_1829600001_RsErnR
Default principal: hvo@CUSP.NYU.EDU

Valid starting     Expires            Service principal
04/01/20 10:11:12  04/02/20 10:11:08  krbtgt/CUSP.NYU.EDU@CUSP.NYU.EDU
      renew until 04/08/20 10:11:08
```

# TASK 2 – Code preparation

Next, we need to clone the GIT repo for this lab:

```
[hvo@cluster ~]$ git clone https://github.com/hvo/bdm_spark_wc.git
Cloning into 'bdm_spark_wc'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 1), reused 4 (delta 0), pack-reused 0
Unpacking objects: 100% (8/8), done.
```

If you get prompted for a host check confirmation of RSA fingerprint, please type 'yes' to continue.

After this, we should see the following contents in the bdm_spark_wc directory:

```
[hvo@cluster ~]$ cd bdm_spark_wc
[hvo@cluster bdm_spark_wc]$ ls -l
-rw-r--r-- 1 htv210 users 259334 Mar 24 17:18 book.txt
-rw-r--r-- 1 htv210 users     10 Mar 24 17:18 README.md
-rw-r--r-- 1 htv210 users    259 Mar 24 17:18 wordcount.py
```

## TASK 3 – Run the WordCount example with Apache Spark

1. Upload data onto HDFS:

```
[hvo@cluster bdm_spark_wc]$ hadoop fs -put book.txt .
```

* Note: if you receive an error regarding your home folder does not exist, login to https://data.cusp.nyu.edu to have your folder automatically created.

2. You can verify the contents of your HDFS folder by running:

```
[hvo@cluster bdm_spark_wc]$ hadoop fs -ls
Found 1 item
-rw-------   3 htv210 users     259334 2016-03-01 22:17 book.txt
```

3. Run the WordCount example:

```
[hvo@cluster bdm_spark_wc]$ spark-submit wordcount.py
…
```

4. After that, we can query the result folder on HDFS:

```
[hvo@cluster bdm_spark_wc]$ hadoop fs -ls output
Found 3 items
-rw-r-----+  3 htv210 users          0 2020-03-24 17:13 output/_SUCCESS
-rw-r-----+  3 htv210 users      77919 2020-03-24 17:13 output/part-00000
-rw-r-----+  3 htv210 users      77075 2020-03-24 17:13 output/part-00001
```

5. And retrieve them to the login node:

```
[hvo@cluster bdm_spark_wc]$ hadoop fs -getmerge output word_counts.txt
[hvo@cluster bdm_spark_wc]$ head -n 3 word_counts.txt
('The', 268)
('Project', 79)
('EBook', 2)
```

# EXTRA 1 – How to upload large/compressed file to HDFS

It is common that we have large, and often compressed files on our local machines, or even from a remote source that we would like to upload onto HDFS for further processing using Hadoop. Of course, we could first copy the data to the login node first, then use **"hadoop fs –put"** to upload them to HDFS. This method works well indeed for data sets that are already available in the CUSP Data Facility since those data should be mounted on the machine the same way you access them on **compute.cusp.nyu.edu**. However, if you have new data from outside of CUSP, this could be an issue, as writing the data to disk could unnecessarily take up storage space and processing time. This is even more complicated if we have a compressed file on local storage, but would like to uncompress them on HDFS. In these cases, a good approach is to use pipes (|) to stream external contents and uncompress them if necessary onto HDFS.
An example here is that we would like to analyze the CitiBike Monthly Trip Data, which are available as ZIP files at: https://s3.amazonaws.com/tripdata/index.html. In particular, let's assume we wanted to have the trips of February, 2015 to be uploaded onto HDFS as an uncompressed CSV file. A naive approach would be:

```
[hvo@login ~]$ wget https://s3.amazonaws.com/tripdata/201502-citibike-tripdata.zip
--2016-03-24 00:02:52--  https://s3.amazonaws.com/tripdata/201502-citibike-
tripdata.zip
Resolving s3.amazonaws.com... 54.231.8.192
Connecting to s3.amazonaws.com|54.231.8.192|:443... connected.
...

[hvo@login ~]$ unzip 201502-citibike-tripdata.zip
Archive:  201502-citibike-tripdata.zip
  inflating: 201502-citibike-tripdata.csv

[hvo@login ~]$ hadoop fs –put 201502-citibike-tripdata.csv .
```

As you can see, this approach requires us storing and uncompress the file to local storage on **cluster.cusp.nyu.edu**, where we would have to remove later on. A more "streaming" way that cuts the "middle-man" storage is to use pipe that fetch the data, uncompress and upload it directly to HDFS as follows:

```
[hvo@login ~]$ wget -O - https://s3.amazonaws.com/tripdata/201502-citibike-
tripdata.zip | gunzip - | hadoop fs -put - 201502-citibike-tripdata.csv
--2016-03-24 00:02:52--  https://s3.amazonaws.com/tripdata/201502-citibike-
tripdata.zip
Resolving s3.amazonaws.com... 54.231.8.192
Connecting to s3.amazonaws.com|54.231.8.192|:443... connected.
...
```

Again, two observations: (1) the dash symbol "**-**" is being used in place of filename to indicate the standard input; and (2) **gunzip** is being used in place of **unzip** since the former can handle data from the standard input while the latter cannot.

NOTE: the command in red above need to be on 1 line, and there should be no space between the two lines (between '…citibite-' and 'tripdata…').

# EXTRA 2 – How to monitor Hadoop jobs with Resource Manager

Modern Hadoop distributions come a Web User Interface, HUE (Hadoop User Experience) for monitoring and managing resources. You can access this interface at https://data.cusp.nyu.edu, and select "Job Browser" on the upper right corner. You should be able to see all of your jobs being run

Sometime when our job fails to run, we need to inspect the error messages to correct the issue. Since NYU CUSP runs in cluster mode, all of the error messages (which are produced on the driver side) cannot be seen on the console. We need to look into the log files.

Yarn keeps the log messages for all application. We can look it up by the application ID. This ID is the one that appears on our screen when we execute our job (highlighted in red).

```
…
INFO  Client:54 - Application report for application_1584636743293_0034 (state: ACCEPTED)
INFO  Client:54 - Application report for application_1584636743293_0034 (state: RUNNING)
…
```

Using that id, we can show the log as follows, for example:

```
[hvo@login ~]$ yarn logs -applicationId application_1584636743293_0034
```

Though we could also manipulate our app through "yarn application". It is better to do that using HUE.