Clusters - Dumbo

Upcoming Events

Looking for one-on-one HPC help? Email us to book an HPC Consultation

New Pages

Spotted any errors or omissions in our new Wiki? Tell us about it!

- Data Transfer Nodes
- Acknowledge Statement
- Slurm Tutorial
- Programming for Biologists
- Licensed Software Available on the HPC Cluster
- Scratch Area Cleanup

Quick Links

HPC Home
Getting an account
Gentle Introduction to using HPC
Getting started on Prince
Prince How-to Articles
Logging in
  Windows
  Mac / Linux
Clusters and Storage
Prince (HPC)
Dumbo (Hadoop)

Brooklyn (OpenStack)

Dalma (NYU Abu Dhabi)

Transferring data to/from the clusters

Transferring data to/from Prince cluster using Globus

Submitting jobs with sbatch

Available software

Licensed Software Available on the HPC Cluster

Building Software

Slurm Tutorial

Tutorials

FAQs

Scratch Area Cleanup

Programming for Biologists

Acknowledge Statement

Research Gallery

HPC People

HPC Policies

(Tip: click "<<" at bottom left to close Confluence sidebar)
Dumbo - Hadoop cluster

**Dumbo** is our 48-node Hadoop cluster, running Cloudera CDH 5.15.0 (Hadoop 2.6.0 with Yarn).

All HPC users also have an account on Dumbo. (See [Getting or renewing an HPC account](#) for instructions to get an account. If you are enrolled in a class using the clusters you may already have an account, try logging in first).

Your $HOME directory on Dumbo is different that the $HOME on the HPC cluster, Prince. See [FAQ](#) for a way to set the command prompt to show which cluster you are on.

- **Dumbo - Hadoop cluster**
- **Hardware specifications**
- **Logging In**
  - Dynamic port forwarding
  - Hadoop Web UIs
- **Storage on Dumbo**
  - Huge Data Transfer between your workstation and Dumbo
  - File permissions and Access Control Lists
- **Using Hive**
- **Using Impala**
- **Using Python**
- **Using Spark**
- **Using Spark 2.2**
- **Hadoop Job Status**
- **Kafka**
- **Zeppelin**

### Hardware specifications

<table>
<thead>
<tr>
<th>System Name</th>
<th>Dumbo</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Master nodes</strong></td>
<td>&quot;babar&quot; and &quot;hathi&quot;</td>
</tr>
<tr>
<td>2x12-core Intel &quot;Haswell&quot; (c 2014) CPUs</td>
<td></td>
</tr>
<tr>
<td>256GB memory</td>
<td></td>
</tr>
<tr>
<td>8TB RAID1 disk</td>
<td></td>
</tr>
<tr>
<td><strong>Login and Compute nodes</strong></td>
<td>2 login/edge nodes: &quot;dumbo0&quot; and &quot;dumbo1&quot;. Logging in to &quot;dumbo&quot; gets a randomly selected login node.</td>
</tr>
<tr>
<td>44 compute nodes: &quot;compute-#-#&quot;.</td>
<td></td>
</tr>
<tr>
<td>2x8-core Intel &quot;Haswell&quot; (c 2014) CPUs</td>
<td></td>
</tr>
<tr>
<td>128GB memory</td>
<td></td>
</tr>
<tr>
<td>16x2 TB disk for HDFS</td>
<td></td>
</tr>
<tr>
<td><strong>Network</strong></td>
<td>10 Gb Ethernet</td>
</tr>
<tr>
<td><strong>Operating System</strong></td>
<td>Linux (Centos 6.9)</td>
</tr>
<tr>
<td><strong>File systems</strong></td>
<td>/home - 40GB quota per user, common $HOME</td>
</tr>
<tr>
<td></td>
<td>HDFS - 1.4 PB configured. 128MB block size</td>
</tr>
</tbody>
</table>

### Logging In

If you wish to use Hadoop Web UIs, you will need to setup dynamic port forwarding before logging in.

### Dynamic port forwarding
HDFS and other Hadoop components have Web-based user interfaces you can access with a web browser. However, all HPC clusters are behind the HPC bastion host and not visible to the NYU network or the outside world.

To make them visible, first configure Firefox to use a SOCKS proxy server.

1. Start Firefox
2. Under "Preferences", select "Advanced" and then "Network":

![Firefox Network Settings](image_url)
3. Next click “Settings” for “How Firefox connects to the Internet”

4. Select “Manual proxy configuration”, and set SOCKS Host to “localhost” on port 8118 (you can choose any port you like, but you must use the same port in the ssh command below)
5. Check “SOCKSv5” and “Remote DNS”
6. Set up your SSH tunnel as per the instructions here (Windows / Mac)

For Mac/Linux, add the following lines to `.ssh/config`:

```
# first we create the tunnel, with instructions to pass incoming
# packets on ports 8024, 8025 and 8026 through it and to specific
# locations
Host hpcgwtunnel
    HostName gw.hpc.nyu.edu
    ForwardX11 no
    LocalForward 8025 dumbo.hpc.nyu.edu:22
    LocalForward 8026 prince.hpc.nyu.edu:22
User NetID

# next we create an alias for incoming packets on the port. The
# alias corresponds to where the tunnel forwards these packets
Host dumbo
    HostName localhost
    Port 8025
    ForwardX11 yes
User NetID

Host prince
    HostName localhost
    Port 8026
    ForwardX11 yes
User NetID
```
For Windows:

Please follow the instructions on the link and then move on to the next step: Logging in from a Windows workstation

In the “Tunnels” dialog of your hpctunnel, add a new forwarded port, but this time the Source port is "8118" and check "Dynamic" instead of "Local"

7. Now, after you have logged in to hpc.nyu.edu, you can log in to dumbo in the same way you log in to Prince.

Hadoop Web UIs

Two very useful UIs are the Cloudera Manager (http://babar.es.its.nyu.edu:7180/) and Hue Interface (http://babar.es.its.nyu.edu:8888/). Please enter either link on the firefox web browser to access the UI.

Cloudera Manager is an administrator tool, but you can view the status of the cluster by logging in as "nyu" with the password "nyu".

The hue interface login is your NetId and Password.

Storage on Dumbo

Files Systems for usage:

The NYU Hadoop clusters have multiple filesystems for users files. Each filesystem is configured differently to serve a different purpose.

<table>
<thead>
<tr>
<th>Space</th>
<th>Environment Variable</th>
<th>Space Purpose</th>
<th>Flushed</th>
<th>Allocation (per user)</th>
</tr>
</thead>
<tbody>
<tr>
<td>/home</td>
<td>$HOME</td>
<td>Program development space; storing small files you want to keep long term, e.g. source code, scripts.</td>
<td>NO</td>
<td>40GB</td>
</tr>
<tr>
<td>/scratch</td>
<td>$SCRATCH</td>
<td>Best suited to large files.</td>
<td></td>
<td>5TB</td>
</tr>
</tbody>
</table>
Moving files between your workstation and HDFS

There are three ways to do this:

1. Use the SSH tunnel to scp between your workstation and $HOME, and use 'hadoop fs -put' and 'hadoop fs -get' to move between $HOME and HDFS. If you do this, please remove extraneous files for $HOME afterwards:

   **Transfer from workstation to HDFS**

   ```
   my_workstation$ scp my_file dumbo:
   my_workstation$ ssh dumbo
dumbo$ hadoop fs -put my_file
dumbo$ rm my_file
   ```

   **Transfer from HDFS to workstation**

   ```
   my_workstation$ ssh dumbo
dumbo$ hadoop fs -get my_file
dumbo$ exit
   my_workstation$ scp dumbo:/my_file .
   ```

2. Use the SSH tunnel and NFSGateway to scp files directly between your workstation and HDFS:

   ```
   my_workstation$ scp my_file dumbo:/mnt/hdfs/user/<my_netid>/
   ```

3. Use the Hue Interface file browser option which is located in the top dashboard. On the right right hand side you will have an option to upload and download files to and from your workstation.

### Huge Data Transfer between your workstation and Dumbo

Using Globus to transfer files to and from NYU HPC storage. Use Globus to transfer the data to /scratch on Prince first. The same /scratch folder is also available on dumbo login nodes.

- /scratch is shared between multiple clusters i.e., Prince and dumbo. But /home is different on Prince and dumbo.
- Globus server is only available on Prince. So, transfer data to /scratch with Globus on Prince and then copy the data from /scratch to your hdfs or /home folder of dumbo.

The Globus project aims at providing powerful tools for scientific data management, to help researchers to focus on their domain subjects and solve data intensive research problems. Globus has been grown maturely to enable grid computing by connecting computing resources distributed globally across organizational boundary. Universities, national laboratories and computing facilities are using services of Globus.

### File permissions and Access Control Lists

Users can share files with others using ACL's.
An access control list (or ACL) gives per-file, per-directory and per-user control over who have permissions on the files. You can see the ACL for a file or directory with the getfacl command:

```
$ hdfs dfs -getfacl /user/<net_id>/testdir
```

To modify permissions for files or directories, use setfacl.

```
$ hdfs dfs -setfacl -R -m user:<net_id>:rwx /user/<net_id>/testdir
$ hdfs dfs -setfacl -R -m default:user:<net_id>:rwx /user/<net_id>/testdir
```

**Using Hive**

Hive defaults to using the "default" database. This means that tables you create in this database might clash with tables others have created. Therefore, we highly recommend creating and using your own database, eg:

```
Hive CLI is deprecated and migration to Beeline is recommended.

-bash-4.1$ beeline
```

```
beeline> !connect jdbc:hive2://babar.es.its.nyu.edu:10000/
Enter username for jdbc:hive2://babar.es.its.nyu.edu:10000/: <net_id>
Enter password for jdbc:hive2://babar.es.its.nyu.edu:10000/: **********
0: jdbc:hive2://babar.es.its.nyu.edu:10000/> use <netid> ;
```

Please make sure to enter the Hive Heap size with your query for a MapReduce job.

Currently we do not have a way to automatically choose a suitable compute node to connect to - you should choose one at random. The Cloudera Manager WebUI can help, on the front page there are charts showing host CPU usage and host memory usage. It is wise to avoid hosts which already show high CPU or memory usage!

You can see a list of all compute hosts by clicking the "Hosts" link on the top dashboard of the Cloudera Manager WebUI main page.

**Using Impala**

To connect to impala use hostname where impala daemon is running. Infact, impala daemons are running on all slaves/datanodes on compute nodes of dumbo. So, connecting to any compute nodes, impala will work.

Here is the process to connect to Impala-shell.

```
-bash-4.1$ impala-shell
Starting Impala Shell without Kerberos authentication
Error connecting: TTransportException, Could not connect to login-2-1.local:21000
*******************************************************************************
Welcome to the Impala shell.
(Impala Shell v2.7.0-cdh5.9.0 (4b4cf19) built on Fri Oct 21 01:07:22 PDT 2016)
Want to know what version of Impala you're connected to? Run the VERSION command to find out!
*******************************************************************************
[Not connected] > connect compute-1-1;
Connected to compute-1:21000
Server version: impalad version 2.7.0-cdh5.9.0 RELEASE (build 4b4cf1936bd6cdf34fda5e2f32827e7d60c07a9c) [compute-1-1:21000] >
```

**Using Python**

Python defaults to version 2.6 on dumbo. Incase there is need to work on python with libraries i.e., NumPy, SciPy, Pandas, NLTK, Gensim, Geopandas, follow below steps which points to Python 2.7.11 with required libraries. Contact hpc@nyu.edu for installing new libraries.

NOTE: If you want to work with Hadoop Streaming, Refer Big Data Tutorial 1: MapReduce for MapReduce Hadoop Streaming execution process.
Using Spark

Spark is a data-processing framework that operates on the distributed data collections. Hadoop and Apache Spark are both big-data frameworks, which really serve the different purposes. Hadoop is essentially a distributed data infrastructure, It distributes massive data collections across multiple nodes within a cluster of commodity servers where as Spark is a data-processing tool.

Refer Big Data Tutorial 3 to learn more about Spark.

Here is the process to work with Spark on Dumbo:

```bash
-bash-4.1$ spark-shell
Welcome to

    __              __
   /  \__  ___ _____/ /__
  /_\  \ /  \_\/  '_/
 /___/  \__/ \_/_/ \_
   \_

Using Scala version 2.10.5 (Java HotSpot(TM) 64-Bit Server VM, Java 1.7.0_79)
Type in expressions to have them evaluated.
Type :help for more information.
Spark context available as sc (master = yarn-client, app id = application_1472492286365_0001).
SQL context available as sqlContext.
scala>
```

Using Spark 2.2

Process to work with Spark 2.2 on Dumbo:

```bash
-bash-4.1$ spark2-shell
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://10.0.255.254:4040
Spark context available as 'sc' (master = yarn, app id = application_1516209193107_0032).
Spark session available as 'spark'.
Welcome to

    __              __
   /  \__  ___ _____/ /__
  /_\  \ /  \_\/  '_/
 /___/  \__/ \_/_/ \_
   \_

Using Scala version 2.11.8 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_152)
Type in expressions to have them evaluated.
Type :help for more information.
scala>
```
Hadoop Job Status

Please find the list of current running jobs on hadoop cluster using 'Yarn' script. Running the yarn script without any arguments prints the description for all commands.

```
yarn application -list
```

To kill a currently running job because the submitted job started malfunctioning or in worst case scenario, job is stuck in infinite loop. Get the Job ID and then Kill it as given below.

```
yarn application -kill <application_ID>
```

Kafka

Kafka is a messaging system. In the real world data exists on many systems in parallel, all of which need to interact with Hadoop and with each other. The situation quickly becomes more complex, ending with a system where multiple data systems are talking to one another over many channels. Infact, each of these channels require their own communication method to move the data. So, we can use Kafka as a central messaging bus. All incoming data is first placed in Kafka and all outgoing data is read from Kafka which is called as “Topic”. Kafka centralizes communication between “producers” of data and “consumers” of that data.

The basic architecture of Kafka is organized around a few key terms: topics, producers, consumers, and brokers. All Kafka messages are organized into topics. If you wish to send a message you send it to a specific topic and if you wish to read a message you read it from a specific topic. A consumer of topics pulls messages off of a Kafka topic while producers push messages into a Kafka topic. Lastly, Kafka, as a distributed system, runs in a cluster. Each node in the cluster is called a Kafka broker.

Zeppelin

Apache Zeppelin is a web-based interactive computational environment that could use Apache Spark as a backend. In some sense it is like the IPython Notebook. Zeppelin is installed on dumbo.

Dumbo is a yarn cluster, not a standalone spark cluster.

Here is the process to work with Zeppelin on Dumbo.

```
# Create personal directories and copy configuration over
mkdir -p $HOME/zeppelin/{conf,logs,notebook,run,webapps}
cp /share/apps/zeppelin/0.7.1/conf/* $HOME/zeppelin/conf/
```

See [users] section in $HOME/zeppelin/conf/shiro.ini for the default user/password and the instruction to change password. The procedures to start and stop a Zeppelin server on dumbo login nodes are:

```
# Start Zeppelin daemons.
/share/apps/zeppelin/0.7.1/bin/zeppelin-daemon.sh --config $HOME/zeppelin/conf start

# Stop the daemons
/share/apps/zeppelin/0.7.1/bin/zeppelin-daemon.sh --config $HOME/zeppelin/conf stop
```

Suppose you have dynamic port forwarding set up on your workstation, you can browse your notebook using URL:

```
# Note: Verify the hostname using command "hostname" at command line.
http://128.122.215.51:<port>/  if running on login-1-1
http://128.122.215.52:<port>/  if running on login-2-1
```

Alternatively if you have X11 set up (see instructions for Windows and Mac / Linux computers), you can also run firefox on dumbo login node, and get firefox window popping up on your workstation:

```
module load firefox/45.8.0
firefox <URL as above>
```
Please refer back to this page in the near future for instructions on how to use Oozie, Flume and Kafka.