INSTITUTE FOR APPLIED COMPUTATIONAL SCIENCE AT HARVARD UNIVERSITY

HARVARD School of Engineering and Applied Sciences

Guide: MPI on AWS Part 2

Ignacio M. Llorente, Simon Warchol

v3.0 - 3 March 2021

Abstract

This is a guideline document to show the necessary actions to set up and use a MPI cluster on AWS Ubuntu (18.04) instances. It includes its configuration on a single node for prototyping and debugging, and its deployment on multiple nodes for production.

Requirements

- **First you should have followed the Guide "First Access to AWS"**. It is assumed you already have an AWS account and a key pair, and you are familiar with the AWS EC2 environment.
- We strongly recommend you use the same instance type used for the shared memory (OpenMP) infrastructure guide so you can compare performance results. The results in this guide have been obtained on a **t2.2xlarge** instance with 8 vCPUs, which is the instance type recommended in the homework assignment.
- The files needed to do the exercises are available for download from **Canvas**.
- There are a wide variety of MPI implementations out there. You are free to use any implementation you wish. **MPICH** is a widely-used implementation of MPI that is developed primarily by Argonne National Laboratory in the United States.

Acknowledgments

The author is grateful for constructive comments and suggestions from David Sondak, Charles Liu, Matthew Holman, Keshavamurthy Indireshkumar, Kar Tong Tan, Zudi Lin,Nick Stern, and Dylan Randle.

1. MPI in a Multi-Node Cluster

This section illustrates how to deploy a simple MPI cluster consisting of only two nodes (VMs) interconnected with a virtual private network. You can easily extend these steps to build a cluster with a higher number of nodes.

2.1. Deploy a Virtual Cluster

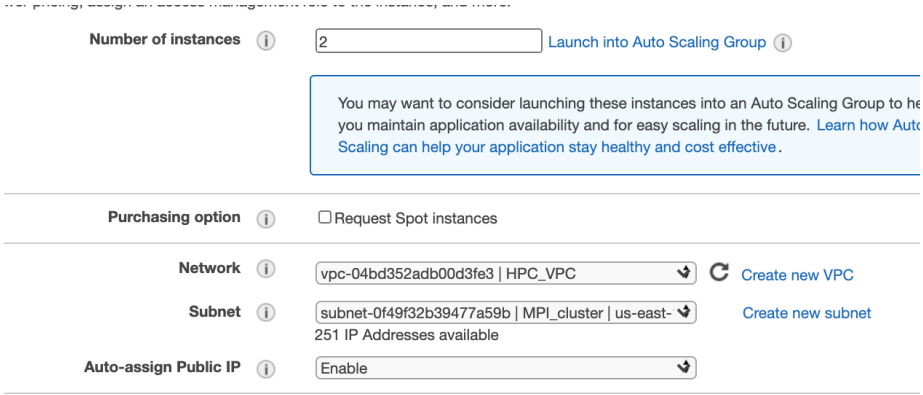
First step is to deploy a cluster and their private interconnection..

- Launch two instances with “Ubuntu Server 18.04” as AMI.
- In Step 3 “Configure Instance Details” of the wizard, create a new VPC (`HPC_VPC`) and a subnet (`MPI_cluster`) for private interconnection between both nodes. A "Virtual Private Cloud" is a method to isolate VMs inside an isolated private network. Other instances can't see instances that are inside a VPC.

  A VPC is denoted by a subnet mask. For example, we specified `172.30.8.0/24`, which means any instances inside this VPC will have an ip `172.30.8.X` where X can be anything between 2 to 254. A VPC can have following global components:

  - A DHCP option set, which is a server that assigns dynamic ips
  - Internet gateway to route non-local traffic
  - One or more subnets, which are sub-networks inside a VPC. We defined `mpi_cluster` to be `172.30.8.0/24`
  - One or more routing tables, which defines what to do with network traffic
  - One or more network ACLs

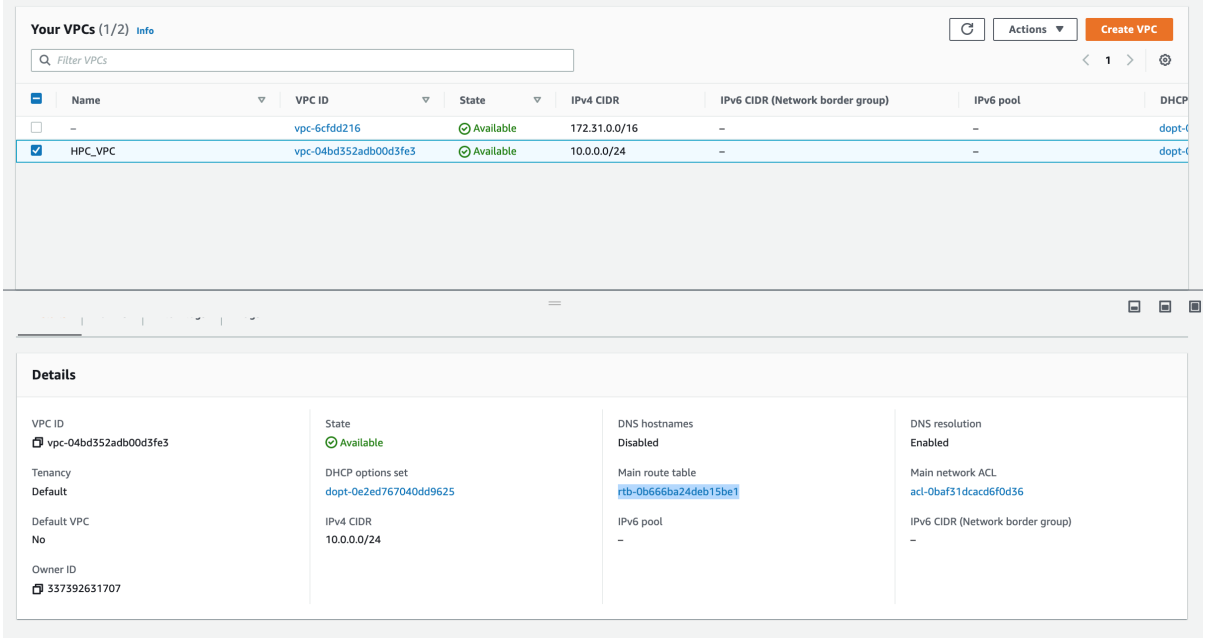
  Do not forget to enable `auto assign public ip` so both VM instances get a public ip through a NAT translation. This is needed to access the instances from outside. However if you run `ip addr show` inside your VM you'll see it has a private ip. The instance itself is not aware of its public ip. Communication between instances should be performed through their private ip for cost and performance reasons. Remember that these public IP addresses are associated with the instances until they are stopped or terminated.



- With the network and subnet selected, you may now spin up the two instances. Do **NOT** do any

configuration of the security groups for the instances.

- To enable the VPC to be connected from the internet, we need to attach an internet gateway to the VPC. (NOTE: it is very important to do this step or you will not be able to connect to your node via SSH)
- In the VPC dashboard (Services Dropdown -> VPC), click on “Internet Gateways”.
- Create a new internet gateway. Name it something like `HPC_VPC_Gateway`
- Right click on the created internet gateway and select “Attach to VPC”.
- Select the VPC you created earlier and attach the internet gateway to your VPC.
- Go back to your VPC and select and view the “Route table” associated with your VPC.



- Click on the routes tab, and add a route.
- For the Destination field, add “0.0.0.0/0”. For the Target field, add the internet gateway you just created.
- This will allow the VPC to be accessed from the internet.

If after starting the VMs you are not able to connect through ssh, check section 4 about troubleshooting in Guide “First Access to AWS”.

If you still can’t connect, then you should make sure that you have attached the appropriate internet gateway to the VPC. You may want to follow the steps at Troubleshooting Connecting to Your Instance to attach the the IGW to the VPC. Check out the bullet point labeled [EC2-VPC].

- In two different terminal windows, SSH into each of your two instances.
- Arbitrarily designate one of those instances “master” and one “node1” and take note of which

instance is which as well as the Public and Private IP of these instances. For instance

Hostname	**Public IP** (changes in each reboot)	**Private IP**
master	52.73.170.178	172.30.8.6
node1	54.173.242.68	172.30.8.11

On the master VM update `/etc/hosts` to include

```
127.0.0.1 localhost
172.30.8.6 ip-172-30-8-6 master
172.30.8.11 ip-172-30-8-11 node1
```

On the node1 VM update `/etc/hosts` to include

```
127.0.0.1 localhost
172.30.8.11 ip-172-30-8-11 node1
172.30.8.6 ip-172-30-8-6 master
```

2.2. Configure SSH

Second step is to allow remote secure interconnection between the nodes.

- Create a new user named `mpiuser` in both VMs, you can use default values. You can set an arbitrary password.

  ```
  master$ sudo adduser mpiuser

  master$ sudo adduser mpiuser sudo

  node1$ sudo adduser mpiuser

  node1$ sudo adduser mpiuser sudo
  ```

- Your machines are going to be talking over the network via ssh. The ssh server is already installed in the AWS VMs
- In order to simplify the configuration you can allow ssh password authentication. In both VMs modify

  `/etc/ssh/sshd_config`

  by changing this line

  ```
  # Change to no to disable tunnelled clear text passwords
  PasswordAuthentication yes
  ```

It should be the first uncommented line, around line 56

Do not forget to restart the service on both machines.

```
master$ sudo service ssh restart
```

- Now you must be able to login to other machines, at which you will be prompted to enter the password of the username.

```
master$ su - mpiuser
mpiuser@master$ ssh mpiuser@node1
```

You can try the same from `node1` to master

- To enable easier login, we generate keys and copy them to other machines' list of authorized_keys.

```
mpiuser@master$ ssh-keygen (use default values)
Generating public/private rsa key pair.
Enter file in which to save the key (/home/mpiuser/.ssh/id_rsa):
Created directory '/home/mpiuser/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/mpiuser/.ssh/id_rsa.
Your public key has been saved in /home/mpiuser/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:paRkgkENmQ+VakcXlG82JgP16Q9CPrKLjFxOCzO6sKE mpiuser@ip-172-30-8-6
The key's randomart image is:
+---[RSA 2048]----+
| .+*.++o         |
|  +o= o. .       |
|  .=.o+.+ .      |
|  o o*o+*o       |
| . .. =*S.       |
|     o o o       |
|o+ o.     .      |
|==B...           |
|Eooo.            |
+----[SHA256]-----+

mpiuser@master$ ssh-copy-id mpiuser@node1
```

This should be also be done in `node1`

```
node1$ su - mpiuser
```

```
mpiuser@node1$ ssh-keygen
```
(use default values)

```
mpiuser@node1$ ssh-copy-id mpiuser@master
```

- Now, to enable passwordless ssh (you do not need to do this step if you did not include a passphrase during creation of the rsa key pair)

```
mpiuser@master$ eval `ssh-agent`
mpiuser@master$ ssh-add ~/.ssh/id_rsa
```

- Check that you can ssh between the machines without using a password. You must ssh at least once to all the nodes that are connected to the machine that you are running the MPI commands.

2.3. Configure NFS

Now let us configure NFS so the master can export a directory which the client mounts to exchange data.

Firstly let us configure the **NFS-Server** in the master

- Install the required packages in the master

```
master$ sudo apt-get install nfs-kernel-server
```

- Now, log in as `mpiuser` and create a folder by the name “cloud” that we will share across in the network

```
master$ su - mpiuser
mpiuser@master$ mkdir cloud
```

Now go back to master.

- To export the cloud directory, add the line

```
/home/mpiuser/cloud *(rw,sync,no_root_squash,no_subtree_check)
```

to the file `/etc/exports`

by executing

```
master$ sudo vi /etc/exports
```

- After you have made the entry, run the following.

```
master$ sudo exportfs -a
```

Run the above command, every time you make a change to `/etc/exports`

If required, restart the nfs server

```
master$ sudo service nfs-kernel-server restart
```

Secondly let us configure the **NFS-Client** in the node1

- Install the required packages

```
node1$ sudo apt-get install nfs-common
```

- Create a directory in the client's machine with the same name cloud

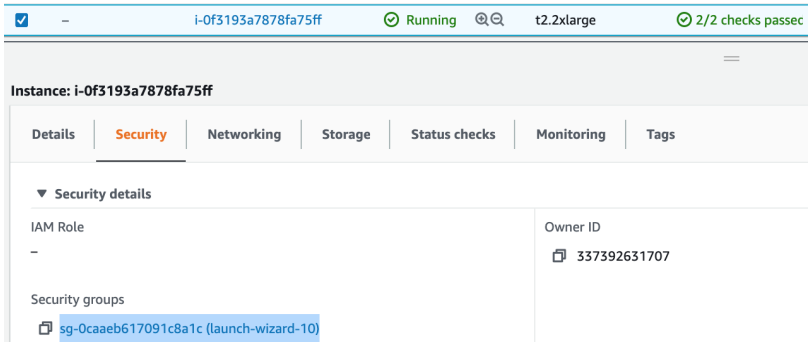
```
node1$ su - mpiuser

mpiuser@node1$ mkdir cloud
```

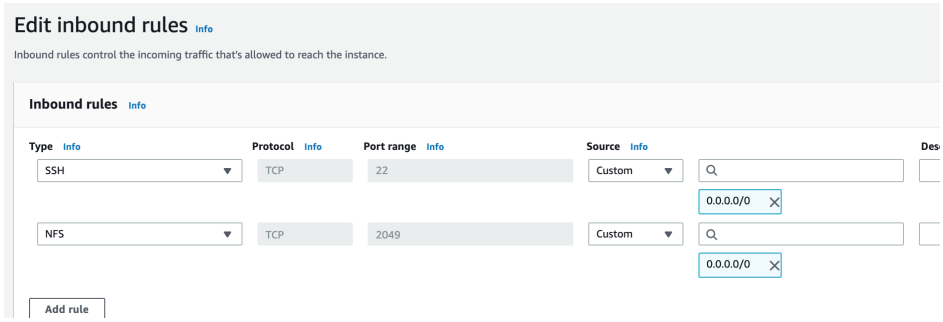
- And now, mount the shared directory like

```
node1$ sudo mount -t nfs master:/home/mpiuser/cloud /home/mpiuser/cloud
```

If this command generates a time-out, check that the security group of the VMs allow NFS connections. You can do this by simply adding a rule to the inbound rules of the security group for the VMs. The security group should be linked from the "Security" tab for either of the instances on the EC2 dashboard.



The new inbound rule should be of type NFS.



- To check the mounted directories

```
node1$$ df -h
Filesystem                  Size  Used Avail Use% Mounted on
udev                        3.9G     0  3.9G   0% /dev
tmpfs                       799M  8.6M  790M   2% /run
/dev/xvda1                  7.7G  1.2G  6.5G  16% /
tmpfs                       3.9G     0  3.9G   0% /dev/shm
tmpfs                       5.0M     0  5.0M   0% /run/lock
tmpfs                       3.9G     0  3.9G   0% /sys/fs/cgroup
tmpfs                       799M     0  799M   0% /run/user/1000
master:/home/mpiuser/cloud  7.7G  1.2G  6.5G  16% /home/mpiuser/cloud
```

- To make the mount permanent so you don't have to manually mount the shared directory every time you do a system reboot, you can create an entry

```
#MPI CLUSTER SETUP
master:/home/mpiuser/cloud /home/mpiuser/cloud nfs
```

in your file systems table - i.e., `/etc/fstab` file by executing

```
node1$ sudo vi /etc/fstab
```

2.4. Install and Configure MPI

You are now ready to install and configure MPI on the cluster.

- Install `mpich` on both systems

```
master$ sudo apt-get update
master$ sudo apt-get install libcr-dev mpich mpich-doc
node1$ sudo apt-get update
node1$ sudo apt-get install libcr-dev mpich mpich-doc
```

- To check the `mpich` installation is successful run following command in the terminal

```
$ mpiexec --version
```

- Upload to the master VM the `mpi_sc.c` code, compile it with `mpicc`, and run the code with

multiple processes (on the master host).

```
mpiuser@master$ mpicc mpi_sc.c -o mpi_sc
mpiuser@master$ mpirun -np 2 ./mpi_sc
Hello world from process 0 of 2
Hello world from process 1 of 2
```

- To run it on multiple nodes you have to configure the MPI environment to use ports 10000-10100

```
mpiuser@master$ export MPICH_PORT_RANGE=10000:10100
mpiuser@master$ export MPIR_CVAR_CH3_PORT_RANGE=10000:10100
```

- You have to allow these connections in the security group of the VMs

Type	Protocol	Port Range	Source	Description
Custom TCP Rule	TCP	10000 - 10100	0.0.0.0/0	
SSH	TCP	22	0.0.0.0/0	
NFS	TCP	2049	0.0.0.0/0	

- Now you can run your applications on multiple nodes. Upload the mpi_sch.c code to master, compile it with `mpicc`, copy the executable to the `cloud` directory, and run the code from the `cloud` directory with multiple processes on both hosts.

```
mpiuser@master$ mpicc mpi_sch.c -o mpi_sch
mpiuser@master$ cp mpi_sch cloud
mpiuser@master$ cd cloud
mpiuser@master$ mpirun -np 4 -hosts master,node1 ./mpi_sch
Hello world!  I am process number: 2 on host master
Hello world!  I am process number: 0 on host master
Hello world!  I am process number: 1 on host node1
Hello world!  I am process number: 3 on host node1
```

Stop your instances when are done for the day to avoid incurring charges
Terminate them when you are sure you are done with your instance