Parallel Computing Using Amazon EC2

Abhimanyu Dubey, Virginia Tech

0 Introduction

This tutorial provides instructions on setting up and running parallel applications on your laboratory computer and/or Amazon EC2 with MIT StarCluster using OpenMPI.

The applications the accompanying scripts were tested for are (a) C++ applications written using the GraphLab API and OpenMPI, and (b) MATLAB applications that parallelize using the MATLAB Parallel Computing Toolbox, but any other application written using OpenMPI libraries or otherwise will run correctly.

To run code on Amazon EC2, StarCluster developed by the MIT Star group is used to efficiently handle the setting up and execution of jobs on clusters.

The code mentioned here can be cloned from
https://github.com/abhimanyudubey/vtmlp.git

1 Setting Up Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale computing easier for developers. If you don't already have an Amazon Web Services account, you can set up your account at http://aws.amazon.com or contact your professor for their lab account details.

After signing up, navigate to 'Security Credentials' on the left sidebar. Under 'Access Credentials', you would be required to create on each of Access Keys, X.509 certificates, and Key Pairs. These keys would be required to securely connect to your remote machine. Different Amazon services require different authorization methods, and EC2 and Elastic Block Store (EBS) require the ones mentioned. If you have already obtained the access keys, certificates and key pairs from your professor, you can skip to the next section (VT-MLP researchers can contact Dr. Batra for the above information).

1.1 Access Key ID and Secret Access Key

There is already one Access Key present on the creation of your account. Copy the Secret Access Key (it is hidden, click 'show' to view it) and Access Key ID to a secure location.

1.2 X.509 Certificate

Create a new certificate. This will open up a dialog box with information about the certificate created. This certificate is an RSA encrypted certificate and has a private-key and a certificate file. You need to download both the Private Key and the X.509 certificate and store it in a secure location (preferably create `~/.amazon/` and store both there).

1.3 Key Pairs

To use Amazon EC2 we would require to create EC2 keypairs and not CloudFront keypairs. To create a new EC2 keypair navigate to the Amazon AWS Console at https://console.aws.amazon.com

Navigate to EC2. In the left sidebar, under 'Network and Security' you'll find 'Key Pairs'. Create a new key pair and store it in a secure location (it could be where you have stored the X.509 certificate).

1.4 AWS Account ID

Now, reopen the AWS 'security credentials' page, and copy your AWS account ID (present under 'Account Identifiers' at the bottom of the page), and store it in a secure location.

It is mandatory that you have all the above information (along with file paths for keys and certificates) before proceeding to the installation of Amazon EC2 Command Line tools and MIT StarCluster.

2 Installing EC2 API/AMI Tools and MIT StarCluster

2.1 Amazon API/AMI Tools

To access EC2 machines and create custom AMIs (Amazon Machine Images) from the terminal, we require the EC2 API and AMI tools. A shell script that installs and configures both is `shell/install-ec2.sh`. This script would download and install all prerequisites for API and AMI tools and configure the security credentials to set up the Amazon EC2 command line interface for your computer. To run, execute the following commands:

```
cd <path to shell/install-ec2.sh from repository>
chmod 111 shell/* #to provide execute access to all scripts present.
./shell/install-ec2.sh   <AccessKey>   <SecretKey>   <UserID>   <X509-Certificate-
Location> <X509-PK-Location> <EC2-Keypair-Name> <EC2-Keypair-Location>
```

Here,
AccessKey : The Access Key ID in section 1.1

SecretKey:	Secret Access Key in section 1.1
UserID :	AWS User ID from section 1.4
X509-Certificate-Location :	Location of the downloaded certificate file from section 1.2
X509-PK-Location:	Location of the private key downloaded from section 1.2
EC2-Keypair-Name:	Name of the key pair created in section 1.3
EC2-Keypair-Location:	Location of the key pair created in section 1.3.

2.2 Installing MIT StarCluster

StarCluster is an open source cluster-computing toolkit for Amazon's Elastic Compute Cloud (EC2) released under the LGPL license. It has been designed to automate and simplify the process of building, configuring, and managing clusters of virtual machines on Amazon's EC2 Cloud. It allows anyone to easily create a cluster computing environment in the cloud suited for distributed and parallel computing applications and systems.

To install StarCluster, cd into the `shell/` directory, and run `./install-starcluster.sh`

Note: Python is required to install and run StarCluster.

This script will install and configure MIT StarCluster with the default cluster profile *default.* This profile has 20 t1.micro machines with the default VT-MLP API loaded on it (Ubuntu 12.04, x86_64, support for StarCluster and GraphLab built).

Note: You might have to run the shell scripts as a super user (sudo) if there are any permissions errors.

3 Running Jobs on EC2 Machines using StarCluster

3.1 Connecting to EC2 using StarCluster

To run tasks on EC2 Machines using StarCluster, you will first have to create a cluster using the cluster profiles present in your StarCluster configuration (which is located at `~/.starcluster/config` by default). A guide to help understand this configuration is present at `help/guide-starclusterconfig` , which is a sample configuration file with comments for each entry.

To launch a cluster *mycluster* using the default cluster profile, type

```
starcluster start mycluster
```

This will start a cluster called *mycluster* using the default cluster profile as listed in *config*. The cluster profile determines the number of computers in the cluster and the default machine image to be used by the cluster. For more information about StarCluster commands, visit http://star.mit.edu/cluster/docs/latest/manual/index.html.

After starting a cluster, you can SSH into any node present in the cluster. To SSH into the master node, type

```
starcluster sshmaster mycluster
```

For additional information about starcluster SSH commands, type `starcluster help` at the console.

3.2 Using the Open Grid Scheduler

To connect to other machines present in the cluster, StarCluster uses the Open Grid Scheduler (formerly known as the Sun Grid Engine). Embarrassingly parallel tasks can be carried out simply by submitting jobs to separate machines.

StarCluster manages the installation and setting up of the scheduler. It requires a shared storage access, and by default StarCluster shares the /home folder of the cluster. Hence all files present within /home would be shared across machines.

For any additional EBS (Elastic Block Store) Volumes that you may have attached, you can view if they are attached to all machines by following the instructions present at http://star.mit.edu/cluster/docs/0.93.3/manual/getting_started.html#ensure-ebs-volumes-are-mounted-and-nfs-shared-optional.

Once you have SSHed into the master node, you can run a few basic commands to submit jobs to nodes in the cluster, which are:

qstat - Show job/queue status

no arguments, Show currently running/pending jobs
-f Show full listing of all queues
-j Shows detailed information on pending/running job
-U Shows current jobs by user

qhost - Show job/host status

no arguments, Show a table of all execution hosts and information about their configuration
-l attr=val Show only certain hosts
-j Shows detailed information on pending/running job
-q Shows detailed information on queues at each host

Using Grid Engine

The main submit commands are qsub, qrsh and qtcsh.

qsub - submit scripts

Started with no arguments it accepts input from STDIN (^D to send submit input)
-cwd Run the job from the current working directory
(Default: Home directory)
-v Pass the variable VAR (-V passes all variables)
-o Redirect standard output (Default: Home directory)
-e Redirect standard error (Default: Home directory)

Example:

```
qsub -cwd -v SOME_VAR -o /dev/null -e /dev/null myjob.sh
```

In general, qsub is used for traditional batch submit, that is where I/O is directed to a file. Note that qsub only accepts shell scripts, not executable files.

qrsh

Qrsh acts similar to the rsh command, except that a host name is not given. Instead, a shell script or an executable file is run, potentially on any node in the cluster. I/O is directed back to the submitter's terminal window. By default, if the job cannot be run immediately, qrsh will not queue the job. Using the '-now no' flag to qrsh will allow jobs to queue. Note that I/O can be redirected with the shell redirect operators. For example, to run the uname -a command:

```
qrsh uname -a
```

The uname of some machine the scheduler selects in the cluster will then be displayed on the submitting terminal. To redirect the output,

```
qrsh uname -a > /tmp/myfile
```

The output from uname will be written to /tmp/myfile on the submitting host. To allow the command to queue:

```
qrsh -now no uname -a
```
(command tutorial from http://gridscheduler.sourceforge.net/howto/basic_usage.html)

3.3 Using OpenMPI on EC2

To use OpenMPI in conjunction with open grid scheduler, you can compile the code using mpicxx and run using mpirun. For example, to run a program *abc* compiled with MPI, we can type

```
mpirun –n <no of instances>  -hostfile <path to hostfile> ./abc
```

wFor a detailed tutorial, follow this link: http://star.mit.edu/cluster/docs/latest/guides/sge.html#submitting-openmpi-jobs-using-a-parallel-environment. (Note: SGE is the former name for Open Grid Scheduler). This method should be followed for execution of GraphLab programs as well.