

Cloud Computing in ISSM using the Amazon EC2 Cloud Solution

Eric Larour

Last update on June 29, 2013

Contents

1 Introduction

We rely on the Amazon Elastic Compute Cloud solution (EC2) to run ISSM on the cloud (?). We provide everything required in the externalpackages and this documentation, to run ISSM, provided an amazon EC2 account has already been setup by the user. EC2 is the backbone for our cloud computing, however, it does not provide facilities to deploy an MPI cluster on an instance of the cloud. To deploy such a cluster, we rely on the third party software, StarCluster (?). This library is a suite of python scripts which communicate with the EC2 servers, and automatically setup a cluster, given a configuration file, using the EC2 account of the user. Once the cluster is spun-up, it can be used to carry out computations using the ISSM framework. To facilitate the install of ISSM, we provide images (AMI), which can be loaded directly, hence avoiding the need for extensive installatin/compilation. Such images are now private, but could potentially be made available to the ISSM community of users in the near-future.

2 Installation

We assume here that you have setup an EC2 account, and that you have the following: the AWS access key id, and the AWS secret access key. Nothing else is needed on the EC2 side.

You need to also install StarCluster. First, be sure that you have successfully installed python from the issm externalpackages, otherwise, you will get permission issues upon install of the StarCluster package. Then just run the install.sh script in the StarCluster external package of issm/trunk. This should install all the scripts necessary to run StarCluster successfully.

3 Configuration

StarCluster needs a configuration file, which will be used to store information proper to your EC2 account, and templates for the clusters you will be spinning up. This file is well described on the StarCluster website, found here [StarCluster user manual](#) For ease of use, we have created an ISSM StarCluster configuration file, that you will find in the issm-jpl/proj-group/CloudComputing directory. In order to use this configuration file, you should create an alias in your local settings:

```
alias st='starcluster -c $PATH_TON_CONFIG_FILE
```

For the ISSM team, the StarCluster configuration file should be replaced by:
 issm-jpl/proj-group/CloudComputing/starcluster.config.

3.1 StarCluster configuration file for ISSM

Here are some of the sections of the starcluster.config file, which we explain within the framework of the ISSM runs:

```
[aws info]
AWS_ACCESS_KEY_ID = put_you_aws_access_key_id_here
AWS_SECRET_ACCESS_KEY = put_your_aws_secrate_access_key_here
AWS_USER_ID = put_your_ec2_acount_logging_here
AWS_REGION_NAME = us-east-1
AWS_REGION_HOST = ec2.us-east-1.amazonaws.com
```

This section holds your settings for the AWS account. You need to provide your access key id and secret access key, along with the user id for the account. The region name and region hosts determine on which region of the Amazon EC2 cloud you will be running your cluster. Beware, some

of the most powerful machines are not always available in all regions. Typically, the us-east-1 has the best instance types, the cc2.8xlarge machine. This is what we will rely on here.

```
[key issm-jpl]
KEY_LOCATION=~/.ssh/issm-jpl.rsa
```

This section specifies the locations where the ssh key for the EC2 account is located. Before you can start using the EC2 account with StarCluster, StarCluster needs to create this key. Run the following command to do so:

```
st createkey issm --outputfile=issm.rsa
```

Once the key is created, move it to the *KEY_LOCATION* specified in the section above. Here, we would move *issm.rsa* to the *ssh* directory in your home. If the key has already been created, and you don't have it, just download it from your EC2 account console, or request it from someone who owns the account and can send you the *rsa* key. Do not attempt to destroy the key by running *stremovekeyissm.rsa*, as this will also deactivate all the other users that were using this key.

```
[cluster issm-jpl]
KEYNAME = issm-jpl
CLUSTER_SIZE = 1
CLUSTER_USER = username
CLUSTER_SHELL = bash
NODE_IMAGE_ID = ami-59106030 root
NODE_INSTANCE_TYPE = cc2.8xlarge
PLUGINS = createusers-jpl, mpich2
#SPOT_BID = 0.27

[plugin createusers-jpl]
setup_class = starcluster.plugins.users.CreateUsers
usernames = username
download_keys = True

[plugin mpich2]
SETUP_CLASS = starcluster.plugins.mpich2.MPICH2Setup
```

These sections really describe your cluster settings. *CLUSTER_SIZE* is the number of instances that will be launched for the cluster. A special EC2 cluster group will be created, ensuring the all the instances of your cluster have the best connectivity. The user of the cluster will have a corresponding user name created.

The *NODE_INSTANCE_TYPE* can be found here: [EC2 instance types](#). The most powerful one for the purpose of running ISSM is the cc2.8xlarge, which can be found on the us-east1 region.

The *NODE_IMAGE_ID* is the image (residing on the EC2 servers) from which the instances will be created. This image is a template for the cluster you will be launching. Here, we use ISSM images that have been created specifically. Please ask the ISSM team for an image if you don't want to create one from scratch.

SPOT_BID is a setting that you can activate to make a spot request at a certain price. The EC2 cloud will provide you instances once the spot price goes below your requested price. This is a way to run low-cost solutions using the market driven EC2 prices.

The plugging sections ensure that the usernames are created and *mpich2* is run as the backbone of the MPI cluster.

4 Running ISSM with StarCluster

In order to run ISSM on the EC2 cloud, you will need to create an instance of your cluster, by running the following:

```
st start -c issm-jpl issm1
```

This will spin-up an issm-jpl type cluster, named issm1. Once the cluster is created, you can log into it by doing the following:

```
st sshmaster issm1 -u username
```

You can run issm locally, or better, rely on the @cloud cluster class already implemented in issm. The only thing to do to run on the issm1 cluster is to activate the @cloud class in issm using the 'issm1' name:

```
md.cluster=cloud('name','issm1','np',num_cpus_requested);
```

Once you are done running on the cluster, terminate the cluster by doing:

```
st terminate issm1
```