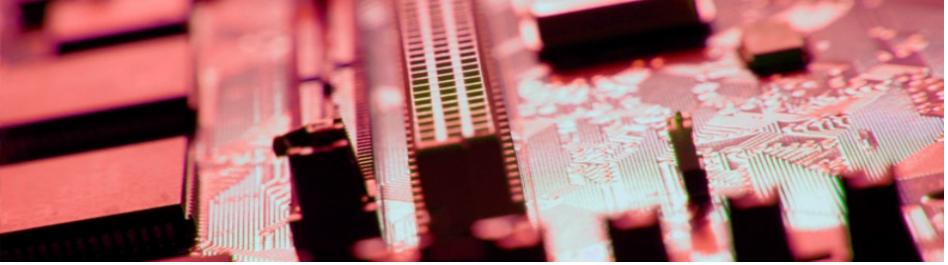


LCI workshop
on high performance clustered computing

Build-A-Cluster Workshop

A hands-on tutorial session for building a cluster to support parallel scientific computing codes

Presented by

Stephen Lien Harrell – SLH@purdue.edu

Amiya Maji – amaji@purdue.edu

What are we doing here anyway?

- This is:
  - A guide to building a simple scientific computing cluster
  - An outline for deploying a high performance computing cluster
  - Targeted at a corporate or academic IT person who is tasked with creating a scientific computing cluster for the first time
- This is NOT:
  - An exact blueprint for creating a cluster that meets your needs
  - An example of best practices throughout the industry
  - A complete solution at scale

Who am I, and why am I here?

- Stephen Lien Harrell
  - Scientific Applications Analyst in Research Computing at Purdue University
  - Specialization in imaging and configuration management for moderately large (500-2000 machine) clusters and HPC in undergraduate education.
  - Teach similar classes to undergraduates at Purdue University

- Amiya Maji
  - Scientific Applications Analyst in Research Computing at Purdue University
  - Ph.D., Purdue University
  - Expertise: Cloud Computing, Software Reliability

Goals and caveats

- Goals
  - Illuminate the technologies needed to build a scientific computing cluster
  - Show how the technologies fit together
  - Show an iterative and scalable configuration management model
- Caveats
  - We have little time for deep dives
  - I will be using simpler technologies in some places than industry standards.
  - Hardened security is out of our scope.
  - Our cluster will be built in EC2 which will be different in design and technologies than hardware clusters.

Prerequisites

- You must have a laptop or computing device that has an internet connection, ssh terminal and a modern browser.
- You must have an Amazon Web Services (AWS) account.

Task list for setting up our cluster

- Spin up machines in EC2
- Bootstrap Puppet
- Firewalls
- DNS
- Shared Storage
- Log aggregation
- Environment Modules
- Accounts
- Scheduler and Resource Manager
- Node Health Checks
- Run MPI pi calculator and HPL benchmark
- Ganglia
- Nagios

Timeline of sessions

LCI workshop
on high performance clustered computing

1	**Mon, 18 May, 3:30-5:30pm** EC2 Intro, Create Head Node, Puppet Repo, Firewall, DNS
2	**Tue, 19 May, 11:00am-12:00pm** Create Storage Server, NFS Mounts, Log Aggregation
3	**Tue, 19 May, 3:30pm-4:30pm** Bootstrap Compute Nodes with Environment Modules
4	**Wed, 20 May, 11:00am-12:00pm** Install Torque/Maui, Test Torque, NHC
5	**Wed, 20 May, 2:00pm-3:00pm** Compile and Run Applications (HPL and Pi calculator)
6	**Wed, 20 May, 4:30pm-5:30pm** Monitoring: Ganglia and Nagios

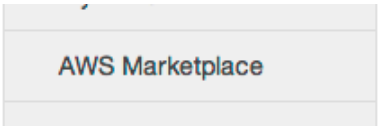
Files for the workshop

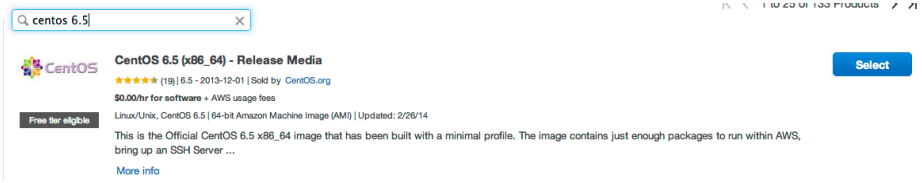
- I have created snippets of code and commands to help us move along quickly
- Each slide will be tagged with the snippet name we will be working with.
- The snippets will be available at
  - http://web.rcac.purdue.edu/~sharrell/buildacluster/
- These slides are intentionally incomplete without these snippets.

Session 1
AWS

Getting started with EC2

- Log into AWS
- Select EC2
- Click Launch Instance
- Click AWS Market Place
- Search for centos 6.5 and click Select

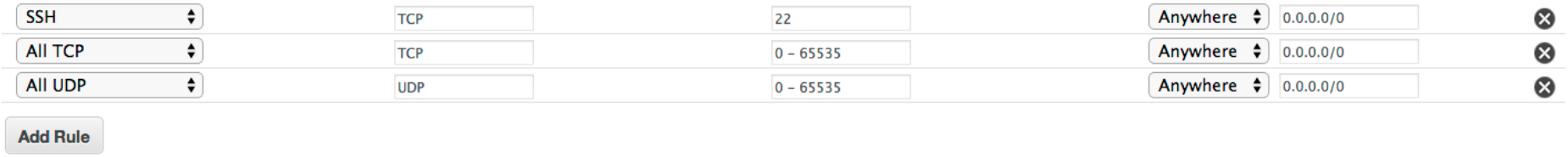




Getting started with EC2 – part 2

- Make sure micro is checked and click Review and Launch

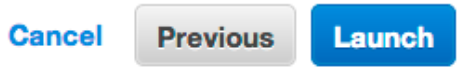
- Click Edit Security Groups
- Click Add Rule
- Select All TCP and change Source to Anywhere
- Do the same for UDP
- **Note**: We are opening the AWS firewalls because we'll be using host based firewalls



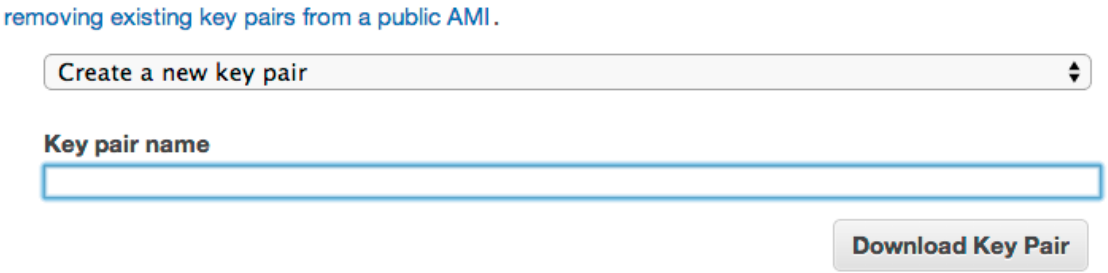
- Click Review and Launch

Getting started with EC2 – part 3

- Click Launch



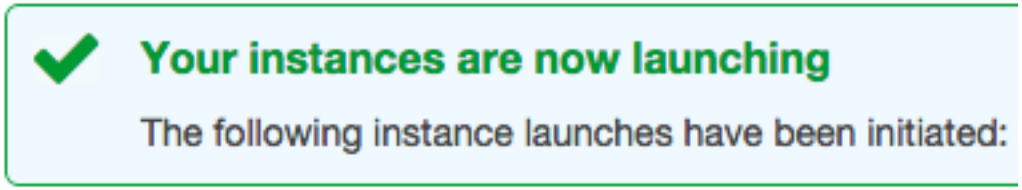
- Select a new pair give a name and click Download Key Pair



- Click Launch Instances

Getting started with EC2 – part 4

- You will see the message



- Click View Instances to see the status and find the IP address

- Select the instance to see the public IP address on the lower part of the screen

- **Note**: The public IP address may change across reboots, especially if the instance is down for any period of time.

- To login, ssh to the public IP address using the key you downloaded
  - ssh -i AWSkey.pem root@public-AWS-IP

Getting started with EC2 – part 5

- If your AWS key file or the directory containing it have permissions allowing others read access you will see the message

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@         WARNING: UNPROTECTED PRIVATE KEY FILE!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0640 for 'Downloads/slide01.pem' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
```

- If you receive the above message, change the permissions on the key file and/or directory holding the key.
    - chmod 600 myAWSKey.pem

Session 1
Setup Head Node

Puppet

- We will be using a subset of the puppet configuration management tool
- Puppet has many capabilities and language abstractions
- My main goals are readability and manageability after the class is over

SVN and Puppet primer

- svn up
  - Update an existing repository with the current commits
- svn di
  - Print the diff of your current tree vs the remote tree at your current commit
- svn ci
  - Check in changes
- puppet apply
  - This applies any changes in the puppet tree, this is followed by the path of site.pp.
  - You may want to make an alias for “puppet apply /etc/puppet/manifests/site.pp”

Bootstrapping puppet – part 1

- SSH into the node as root
  - ssh -i AWSkey.pem root@public-AWS-IP
- Install the puppet repository
  - yum –y install http://yum.puppetlabs.com/puppetlabs-release-el-6.noarch.rpm
- Install puppet, git, subversion, apache with mod ssl and vim
  - yum -y install puppet git mod_ssl vim subversion

Bootstrapping puppet – part 2

- Create subversion repository and initiliaze puppet tree
  - mkdir /var/svn
  - svnadmin create /var/svn/puppet
  - svn import /etc/puppet file:///var/svn/puppet -m "Initial puppet import.”
- Move original puppet directory out of the way and check out version controlled puppet
  - mv /etc/puppet /etc/puppet.orig
  - svn co file:///var/svn/puppet /etc/puppet
- Install puppet modules we will be using throughout the setup
  - for package in puppetlabs-apache puppetlabs-firewall spiette-selinux AlexCline-mounts torrancew-account saz-resolv_conf saz-rsyslog petems-swap_file; do puppet module install $package;done
  - git clone https://github.com/rlex/puppet-dnsmasq.git /etc/puppet/modules/dnsmasq
  - git clone https://github.com/haraldsk/puppet-module-nfs.git /etc/puppet/modules/nfs

Bootstrapping puppet – part 3

- Set editor then add and check in modules
  - export EDITOR=vim
  - svn add /etc/puppet/modules/*
  - svn ci /etc/puppet
- Create directory for puppet configs and add and check in
  - mkdir /etc/puppet/manifests
  - touch /etc/puppet/manifests/site.pp
  - touch /etc/puppet/hiera.yaml
  - svn add /etc/puppet/manifests
  - svn add /etc/puppet/hiera.yaml
  - svn ci /etc/puppet

Bootstrapping puppet – part 4

- Generate self-signed certificate for use with apache and subversion
  - mkdir /etc/httpd/ssl
  - openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/httpd/ssl/apache.key -out /etc/httpd/ssl/apache.crt
    - Common Name will be head.cluster

- Create http user for subversion remote access
  - htpasswd -c /etc/httpd/auth_user_file root
  - chown apache:apache /etc/httpd/auth_user_file

Puppet layout – part 1

- /etc/puppet/manifests/site.pp
  - This is where we will be doing the lion's share of our configuration of the cluster
  - We want to minimize the amount of software setup commands that we run on the command line
  - We want to maximize the documentation and scripting of what needs to be done on the machines
  - We will be applying updates with puppet apply
    - This avoids a lot of extra setup of puppet and lets us get into the things that matter for our cluster.

Puppet layout – part 2

- Head Node
  - Puppet repository
  - Scheduling server
- Storage Node
  - Shared file system server
- Compute Node
  - Scheduling client
  - User libraries

```
class base_cluster {   }
class head_node {   }
class storage_node {   }
class compute_node {   }
# head node
node 'head.cluster',  {
   include head_node
   include base_cluster
}
# storage node
node 'storage.cluster' {
   include storage_node
   include base_cluster
}
# compute nodes
node  'compute1.cluster', 'compute2.cluster' {
   include compute_node
   include base_cluster
}
```

Puppet subversion repository deployed by puppet – Part 1

- Boilerplate apache module instantiation

```
class { 'apache':
  default_confd_files => false,
  purge_configs => false,
}
class { 'apache::mod::dav_svn': }
```

- Define apache vhost

```
apache::vhost { 'head.cluster':
  port => 443,
  docroot => '/var/www/html/',
  ssl => true,
  ssl_cert => '/etc/httpd/ssl/apache.crt',
  ssl_key  => '/etc/httpd/ssl/apache.key',
  custom_fragment => '
   <Location /puppet >
    AuthType Basic
    AuthName "Puppet Cluster Repository"
    AuthUserFile "/etc/httpd/auth_user_file"
    Require valid-user
    DAV svn
    SVNPath /var/svn/puppet/
   </Location>'
}
```

Puppet subversion repository deployed by puppet – Part 2

- puppet apply
  - Will get error message with fqdn of host
  - Add short host as head node
  - # headnode
  - node ‘head.cluster’, 'ip-172-31-7-24' { … }
- puppet apply
- fix permissions
  - chcon -R -h -t httpd_sys_content_t /var/svn/puppet
  - chown -R apache:apache /var/svn/puppet

General system housekeeping

- Add puppet apply on boot
  - file_line { 'puppet-apply-on-boot':
    path => '/etc/rc.d/rc.local',
    ensure => present,
    line => '/usr/bin/puppet apply /etc/puppet/manifests/site.pp',
    }
- Add a swap file
  - swap_file::files { 'default':
    ensure => present,
    }
- Add Exec path globally for older module compatibility
  - Exec { path => ["/bin/", "/sbin/" , "/usr/bin/", "/usr/sbin/"] }
- Make SELinux not bother us that much
  - class { 'selinux':
    mode => 'permissive',
    }
- Add miscellaneous utilities to make this easier.
  - package { 'bind-utils':
    ensure => present,
    }
  - wget
  - lsof
  - mlocate
  - strace
  - telnet
  - netcat
  - screen
- **puppet apply**

Puppet firewall prep

- Make directory for new module in puppet
  - mkdir -p /etc/puppet/modules/my_fw/manifests

- **Flush IPTables** to prevent ssh-blocking race condition
  - iptables -F

Basic firewall – part 1

```
class my_fw {

   $ipv4_file = $operatingsystem ? {
      "debian"         => '/etc/iptables/rules.v4',
      /(RedHat|CentOS)/ => '/etc/sysconfig/iptables',
   }

   firewall { "001 accept all icmp requests":
      proto => 'icmp',
      action  => 'accept',
   }
 firewall { '002 INPUT allow loopback tcp':
      iniface => 'lo',
      chain   => 'INPUT',
      action    => 'accept',
      proto => 'tcp',
   }
   firewall { '002 INPUT allow loopback udp':
      iniface => 'lo',
      chain   => 'INPUT',
      action    => 'accept',
      proto => 'udp',
   }
firewall { '000 INPUT allow related and established':
      state => ['RELATED', 'ESTABLISHED'],
      action  => 'accept',
      proto => 'all',
   }
firewall { '100 allow ssh':
      state => ['NEW'],
      dport => '22',
      proto => 'tcp',
      action  => 'accept',
   }

   firewall { "998 deny all other requests":
      action   => 'reject',
      proto  => 'all',
      reject => 'icmp-host-prohibited',
   }

   firewall { "999 deny all other requests":
      chain  => 'FORWARD',
      action   => 'reject',
      proto  => 'all',
      reject => 'icmp-host-prohibited',
   }
}
```

This will all go into: /etc/puppet/modules/my_fw/manifests/init.pp

Basic firewall – part 2

- **Back to site.pp**
- We will set the IPs for the entire cluster here
  - $headnodeip=' 172.31.7.24'
  - $storagenodeip='127.0.0.2'
  - $computeoneip='127.0.0.3'
  - $computetwoip='127.0.0.4'
- Firewall Boilerplate
  - class base_cluster {
    resources { "firewall":
    purge => true
    }

    class { 'my_fw': }
    class { 'firewall': }

- Create firewall rule for each machine to allow all machines to communicate freely.
  - firewall { '003 INPUT allow head ip':
    chain => 'INPUT',
    action => 'accept',
    proto => 'all',
    source => "${headnodeip}/32",
    }

  … and repeat for the rest of the machines.
- Allow access to our web SVN tree from anywhere
  - firewall { '100 allow https access':
    state => ['NEW'],
    dport => 443,
    proto => tcp,
    action => accept,
    }
- **svn add modules/my_fw**
- **svn ci** and **puppet apply**

Local DNS setup

- Fix race condition
  - Class['dnsmasq'] -> Class['resolv_conf'] -> Exec['set-hostname-to-dns']
- DNSMasq module boilerplate
  - class { 'dnsmasq':
    interface => 'lo',
    …
    }
- Set outbound DNS server
  - dnsmasq::dnsserver { 'dns':
    ip => '8.8.8.8',
    }
- Hacky reverse name generation
  - $iparray_head = split($headnodeip, '[.]
  - $headnode_reverse = join(… ,'')
- Set forward and reverse for the head node
  - dnsmasq::address { "head.cluster":
    ip => $headnodeip,
    }
  - dnsmasq::ptr {$headnode_reverse:
    value => 'head.cluster',
    }
- Setup resolv.conf to point to dnsmasq
  - class { 'resolv_conf':
    nameservers => ['127.0.0.1'],
    searchpath => ['cluster'],
    }
- Hacky Hostname correction
  - "hostname $(dig +short -x `hostname -I` | sed 's/\.\+$//')",

}

- **svn ci** and **puppet apply**

Session 2
Setup Storage Server

Bootstrapping storage (and compute) node(s)

- Launch new instance from EC2
  - Make sure you use the same security group and keys as before
- Login to the node
- Install Puppet Repository
  - yum install -y http://yum.puppetlabs.com/puppetlabs-release-el-6.noarch.rpm
- Install puppet, vim and subversion
  - yum -y install puppet vim subversion
- Remove default puppet configs
  - rm -rf /etc/puppet
- checkout puppet svn to /etc/puppet
  - svn co https://LOCAL_HEADNODE_IP/puppet /etc/puppet/
- Flush IPTables
  - iptables -F
- Change ip address for the storage/ compute node
- Add the short name to the storage/ compute node definition
- **svn ci** and **puppet apply**
- **reboot**

NFS Server configuration

- Create /apps/ directory for shared software
  - file { "/apps":
    ensure => "directory",
    }
- Create NFS exports for /home and /apps
  - include nfs::server
  - nfs::server::export{ '/home/':
    ensure => 'mounted',
    clients => '172.31.0.0/16(rw,insecure,async,no_root_squash) localhost(rw)',
    }
  - nfs::server::export{ '/apps/':
    ensure => 'mounted',
    clients => '172.31.0.0/16(rw,insecure,async,no_root_squash) localhost(rw)',
    require => File['/apps']
    }
- **svn ci** and **puppet apply**

NFS mounts on head and compute nodes

- Mount /home
  - mounts { 'storage server home':
    ensure => present,
    source => "${storagenodeip}:/home",
    dest => '/home',
    type => 'nfs',
    opts => 'nofail,defaults,vers=3,rw,noatime',
  }
- Mount /apps
  - mounts { 'storage server apps':
    ensure => present,
    source => "${storagenodeip}:/apps",
    dest => '/apps',
    type => 'nfs',
    opts => 'nofail,defaults,vers=3,rw,noatime',
    require => File['/apps'],
  }
- Add this for the head node and cluster node classes and puppet apply

Log aggregation

- Add rsyslog server to the head node
  - Create directory to hold all of the logs
    - file {'/var/log/multi/':
      ensure => 'directory',
      before => Class['rsyslog::server'],
      }
  - Add rsyslog module
    - class { 'rsyslog::server':
      server_dir => '/var/log/multi/',
      }
- Add rsyslog client to send logs to server
  - class { 'rsyslog::client':
    remote_type => 'tcp',
    server => 'head.cluster',
    }

Session 3
Setup Compute Nodes

Bootstrapping compute nodes

- Launch 2 more EC2 machines
- Change the ip addresses for the compute nodes on the head node
  - $computeoneeip='computenode ip 1 here'
  - $computetwoeip='computenode ip 2 here'
  - **svn ci and puppet apply on the head node**
  - **svn up and puppet apply on the storage node**
- Go back to bootstrapping storage slide and bootstrap the two nodes

Accounts

- In an academic or corporate environment you will most likely be using ldap or something similar. This method is an easy way around having to setup an ldap.

```
account {
  'login_name_here':
    home_dir => '/home/login_name_here',
    groups   => [ 'wheel', 'users' ],
    comment   => 'Full Name',
    uid => 500,
 }
```

- This will allow us to have a UID consistent user everywhere without setting up a full accounting system.
- **puppet apply**

Environment Modules and OpenMPI

- Environment modules can provide pluggable software.

- Install basic Packages
  - package { 'environment-modules':
    ensure => present,
    }

    package { 'gcc-c++':
    ensure => present,
    }

    package { 'gcc-gfortran':
    ensure => present,
    }

- OpenMPI Software
  - cd /apps/
  - wget openmpi-1.7.5.tar.gz
  - tar xfvz openmpi-1.7.5.tar.gz

- OpenMPI Module
  - Create the directory for the module files
    - file { "/usr/share/Modules/modulefiles/openmpi":
      ensure => "directory"
      }

OpenMPI module

- Create the .version file. This file contains the default version for the module.
  - "#%Module1.0
    set ModulesVersion \"1.7.5\""
- Create the actual module file
  - #%Module1.0
    module-whatis \"invoke openmpi-1.7.5\"
    set version 1.7.5
    set app openmpi
    set modroot /apps/openmpi-1.7.5/
    prepend-path PATH \$modroot/bin
    prepend-path LD_LIBRARY_PATH \$modroot/lib
    setenv MPI_HOME \$modroot
    setenv CC mpicc
    setenv CXX mpiCC
    setenv F77 mpif77
    setenv FC mpif90\n"

OpenBLAS module

- OpenBLAS Software
  - cd /apps/
  - wget openblas-0.2.10.tar.gz
  - tar xfvz openblas-0.2.10.tar.gz
- OpenBLAS Module
  - Create the directory for the module files
    - file { "/usr/share/Modules/modulefiles/openblas":
      ensure => "directory"
      }
- Create the .version file. This file contains the default version for the module.
  - "#%Module1.0
    set ModulesVersion \"0.2.10\""
- Create the actual module file
  - "#%Module1.0
    module-whatis \"invoke openblas-0.2.10\"
    set version 0.2.10
    set app openblas
    set modroot /apps/openblas-0.2.10/
    prepend-path PATH \$modroot/bin
    prepend-path LD_LIBRARY_PATH \$modroot/lib"

Session 4
Setup Scheduler

Setting up torque

LCI workshop
on high performance clustered computing

- Setup torque libs and files across the whole cluster
  - package { 'libxml2':
    ensure => present,
    }
    package { 'torque':
    ensure => 'installed',
    source => 'torque-4.1.7-1.adaptive.el6.x86_64.rpm',
    provider => 'rpm',
    }
    file { '/var/spool/torque/server_name':
    content => "head.cluster\n",
    }
- Install torque server and scheduler packages on head node
  - package { 'maui':
    ensure => 'installed',
    source => 'maui-3.3.1-x86_64-fpmbuild.rpm',
    provider => 'rpm',
    require => Package['torque']
    }
  - package { 'torque-server':
    ensure => 'installed',
    source => 'torque-server-4.1.7-1.adaptive.el6.x86_64.rpm',
    provider => 'rpm',
    require => Package['torque']
    }
- Setup services and config files for torque on the head node
  - service { "pbs_server":
    #ensure => "running",
    enable => "false",
    }
  - service {"maui.d":
    ensure => "running",
    enable => "false",
    }
  - file { '/var/spool/torque/server_priv/nodes':
    content => "compute1.cluster np=1\n
    compute2.cluster np=1\n",
    }
- **Puppet apply**
- **Stop Torque**: /etc/init.d/pbs_server stop
- **Run the Torque Setup**: /usr/share/doc/torque-server-4.1.7/torque.setup
- Allow pbsnodes to work on the nodes
  - qmgr -c 'set server managers = root@*.cluster'
- Change pbs_server and pbs_sched stanza to uncomment ensure running

Setting up torque – part 2

- Setup torque on the compute nodes
  - package { 'torque-client':
    ensure => 'installed',
    source => 'torque-client-4.1.7-1.adaptive.el6.x86_64.rpm',
    provider => 'rpm',
    require => Package['torque']
    }
    service { "pbs_mom":
    ensure => "running",
    enable => "true",
    require => Package["torque-client"],
    }
    file { '/var/spool/torque/mom_priv/config':
    content => "\$pbsserver head.cluster
    \$usecp *:/home /home\n",
    require => Package['torque-client'],
    notify => Service["pbs_mom"]
    }
- **svn ci** on head node and **svn up** on compute nodes, followed by **puppet apply**

Testing torque

- Make sure that our compute nodes are free
  - pbsnodes -a
- Start an interactive job
  - su login_user
  - qsub –I
- Start an interactive job with two nodes
  - qsub -I -l nodes=2
- Getting Debug Information
  - Show all jobs
    - qstat –a
  - Get information about specific job
    - qstat {jobid}
    - tracejob {jobid}
  - Show downed nodes
    - pbsnodes –ln
  - Important logs to check
    - /var/spool/torque/server_logs/*
    - /var/spool/torque/mom_logs/*

Node Health Checks

- Install the NHC package
  - package { 'warewulf-nhc':
    ensure => 'installed',
    source => 'http://warewulf.lbl.gov/downloads/repo/rhel6/warewulf-nhc-1.3-1.el6.noarch.rpm',
    provider => 'rpm',
    }
- Run the health check at jobstart and offline the node if problems
  - \$node_check_script /usr/sbin/nhc
  - \$node_check_interval jobstart
  - \$down_on_error 0\n”
- Add these lines to the existing mom_config file. Watch for the “, file contents terminator.
- Set the checks
  - Check if / is mounted
    - /./ || check_fs_mount_rw /
    - check_fs_mount_rw /apps
  - Check if SSH is running
    - * || check_ps_daemon sshd root\n
  - Check if there is the correct amount of physical memory
    - * || check_hw_physmem 1024 1073741824\n
  - Check if there is any free
    - * || check_hw_physmem_free 1\n”
- Are there any other checks that could be important for job starts?

Testing Node Health Checks

- Check to make sure both nodes are up and test a 2 node job
  - qsub -I -l nodes=2
- Unmount /apps on compute1.cluster
  - umount /apps
- Wait for the node too offline itself (should take 45 seconds or less)
  - pbsnodes –a

Session 5
Run Applications

Compiling and running MPI pi calculator
module and qsub commands

- Change user on head node to login user
  - su login_user
- Start an interactive job
  - qsub -I -l nodes=2
- Generate ssh keys and authorize them
  - ssh-keygen
  - cp ~/.ssh/id_rsa.pub ~/.ssh/authorized_keys
- Get the MPI pi calculator
  - wget pi.c
- List available modues
  - module avail
- Load the mpi module
  - module load openmpi
- Compile the program
  - mpicc pi.c -o pi
- Test pi single threaded
  - ./pi
- Run mpiexec to execute pi across two nodes
  - mpiexec -prefix /apps/openmpi-1.7.5/ -machinefile $PBS_NODEFILE /home/login_user/pi

Compiling HPL

- Remaining in the interactive job
- Download HPL
  - wget hpl-2.1.tar.gz
  - tar xfvz hpl-2.1.tar.gz
  - mv hpl-2.1 hpl
- Load openmpi module
  - module load openmpi
- Grab a working makefile
  - cd hpl
  - cp setup/ Make.Linux_PII_CBLAS_gm ./
- Edit the makefile and set the correct LAdir and LAlib paths
  - LAdir = /apps/openblas-0.2.10/ lib/
  - LAlib = $(LAdir)/libopenblas.a
- Compile HPL
  - make arch=Linux_PII_CBLAS_gm

Running HPL

- Modify HPL.dat
  - cd bin/Linux_PII_CBLAS_gm
  - Edit HPL.dat
  - Modify the Ps and Qs
    - 1 1 1 Ps
    - 1 1 1 Qs
- HPL tuning
  - http://www.netlib.org/benchmark/hpl/tuning.html

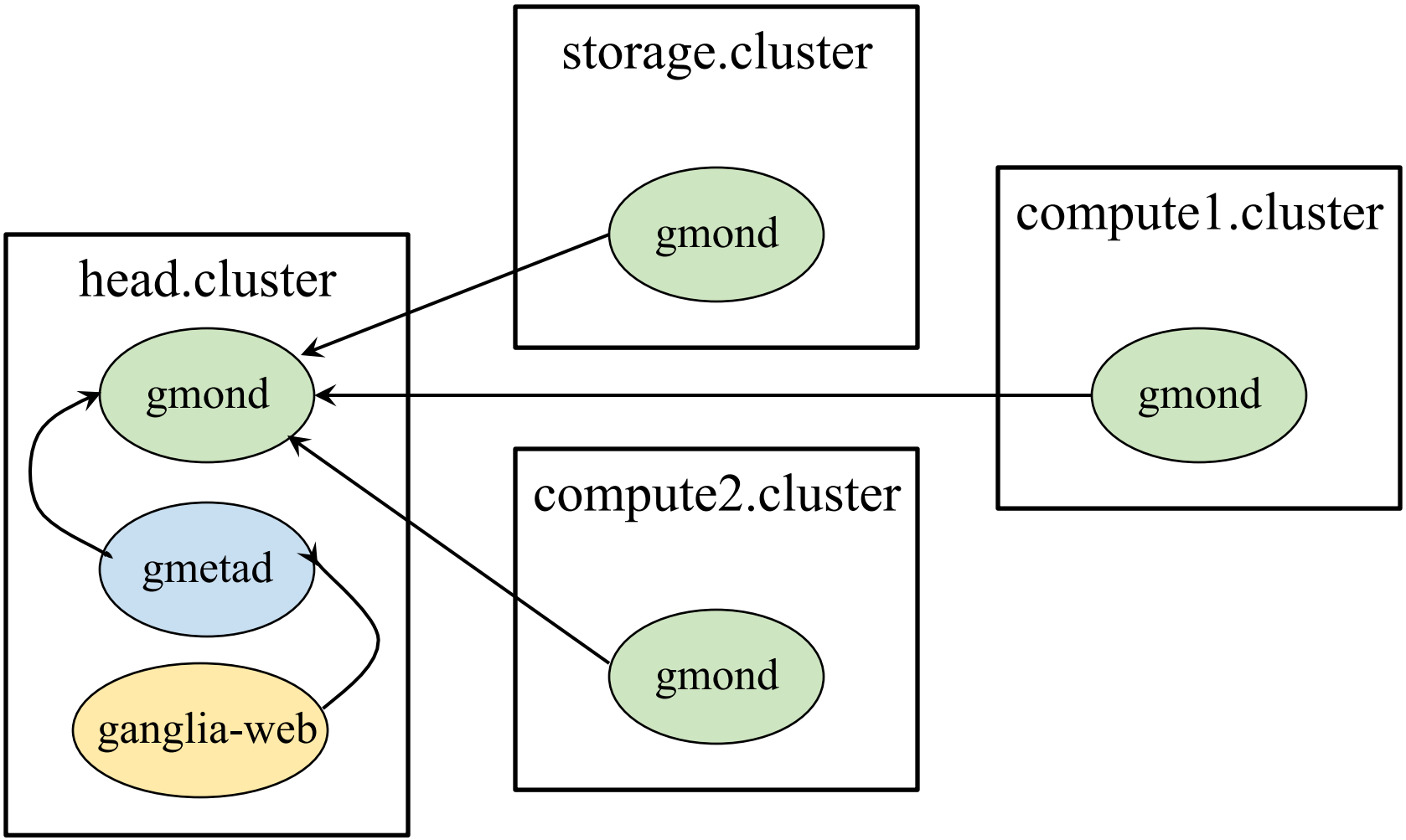
- Execute hpl
  - mpiexec -prefix /apps/openmpi-1.7.5/ -np 2 -machinefile $PBS_NODEFILE /home/sharrell/hpl/bin/Linux_PII_CBLAS_gm/xhpl
- Marvel at the speed of our cluster in comparison to the top 500.
  - http://www.top500.org/lists/2014/06/

Session 6
Monitoring: Ganglia and Nagios

Ganglia monitoring framework

- Allows monitoring a wide variety of utilization metrics from the hosts
- Consists of three components
  - Ganglia Monitoring Daemon (**gmond**)
    - Run on all nodes
    - Collects various metrics
  - Ganglia Metadata Daemon (**gmetad**)
    - Runs on head node
    - Keeps a RRD of metrics
  - Ganglia Web Interface
    - Runs on head node
    - Queries gmetad and plots metrics
- We shall use EPEL binaries (RPM) for Ganglia/Nagios
  - Simplifies some of the configuration steps
- We shall use predefined config files for gmond and gmetad.

Ganglia - components



Ganglia - preparation

- We make puppet changes in head_node
- Create a custom directory for Ganglia configs in Puppet
  - mkdir -p /etc/puppet/modules/my_configs/files
  - Call this $CONFDIR
- Create custom configs for gmond and gmetad
  - wget http://web.rcac.purdue.edu/~sharrell/buildacluster/023A-gmond-server-conf
  - Download the other two files (023B, 023C) from website
  - cp “023A-gmond-server-conf” “$CONFDIR/gmond-server.conf”
  - cp “023B-gmond-client-conf” “$CONDIR/gmond-client.conf”
  - cp “023C-gmetad-server-conf” “$CONFDIR/gmetad-server.conf”
- **svn add** modules/my_configs
- **svn ci**

Ganglia – base_cluster

- Edit in class base_cluster { … }
- Setup EPEL repositories

```
package { 'epel-release.noarch':
   ensure => 'present',
 }
```

- Install Ganglia base package and Gmond

```
package { 'ganglia':
   ensure => 'present',
   require => Package['epel-release.noarch']
 }
 package { 'ganglia-gmond':
   ensure => 'present',
   require => Package['epel-release.noarch', 'ganglia']
 }
```

- Create service gmond

```
service { "gmond":
  ensure  => "running",
  enable  => "true",
  require => [Package['ganglia-gmond'], File['/etc/ganglia/gmond.conf']],
}
```

Ganglia – head_node

- Edit in class head_node { … }
- Install gmetad and ganglia-web

```
package { 'ganglia-gmetad':
  ensure => 'present',
  require => Package['epel-release.noarch',
  'ganglia']
 }
 package { 'ganglia-web':
  ensure => 'present',
  require => Package['epel-release.noarch',
  'ganglia']
 }
```

- Edit Apache configs so that we can see Ganglia UI

```
# default ganglia.conf is too restrictive. let's
  change it.
 file { '/etc/httpd/conf.d/ganglia.conf':
  ensure  => file,
  content => '
            Alias /ganglia /usr/share/ganglia
            <Location /ganglia>
              Allow from all
            </Location>',
  require => Package['ganglia-web'],
  notify => Service['httpd'],
 }
```

- Now to configure gmond and gmetad
  - What's in the config files?
- Create gmond config

```
file { '/etc/ganglia/gmond.conf':
  ensure => file,
  owner  => 'root',
  group  => 'root',
  mode   => '0644',
  source => 'puppet:///modules/my_configs/gmond-
server.conf',
  require => Package['ganglia-gmond'],
  notify => Service['gmond'],
 }
```

- Similar for gmetad config
- Create service gmetad

```
service { "gmetad":
  ensure  => "running",
  enable  => "true",
  require => [Package["ganglia-gmetad"],
Service["gmond"]],
 }
```

Ganglia – storage and compute

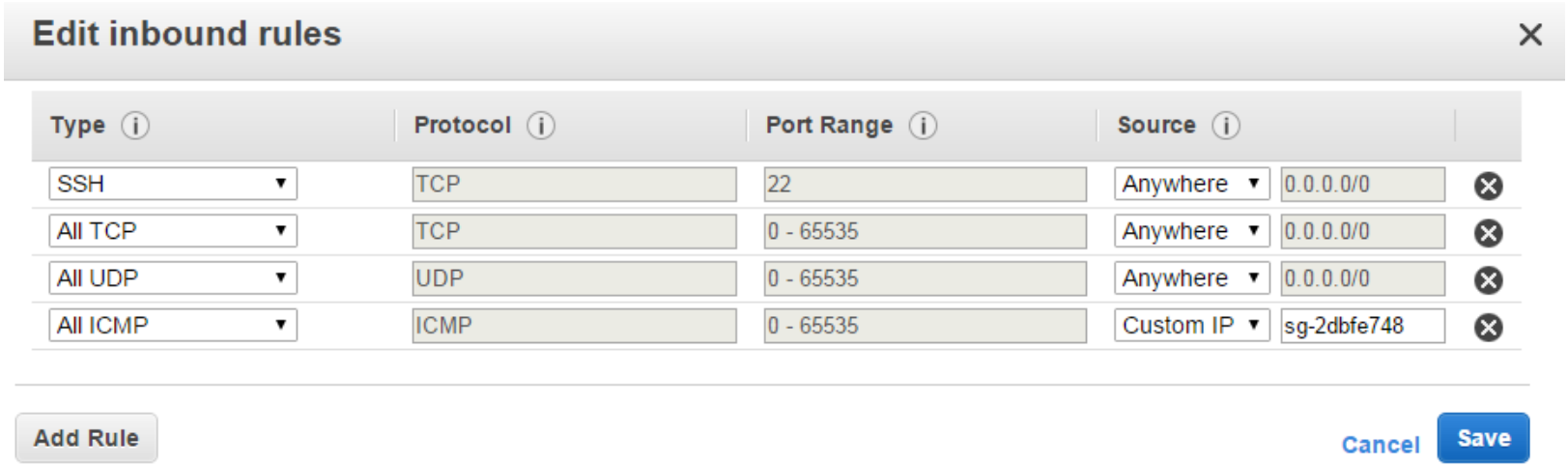
- Configure gmond on storage and compute nodes
- Edit classes storage_node { … } and compute_node { … }

```
### BEGIN GANGLIA
  file { '/etc/ganglia/gmond.conf':
    ensure => file,
    owner  => 'root',
    group  => 'root',
    mode   => '0644',
    source => 'puppet:///modules/my_configs/gmond-client.conf',
    require => Package['ganglia-gmond'],
    notify => Service['gmond'],
  }
### END
}
```

- To see Ganglia in action visit:
  - https://headnode-public-ip/ganglia
- **svn ci** and **puppet apply** on head_node
- **svn up** and **puppet apply** on all other nodes

Nagios

- Open source infrastructure monitoring software
- Allows you to monitor state of host, network, services and create alerts
- We shall set up Nagios on head node and create two simple checks
  - ping check
  - ssh check
- Enable all ICMP in AWS Security policy so that we can ping hosts



Nagios head_node

- Edit in class head_node { … }
- Install Nagios base packages and plugins

```
package { ['nagios-common', 'nagios',
  'nagios-plugins']:
  ensure => 'present',
  require => Package['epel-release.noarch'],
 }
 package { ['nagios-plugins-ssh', 'nagios-
  plugins-ping']:
  ensure => 'present',
  require => Package['nagios-plugins'],
 }
```

- Remove the default localhost config that was created

```
exec { 'remove_locahost_conf':
  command => 'mv /etc/nagios/objects/
  localhost.cfg /etc/nagios/objects/
  localhost.cfg.orig; touch /etc/nagios/
  objects/localhost.cfg',
  require => Package['nagios'],
 }
```

- Now define our own set of hosts that we shall monitor

```
file { '/etc/nagios/conf.d/hosts.cfg':
ensure => file,
content => "
        define host{
                use     linux-server
                host_name head.cluster
                alias   head
                address ${headnodeip}
        }
    …. . # Add other nodes here
    …..
```

- Define a hostgroup for setting up checks easily (in hosts.cfg)

```
define hostgroup{
    hostgroup_name  linux-servers
    alias Linux Servers
    members head.cluster, storage.cluster,
   compute1.cluster, …..
 }",
require => Package['nagios'],
notify => Service['nagios'],
} #end file_hosts_cfg
```

Nagios – service configs

- Now create checks that we want: ping and ssh

```
file { '/etc/nagios/conf.d/services.cfg':
  ensure => file,
  content => "
          define service{
                use             local-service
                hostgroup_name  linux-servers
                service_description PING
                check_command   check_ping!
  100.0,20%!500.0,60%
          }
          define service{
                use             local-service
                hostgroup_name  linux-servers
                service_description SSH
                check_command   check_ssh
                notifications_enabled 0
          }",
  require => [Package['nagios'], File['/etc/nagios/
  conf.d/hosts.cfg']],
  notify => Service['nagios'],
 }
```

- Create the nagios service

```
service { "nagios":
    ensure  => "running",
    enable  => "true",
    require => [Package["nagios"], File['/etc/
 nagios/conf.d/services.cfg']],
  }
```

- Check status by browsing the Nagios UI.
  - https://headnode-public-IP/nagios
- Default username/password are
  - nagiosadmin/nagiosadmin
- Click “Hosts” in left panel and view current node status
- For advanced Nagios monitoring use NRPE plugins

We have learned!

- Creating and Using EC2 Instances
- Basic Puppet Use
- Firewall Configuration
- Local DNS Setup
- Shared Storage
- Install and Configure Torque and Maui
- Environment Modules
- Node Health Checks
- Log Aggregation
- Ganglia
- Nagios
- Run MPI pi calculator and HPL benchmark

Questions? Comments?

- Contact:
  - Stephen Lien Harrell – SLH@purdue.edu
  - Amiya Maji – amaji@purdue.edu

- Don't forget to shutdown your EC2 instances!!!