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

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 isn’t:
  • 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
  • Senior System Administrator in Research Computing at Purdue University
  • Previously a System Administrator at Google
  • Specialization in imaging and configuration management for moderately large (500-2000 machine) clusters.
  • Teach similar classes to undergraduates at Purdue University

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 account.

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/](http://web.rcac.purdue.edu/~sharrell/buildacluster/)
  - These slides are intentionally incomplete without these files.
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

• 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

Puppet

• We will be using a subset of the puppet configuration management tool

• Puppet has many capabilities and language abstractions

• My main concern is readability and manageability after the class is over
Bootstrapping puppet – part 1

• SSH into node
  • ssh -i AWSKey.pem root@aws-ip

• Install the puppet repository
  • yum --install http://yum.puppetlabs.com/puppetlabs-release-el-6.noarch.rpm

• Install puppet, git, subversion, apache with mod_ssd and vim
  • yum -y install puppet git mod_ssl vim subversion

Bootstrapping puppet – part 2

• Install the puppet modules we will be using today
  • for package in puppetlabs-apache puppetlabs-firewall spjette-sellinux AlexCline-mounts torrancew-account saz-resolv_conf
    saz-ryssig petsens-swap_file, saz-ryssig
  • 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

• Setup SSL
  • 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

• Setup user account for our upcoming SVN tree
  • htpasswd -c /etc/httpd/auth_user_file root
  • chown apache:apache /etc/httpd/auth_user_file

• Create Subversion repository
  • mkdir /var/svn/
  • svnadmin create /var/svn/puppet

• Create directory for puppet configurations
  • mkdir /etc/puppet/maroltests
Puppet layout – part 1

- **Head Node**
  - Puppet repository
  - Scheduling server

- **Storage Node**
  - Shared file system server

- **Compute Node**
  - Scheduling client
  - User libraries

```ruby
class base_cluster {
}  # head node
include base_cluster

class head_node {
}  # head node
include head_node
include base_cluster

class storage_node {
}  # storage node
include storage_node
include base_cluster

class compute_node {
}  # compute nodes
include compute_node
include base_cluster
```

Puppet layout – part 2

- **/etc/puppet/manifests/site.pp**

  - This is where we will be doing the lionshare 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 subversion repository deployed by puppet – Part 1

- Boilerplate apache module instantiation

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

- define apache vhost

```puppet
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
Puppet subversion repository deployed by puppet – Part 3

- The old puppet switcheroo
  - `svn co https://localhost/puppet/ /root/puppet/`
  - `cp -R /etc/puppet/ /root/puppet/`
  - `svn add /root/puppet/*`
  - `export EDITOR=vim`
  - `svn ci`

  - `rm -rf /etc/puppet`
  - `svn co https://localhost/puppet/ /etc/`

- Now we have a version controlled puppet install

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 main 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”
General System housekeeping

- Lets add a swap file
  - include swap_file

- 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
  - milocate
  - strace
  - telnet
  - netcat
  - screen

- puppet apply
- reboot

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

- 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 => "$\{\text{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 ci and puppet apply
Local DNS Setup

- Fix race condition
  - Class['dnsmasq'] -&gt; Class['resolv_conf'] -&gt; Exec['set-hostname-to-dns']

- DNSMasq module boilerplate
  - class {'dnsmasq':
    interface =&gt; 'lo',
  } ...

- Set outbound DNS server
  - dnsmasq::dnsserver {'dns':
    ip =&gt; '8.8.8.8',
  }

- Set forward and reverse for the head node
  - dnsmasq::address {'head.cluster':
    ip =&gt; 'sheadnodeip',
  }
  - dnmasq::ptr {'24.7.31.172.in-addr.arpa.':
    value =&gt; 'head.cluster',
  }

- Setup resolv.conf to point to dnsmasq
  - class {'resolv_conf':
    nameservers =&gt; ['127.0.0.1'],
    searchpath =&gt; ['cluster'],
  }

- Hacky Hostname correction
  - "hostname $(dig +short -x `hostname -I` | sed 's/\.

- svn ci and puppet apply

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 and reverse DNS for the storage node

- Add the short name to the storage 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 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'],
    }
- Need to do this for the head node as well as the compute nodes and puppet apply

011-puppet-nfs-server

012-storage-firewall-and-mounts
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
  • 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
Setting up torque – part 1

- Setup torque libs 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,
- Install torque server and scheduler packages on head node
  - package {'torque-scheduler'}:
    ensure => 'installed',
    source => 'torque-scheduler-4.1.7-1.adaptive.el6.x86_64.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 => "true",
    require => Package["torque-server"],
  } service { "pbs_sched"
    ensure => "running",
    enable => "true",
    require => Package["torque-scheduler"],
  }
  - file { '/var/spool/torque/server_priv/nodes':
    content => "$pbsserver head
    \$usecp */home /*",
    require => Package["torque-client"],
    notify => Service["pbs_server"],
  }
- Puppet apply
- Run /usr/share/doc/torque-server-4.1.7/torque.setup
- 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
    \$usecp */home /*n",
    require => Package["torque-client"],
    notify => Service["pbs_mom"]
  }
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 -l n`

- Important logs to check
  - `/var/spool/torque/server_log`
  - `/var/spool/torque/mom_log`

Node Health Checks

- Install the NHC package
  - `package { 'warewulf-nhc':
    ensure => 'installed',
    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 1

- Set the checks
  - Check if / is mounted
    - `\./ | | check_fs_mount_rw /`
  - Check if SSH is running
    - `* | | check_ps_daemon sshd root`
  - Check if there is the correct amount of physical memory
    - `* | | check_hw_physmem 1024 1073741824`
  - Check if there is any free
    - `* | | check_hw_physmem_free 1

- Are there any other checks that could be important for job starts?
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/open mpi':
        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"

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 -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 modules
  - 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`
    - `Ps`
    - `Qs`
- HPL tuning
  - [http://www.netlib.org/benchmark/hpl/tuning.html](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/](http://www.top500.org/lists/2014/06/)
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',
    }

Other diagnostic tools at scale

- Monitoring
  - OpenNMS
  - Nagios

- Gathering information
  - pdsh

- Useful charts and graphs
  - Ganglia
  - Logstash/Kibana
Scaling

- Finding Contention
  - Usage
  - Saturation
  - Errors

- Tools to help
  - Ganglia
  - Observium
  - Cacti
  - Logstash/Kibana

Scaling: Networking and Storage

- 1Gbps/10Gbps/40Gbps Ethernet
  - Top-of-rack and Row-Aggregation
  - Add links
  - Up link speeds
  - VPC/Trill

- 40/56Gbps Infiniband
  - More links in a Fat Tree topology
  - Hypercube
  - Torus
  - “Islands” Topology

- Local Storage - Smallest client number
  - RAID
  - LVM
  - Ext4, XFS, BtrFS

- Simple Network Storage
  - NFS
  - Samba

- Distributed Storage
  - GlusterFS
  - Ceph
  - AFS

- Parallel Storage — Largest client number
  - Lustre
  - GPFS
Scaling: Scheduling and Resource Management

- Job Throughput
  - PBS Sched
  - Maui
  - Moab
- Cluster Size
  - Torque
  - HT Condor
  - SLURM

Scaling: Cluster Management

- IP management solutions
  - Consistent network spaces and physical locations
  - Working DNS
  - DHCP

- Imaging Solutions
  - Kickstart
  - xCAT

- More robust config management system
  - Expanding puppet to use facter and hiera
  - xCAT
Software Management and User Considerations

• Adding repos of software you need

• Creating your own repo

• Keeping tight version control so there are fewer moving parts (no auto upgrading)

• Usage models
  • Condo
  • Allocation
  • Open access/Fair share

• Software Support
  • How far should I go?

Questions? Comments?

• Contact:
  • Stephen Lien Harrell – SLH@purdue.edu

• Don’t forget to shutdown your EC2 instances.