Deploying an open source HPC environment on an IBM @server BladeCenter JS20 using SUSE LINUX Enterprise Server 8 and CSM

September 7, 2004

Written By:
Kevin P. McCombs
IBM Corporation
mccombsk@us.ibm.com

Contributors:
Vallard Benincosa
IBM Corporation
vallard@us.ibm.com

Erik Salander
IBM Corporation
salander@us.ibm.com

Guanshan Tong (Linpack)
IBM Corporation
gtong@us.ibm.com

Introduction

Cluster Systems Management (CSM) provides a distributed system management solution that allows a system administrator to set up and maintain a cluster of nodes that run the Linux® operating system. CSM simplifies cluster administration tasks by providing management from a single point-of-control (a Management Server). CSM has the ability to ease the setup of High Performance Computing (HPC) clusters. This document gives instructions on how to install, configure, and test several common open source high performance computing applications.

xCAT (Extreme Cluster Administration Toolkit) is a tool kit that can be used for the deployment and administration of Linux clusters. xCAT provides additional features such as hardware setup, HPC stack configuration, and xCAT integration. Many of the scripts in xCAT are early versions of tools that will eventually be merged into CSM in future releases. For the purposes of this paper, we will use the HPC setup scripts in xCAT.

Torque is a portable batch scheduler (PBS)/resource manager package that includes a client, a server, and a scheduler. It was spun off of Open PBS to provide a free PBS version.

Maui is a batch scheduler that works with Torque or other batch schedulers. It is the scheduler part of the operation that runs jobs and maintains the queue with the PBS server.

MPICH and LAM are MPI libraries used for parallel programming in a cluster environment. In general LAM produces better performance on Ethernet connections, while mpich has given better performance on Myrinet™ switch.

If you are going to install and maintain HPC components then it is highly recommended that you subscribe to the products mailing lists as things change very quickly and this document can be outdated at anytime and probably already has been by the time you read this. The best way to keep up is to be on the mailing list and see what is happening in the community. See a full collection of mailing lists at the end of this paper.

Method

To briefly describe our method we will first install CSM and xCAT core on the Management Server. Next we will define nodes to CSM but not install them. Next we will use xCAT scripts and commands to set up the HPC components on the Management Server and then build node install images. Finally, using CSM, we will deploy the compute nodes.

CSM Overview

It is worth noting two CSM components, SMS and CFM, since they will play a large role in our HPC stack deployment. For more information on these please refer to the CSM Administration Guide.

SMS - Software Management System

SMS is used to manage RPMs across your cluster. We will use it to install RPMs on top of the base Linux operating system installed by CSM. The packages must reside in the following directory to be applied…

/csminstall/*InstallOSName/InstallDistributionName/InstallDistributionVersion/ InstallPkgArchitecture/install/.*

For example, in the case of SUSE LINUX Enterprise Server 8 (SLES 8)…
/csminstall/Linux/SLES/8.1/ppc64/install/.

SMS will manage RPMs unattended via cron overnight, during a CSM installnode or manually via the smsupdatenode command.

CFM – Configuration File Manager

CFM keeps files consistent across the compute nodes. /cfmroot serves as a repository for configuration files you want to keep common. The files must reside in the /cfmroot directory to be distributed. CFM decides where the files belong based on the path/name (less /cfmroot). For example, if you want to distribute /etc/profile, either copy or link it into /cfmroot…

cp /etc/profile /cfmroot/etc/profile

CFM will distribute files unattended via cron overnight, during a CSM installnode or manually via the cfmupdatenode command.

HPC Component Install Methods

There are multiple ways to install most of the HPC components. We are going to discuss up to 3 ways to implement each of these components across your cluster

1. Build components on each compute node using CFM.
2. Using NFS, share key directories across the cluster that contain executables and configuration files.
3. Using SMS or CFM, distribute RPMs and install.

It is important to consider which approach you wish to take before you begin. The following table shows the components and the methods we will cover in this paper.

	Torque	Maui	MPICH	LAM	VAC	XLF	ESSL	Linpack
Build	X	X	X	X				X
NFS	X		X	X				
RPM			X	X	X	X	X	

Install CSM and xCAT Core

CSM is available here:

http://techsupport.services.ibm.com/server/cluster/fixes/csmfixhome.html

Install CSM by following the instructions in the CSM Planning and Installation Guide.

There is also a whitepaper that specifically deals with installing SLES 8 on IBM @server BladeCenter™ JS20 nodes using CSM, "**Installing SUSE SLES 8 SP3 on a BladeCenter JS20**". This is good prerequisite reading to this document.

Next install xCAT core (this is actually a gzip of a tarball). xCAT is available here:

http://www.alphaworks.ibm.com/tech/xcat

(assuming the gzip file is in /tmp)
gunzip xcat-dist-core-1.2.0-pre6.tar.gz
cd /opt/csm
tar xvf /tmp/xcat-dist-core-1.2.0-pre6.tar

Install HPC Components

The following is a list of prerequisites you should obtain before continuing

Cross Compilers:
cross-ppc64-linux-2.4.19-139
cross-ppc64-binutils-2.13.90.0.4-84
cross-ppc64-gcc-3.2.2-50
cross-ppc64-glibc-2.2.5-140

Development Environment
glibc-devel
cpp
gcc-
libstdc++
libstdc++-devel
gcc-c++

Note: that if you are using gcc-3* or greater you need to install the compat libraries as well.

Also note that if you plan to use the IBM compilers, it would be best to install them before installing the HPC components.

Install Torque

Prerequisites
For SLES 8 you will need xdevel and tcl-devel in addition to the standard c/c++ compilers.

Note: Put RPMs in the csminstall directory (i.e. /csminstall/Linux/SLES/8.1/ppc64/install/)

You can download Torque from
http://www.supercluster.org/downloads/torque/

Copy it to the /tmp directory and untar it there. Next, copy the conf.cmd from /opt/csm/xcat/csm/hpc/torque into this directory.
cd /tmp
tar zxvf torque-1.0.1p6.tar.gz
cd torque-1.0.1p
cp /opt/csm/xcat/csm/hpc/torque/conf.cmd .

Check out the configuration options of the conf.cmd. This command will configure, build, and install the server on the machine. There are a few things to note here:

- For simplicity, modify the conf.cmd script to remove the $ARCH and $ARCHBIN directory levels.
- The server home is in /var/spool/pbs. The default PBS installation uses /usr/spool, however, since logs fill up and can adversely machine functionality, it is best to keep these all in /var. In addition, all commands are put in /usr/local/pbs so as to distinguish pbs commands from other commands from maui and mpich.
- Enable syslog so that we can see errors from pbs in /var/log/messages
- Build pbs with scp. While some people prefer rcp, there have been known scaling problems with rsh. In general, if it is not a significant amount of nodes and security is not a concern, rsh will probably give you faster performance. (not sure if that is true or not, but people think it is)

First, some prep work:
cp /usr/lib/rpm/config.sub /tmp/torque-1.0.1p6/buildutils/.
cp /usr/lib/rpm/config.guess /tmp/torque-1.0.1p6/buildutils/.

Now you can install torque:
./conf.cmd

All output is stored in a log. To see it while it builds:
tail -f /tmp/pbsbuild.log

If everything was successful then you should have files in /usr/local/pbs/bin and /usr/local/pbs/sbin.

Let's also build the man pages for torque

cd /tmp/torque-1.0.1p6/doc

Edit Makefile and change the variable "prefix" from /usr/local/pbs to /usr/local

make install
man qsub (see if it works)

Install Maui

Download Maui from:
http://66.237.84.51/cgi-bin/login.cgi

You may have to set up an account and log in to get the tarball.

Copy it into the /tmp directory, untar it, and build it.

```
# cd /tmp
# tar zxvf maui-3.2.6p8.tar.gz
# cd maui-3.2.6
# ./configure --with-pbs=/usr/local/pbs
# make
# make install
```

Look for error messages. If everything worked out, then you should have binary files in /usr/local/maui/bin.

Install MPICH

The most commonly used implementations of MPI are mpich and mpich-gm. For the purpose of this document we will discuss mpich. If you are using the GM protocol over a Myrinet network, you will want to use mpich-gm. We will discuss two installs of mpich. First using make, then using RPMs. It is up to you which you use.

Download mpich tarball to /tmp from
http://www-unix.mcs.anl.gov/mpi/mpich/
(version 1.2.5.2 is about 12.4 MB)

The download may be named mpich.tar.gz. Move this to mpich-<version>.tar.gz so mpimaker can use it.
cd /tmp
cp mpich.tar.gz mpich-1.2.5.2.tar.gz

Copy mpimaker from /opt/csm/xcat/build/mpi to /tmp as well and run it:
cp /opt/csm/xcat/build/mpi/mpimaker .
./mpimaker 1.2.5.2 smp gnu ssh
mpimaker: 1.2.5.2 smp gnu ssh build start
mpimaker: 1.2.5.2 smp gnu ssh make
mpimaker: 1.2.5.2 smp gnu ssh build successful

MPICH installed in /usr/local/mpich/1.2.5.2/ip/smp/gnu/ssh

Please check config.cmd make.log install.log configure.log in
/usr/local/mpich/1.2.5.2/ip/smp/gnu/ssh
for errors. config.cmd was the command used to build MPICH

This code makes the following assumptions:
- /usr/local is the install directory

Watch the build. If there are errors check the logs
cd mpich-1.2.5.2
tail -f configure.log

Using MPICH RPMS

You can download the MPICH rpms from
http://ppclinux.ncsa.uiuc.edu/

If you install mpich or mpich-gm by rpm it is recommended that you also install mpi-wrappers rpm. This package basically simplifies the many compiler and run time options available for mpi by reading environment variables and setting paths accordingly. The location of these compilers is not required knowledge for the end user. It is also recommended that you install mpi-wrappers on compute nodes.

Download the rpms into /tmp and install…
rpm -i mpich-1.2.5-2.ppc64.rpm
rpm -i --nodeps /tmp/mpi-wrappers-1.0.0-1.ppc64.rpm

Install LAM

To be able to pass compute messages across your cluster you will either need mpich or lam-mpi, you shouldn't need both unless you want to test for performance. In general for clusters where the interconnect is Ethernet you should use lam/mpi. We will discuss two installs of lam. First using make, then using rpms. It is up to you which you use.

Download lam to the /tmp directory
http://www.lam-mpi.org/7.0/download.php

Next copy lammaker to the /tmp file as well and build it. lammaker was created by Matt Bonsack and is more flexible then the other programs. Read the source to see what it can do. It also makes several assumptions and does some configuring:

- builds with gnu or pgi compilers
- requires a Fortran compiler
- builds in /usr/local/…
- puts lam in the path in /usr/local/{LAM_HOME}/etc/conf.[csh|sh]

```
# cd /tmp
# cp /opt/csm/xcat/build/lam/lammaker .
# ./lammaker lam-7.0.5.tar.gz gnu ssh
./lammaker: lam-7.0.5.tar.gz gnu ssh build start
See lam-7.0.5/configure.log.
./lammaker: lam-7.0.5.tar.gz gnu ssh make
See lam-7.0.5/make.log.
./lammaker: lam-7.0.5.tar.gz gnu ssh build successful
```

Watch the build:

```
# cd lam-7.0.5
# tail -f configure.log
# tail -f make.log
# tail –f install.log
```

Using LAM RPMS

If you install LAM by rpm it is recommended that you also install the mpi-wrapper scripts. This package simplifies the many compiler options available for mpi by reading environment variables and setting paths accordingly. The location of these compilers is not required knowledge for the end user. It is also recommended that you install mpi-wrappers on compute nodes.

Note: Put RPMs in the csminstall directory (i.e. /csminstall/Linux/SLES/8.1/ppc64/install/)

You can get the LAM rpm from
http://ppclinux.ncsa.uiuc.edu/

Download the rpm into /tmp and install…
rpm -i --nodeps /tmp/lam*.rpm
rpm -i --nodeps /tmp/mpi-wrappers-1.0.0-1.ppc64.rpm

Server and Compute Node Setup

Now that binaries are built we must configure the HPC stack on the management server as well as the compute nodes that are to be installed. Remember, the nodes have not yet been installed so the node setup tasks will be done in CSM directories on the management server.

Torque On MS

- Add these directories to $PATH for all cluster users. Make sure that however you accomplish this gets copied to /cfmroot (ie /cfmroot/etc/profile).
  export PATH=$PATH:/usr/local/pbs/bin:/usr/local/pbs/sbin
- Run default torque setup as root
  # /tmp/torque-1.0.1p6/torque.setup root
- Create file /var/spool/pbs/server_priv/nodes. Must contain one node name per line.
- Copy pbs_server.suse into /etc/init.d/ for SLES and rcpbs_server needs to be linked. Then turn on and start the server service
  # cp /opt/csm/xcat/csm/hpc/lib/pbs_server.suse /etc/init.d/pbs_server
  # ln –s /etc/init.d/pbs_server /usr/sbin/rcpbs_server
  # chkconfig –s pbs_server on
  # rcpbs_server start

- If you intend on sharing Torque directories instead of copying executables to the compute nodes, make sure you share them.
  - Add /usr/local to /etc/exports
    /usr/local *.ibm.com(rw,async,no_root_squash)
  - Make it live
    # exportfs –a

Torque on Compute Nodes

- If you choose not to NFS share directories, you will have to copy the following binaries to /cfmroot
  # cd /cfmroot
  # mkdir -p usr/local/pbs/bin
  # mkdir -p usr/local/pbs/sbin
  # cp -var /usr/local/pbs/bin/* ./usr/local/pbs/bin/
  # cp -var /usr/local/pbs/sbin/* ./usr/local/pbs/sbin/

- If you intend on sharing directories rather than copying binaries to each compute node, /usr/local from the MS has to be NFS mounted on each node. We are going to create the entry in a file and then append the file to /etc/fstab
    - Add the following line to /cfmroot/tmp/fstab.local
      tomato.ibm.com:/usr/local /usr/local nfs
      rsize=8192,wsize=8192,timeo=14,intr 0 0
    - Create a script /tmp/fstab.local.post (mode 755) to make it effective
      cat /tmp/fstab.local >> /etc/fstab
      mount –a –t nfs
    - Create a script (mode 755) in the directory
      /csminstall/csm/scripts/installpostreboot/ that adds and starts the NFS service after the CSM install with the following command.
      chkconfig –s nfs on

  Copy the following directories to /cfmroot.
  # cp -var /var/spool/pbs/* /cfmroot/var/spool/pbs/.

  Remove scheduler and server directories
  # cd /cfmroot/var/spool/pbs
  # rm –r sched_priv
  # rm –r sched_logs
  # rm –r server_priv
  # rm –r server_logs

  Note: Depending on the CSM version CFM may not create empty directories on the compute nodes. These directories are required for the moms to run. A simple way to get around this is to put an empty file in these directories as such
  # > /cfmroot/var/spool/pbs/spool/foo
  # > /cfmroot/var/spool/pbs/aux/foo
  # > /cfmroot/var/spool/pbs/mom_logs/foo
  # > /cfmroot/var/spool/pbs/mom_priv/jobs/foo

- Create the mom configuration /cfmroot/var/spool/pbs/mom_priv/config (mode 644) with the following minimum config parameters
  $clienthost *management-server-name*
  $logevent 0x1ff

- Create prologue and epilogue scripts on nodes
  # cp /opt/csm/xcat/csm/hpc/lib/*logue /cfmroot/var/spool/pbs/mom_priv/.

- Copy pbs_mom.suse from xCAT into /cfmroot/etc/init.d/
  # mkdir –p /cfmroot/etc/init.d/
  # cp /opt/csm/xcat/csm/hpc/lib/pbs_mom.suse /cfmroot/etc/init.d/pbs_mom

- rcpbs_mom needs to be linked. Create a CFM .post script /cfmroot/etc/init.d/pbs_mom.post (mode 755) that executes this command
  ln –s /etc/init.d/pbs_mom /usr/sbin/rcpbs_mom
- Create a script (mode 755) in the directory /csminstall/csm/scripts/installpostreboot/ that adds and starts the mom service after the CSM install with the following commands
  chkconfig –s pbs_mom on
  rcpbs_mom start
- Copy prerequisite rpms to /csminstall/Linux/*rel/ver/arch*/
  tcl-devel-8.4-47.ppc.rpm
  xdevel-4.2.0-143.ppc.rpm

Maui On MS

- Add maui to environment and make permanent
  export PATH=$PATH:/usr/local/maui/bin
- maui.suse needs to be copied from xCAT into /etc/init.d/ for SLES and rcmaui needs to be linked
  # cp /opt/csm/xcat/build/maui/maui.suse /etc/init.d/maui
  # ln –s /etc/init.d/maui /usr/sbin/rcmaui
  # chkconfig –s maui on
- maui needs to be started on the MS
  # rcmaui start

MPICH on MS

- The location of mpich needs to be included in in your path. If you look in the mpich directories, you can see there are many choices, make sure you are pointing at the path that fits your environment. Please note that if you install the rpm version, the settings will be different. For example, if mpich-1.2.5.2 was used, add the following to your environment (ie. .bashrc, /etc/profile).

```
export MPICH="/usr/local/mpich/1.2.5.2/ip/smp/gnu/ssh"
export MPICH_PATH="${MPICH}/bin"
export MPICH_LIB="${MPICH}/lib"
export PATH="${MPICH_PATH}:${PATH}"
export LD_LIBRARY_PATH="${MPICH_LIB}:${LD_LIBRARY_PATH}"
```

- If you intend on NFS sharing MPICH directories instead of installing MPICH on the compute nodes, make sure you share them.
    - Add /usr/local to /etc/exports
      /usr/local *.ibm.com(rw,async,no_root_squash)
    - Make it live
      # exportfs –a

- If installing the RPMs, we will use the wrapper scripts to customize our MPICH environment. Instead of the environment variable settings above, make the following settings permanent…

```
export MP_RSH=ssh
export MP_EUILIB=ip
export OBJECT_MODE=64
export P4_GLOBMEMSIZE=512000000
```

MPICH on Compute Nodes

- Copy (or link) environment updates to /cfmroot
  # cp /etc/profile /cfmroot/etc/profile

There are basically 3 ways to install mpich on the compute nodes, via NFS sharing of libraries, running the make procedure (mpimaker) on each compute node or by rpm install on each node.

If you intend on sharing directories rather than installing MPICH on each compute node, /usr/local from the MS has to be NFS mounted on each node. We are going to create the entry in a file and then append the file onto /etc/fstab (you may have already done these steps if you are using NFS for Torque on the compute nodes).

- Add the following line to /tmp/fstab.local
  tomato.ibm.com:/usr/local /usr/local nfs
  rsize=8192,wsize=8192,timeo=14,intr 0 0
- Create a script /tmp/fstab.local.post (mode 755) to make it effective
  cat /tmp/fstab.local >> /etc/fstab
  mount –a –t nfs
- Create a script (mode 755) in the directory /csminstall/csm/scripts/installpostreboot/ that adds and starts the mom service after the CSM install with the following command.
  chkconfig –s nfs

If installing via make on each compute node individually

- Copy tarball and install script to /cfmroot
  # cp /tmp/mpimaker /cfmroot/tmp/mpimaker
  # cp /tmp/mpich-1.2.5..2.tar.gz /cfmroot/tmp/.
- Then build a script named /cfmroot/tmp/mpimaker.post (mode 755) that does the following
  cd /tmp
  ./mpimaker 1.2.5.2 smp gnu ssh >>/tmp/mpimaker.out
- As mentioned earlier you can also check config.cmd make.log install.log configure.log in/usr/local/mpich/1.2.5.2/ip/smp/gnu/ssh for errors after the node is installed.

If using an rpm, copy it to /csminstall, ie,

cp /tmp/mpich-1.2.5-2.ppc64.rpm /csminstall/Linux/SLES/8.1/ppc64/install/.

- Install the wrapper scripts as well. This has to be done outside of SMS because –nodeps is required. So we'll copy it to tmp and create a .post script to install it.
  # cp /tmp/mpi-wrappers-1.0.0-1.ppc64.rpm /cfmroot/tmp/.

- Contents of /cfmroot/tmp/mpi-wrappers-1.0.0-1.ppc64.rpm.post (mode 755)…
  # rpm –i --nodeps /tmp/mpi-wrappers-1.0.0-1.ppc64.rpm

LAM/MPI on MS

- The location of lam needs to be included in in your path. Please note that if you installed the rpm version, these settings will be different. Add the following to your environment (note: put these in .bashrc)…

```
export LAM="/usr/local/lam-7.0.5/gnu/ssh"
export LAM_PATH="${LAM}/bin"
export LAM_LIB="${LAM}/lib"
export PATH="${LAM_PATH}:${PATH}"
export LD_LIBRARY_PATH="${LAM_LIB}:${LD_LIBRARY_PATH}"
```

- If you intend on sharing LAM directories instead of installing LAM on the compute nodes, make sure you share them.
    - Add /usr/local to /etc/exports
      /usr/local *.**ibm.com**(rw,async,no_root_squash)
    - Make it live
      # exportfs –a

- If installing the RPMs, we will use the wrapper scripts to customize our LAM environment. Instead of the environment variable settings above, make the following settings permanent…

```
export MP_RSH=ssh
export MP_EUILIB=ip
export OBJECT_MODE=64
export P4_GLOBMEMSIZE=512000000
```

LAM/MPI on Compute Nodes

- Copy (or link) environment updates to /cfmroot
- Copy prerequisite rpms to /csminstall/Linux/*rel/ver/arch*/install. This may or may not be required in your case, but is worth noting here.
    - libaio-0.3.15-19.ppc.rpm

As with mpich there are basically 3 ways to install lam on the compute nodes, NFS sharing of libraries, running the make procedure (lammaker) on each compute node or by installing the RPMs on each node.

If you intend on sharing directories rather than installing LAM on each compute node, /usr/local from the MS has to be NFS mounted on each node. We are going to create the

entry in a file and then append the file onto /etc/fstab (you may have already done these steps if you are using NFS for Torque on the compute nodes).

- Add the following line to /tmp/fstab.local
  tomato.ibm.com:/usr/local /usr/local nfs
  rsize=8192,wsize=8192,timeo=14,intr 0 0
- Create a script /tmp/fstab.local.post (mode 755) to make it effective
  cat /tmp/fstab.local >> /etc/fstab
  mount –a –t nfs
- Create a script (mode 755) in the directory
  /csminstall/csm/scripts/installpostreboot/ that adds and starts the NFS service after the CSM install with the following command.
  chkconfig –s nfs

Installing via make on each compute node individually

- Copy lam tarball and install script to /cfmroot
  # cp /tmp/lammaker /cfmroot/tmp/lammaker
  # cp /tmp/lam-7.0.5.tar.gz /cfmroot/tmp/.
- Then build a script named /cfmroot/tmp/lammaker.post (mode 755) that does the following
  cd /tmp
  lammaker lam-7.0.5.tar.gz gnu ssh >>/tmp/lammaker.out
- As mentioned earlier you can check configure.log and make.log for errors after the node is installed.

Installing RPMs on each compute node individually, copy it to /cfmroot. This has to be done outside of SMS because --nodeps is required. So we'll copy it to tmp and create a .post script to install it. Install the wrapper scripts as well.

Copy RPMS
cp /tmp/lam-7.0.4-1.ppc64.rpm /cfmroot/tmp/.
cp /tmp/mpi-wrappers-1.0.0-1.ppc64.rpm /cfmroot/tmp/.

Contents of /cfmroot/tmp/mpi-wrappers-1.0.0-1.ppc64.rpm.post (mode 755)…

rpm –i --nodeps /tmp/mpi-wrappers-1.0.0-1.ppc64.rpm

Contents of /cfmroot/tmp/lam-7.0.4-1.ppc64.rpm.post (mode 755)…

rpm –i --nodeps /tmp/lam-7.0.4-1.ppc64.rpm

Node Install

Pre CSM Install Tasks

It is a good idea to copy some additional files into /cfmroot, depending on your configuration (these can be links as well)

/etc/profile
/etc/profile.local
/etc/passwd
/etc/group
/etc/shadow
/etc/hosts
/home/*

You may want to expand the number of default packages that CSM installs to your cluster. More than likely there will be packages required by one of more of your HPC components that is not in the default install. The safest approach is to include the same base Linux packages on the nodes that are on the Management server. You can add packages to the CSM autoyast template. To do so, modify the following file...

/opt/csm/install/yastcfg.SLES8.1-ppc64.xml

For the HPC stack, I would recommend adding the following lines

```
<package>xntp</package>
<package>rsync</package>
<package>gnome-libs</package>
<package>compat</package>
<package>ypbind</package>
<package>ypserv</package>
<package>yptools</package>
<package>glibc</package>
<package>gcc</package>
<package>gcc-c++</package>
<package>java2</package>
<package>java2-jre</package>
<package>ppc64-utils</package>
```

CSM Install

At this point you should be able to install the nodes via CSM and the HPC stack should be configured automatically on them. See CSM documentation on how to deal with node groups.

csmsetupyast –xn node1,node2,…
installnode –n node1,node2,…

Post CSM Install Tasks

There are a few things that have to be done before the node(s) are ready to go.

- SMS doesn't always install all the rpms in /csminstall/Linux/SLES/8.1/ppc64/install, specifically ones that aren't ppc64. If you find this to be the case, we can use a short script to do this.

  ```
  for i in `ls /csminstall/Linux/SLES/8.1/ppc64/install/*.rpm`
  do
      smsupdatenode -v -i $i -a

  done
  ```

- Add cluster users and copy home directories. Some choose to NFS share the /home directory from the Management Server instead of copying. CSM will not copy files owned by users that are not defined. So before we can copy we must add a user by running the xCAT utility addclstrusr.

  # /opt/csm/xcat/csm/bin/addclstrusr –N AllNodes ***ibm***
  (where AllNodes is the CSM groupname for all nodes and ibm is the user to be added)

  In case addclstrusr does not work, I've included the manual user add instructions to add user ibm. Run these commands from the management server.

  ```
  # useradd ibm
  # mkdir –p /home/ibm/.ssh
  # chmod 700 /home/ibm/.ssh
  # ssh-keygen -t rsa -q -N "" -f /home/ibm/.ssh/id_rsa
  # cd /home/ibm/.ssh
  # cp id_rsa.pub authorized_keys
  ```

```
# vi config  (add the following)
        ForwardX11 yes
        StrictHostKeyChecking no
        CheckHostIP no
# ssh-keyscan –trsa –f /var/spool/pbs/server_priv/nodes >> authorized_keys
# chmod go-rwx authorized_keys
# mkdir –p /cfmroot/home/ibm/.ssh
# chown –R ibm:users /home/ibm
# cp –var /home/ibm/.ssh/* /cfmroot/home/ibm/.ssh
```

- If you aren't sharing /home and there are files in cfmroot owner by cluster user ibm, we must copy them to the nodes (i.e. /home/ibm)
  ```
  # cfmupdatenode -a
  ```

Verifying the HPC Cluster environment

We should run some basic tests to make sure our HPC environment is functioning correctly. Note: these are only examples, your path names may vary.

- As a cluster user, make sure you can ssh to other nodes without a password.

Verify PBS

- Run a simple PBS job
  # su – ibm
  ibm@tomato:~> qsub -l nodes=1,walltime=10:00 –I
  qsub: job 14.tomato.ibm.com ready

  Begin PBS Prologue Wed Jul 7 14:57:10 PDT 2004
  Job ID: 14.tomato.ibm.com
  Username: ibm
  Group: users
  Nodes: pepper
  End PBS Prologue Wed Jul 7 14:57:10 PDT 2004
  ---
  ibm@pepper:~> date
  Wed Jul 7 15:00:33 PDT 2004
  ibm@pepper:~> exit
  logout

  qsub: job 14.tomato.ibm.com completed

- Verify environment is correct for MPICH or LAM. As Cluster user
  # which mpicc
  /opt/osshpc/mpich-1.2.5/64/ch_shmem/bin/mpicc

Verify MPICH

- Run a simple MPICH job via PBS. First build the program that computes pi using mpicc then distribute it to the cluster and run. Depending on which MPICH you have built, you may have to search for cpi.c.
  # su – ibm
  # cp /opt/osshpc/mpich-1.2.5/64/ch_shmem/examples/cpi.c .
  # mpicc –o cpi cpi.c

```
# dcp -a  /home/ibm/cpi /home/ibm/cpi  (copy executable to nodes unless /home is shared)
# qsub -l nodes=4,walltime=10:00:00 –I
qsub: job 19.tomato.ibm.com ready

----------------------------------------
Begin PBS Prologue Thu Jul  8 09:26:18 PDT 2004
Job ID:         19.tomato.ibm.com
Username:       ibm
Group:          users
Nodes:          pepper carrot cucumber onion
End PBS Prologue Thu Jul  8 09:26:18 PDT 2004
----------------------------------------

ibm@pepper:~> mpirun -machinefile $PBS_NODEFILE -np 1 cpi
Process 0 on pepper.ibm.com
pi is approximately 3.1416009869231254, Error is 0.0000083333333323
wall clock time = 0.000095
Process 3 of 4 on carrot
Process 1 of 4 on cucumber
Process 2 of 4 on onion
ibm@pepper:~> logout

qsub: job 19.tomato.ibm.com completed
```

Verify LAM

- Run a simple MPI/LAM job via PBS. First build the program that computes pi using mpicc then distribute it to the cluster and run.

```
# su – ibm
# cp /tmp/lam-7.0.5/examples/pi/cpi.c .
# which mpicc           (verify you are using the right mpicc)
/usr/local/lam-7.0.5/gnu/ssh/bin/mpicc
# mpicc –o cpi cpi.c
# dcp -a  /home/ibm/cpi /home/ibm/cpi  (copy executable to nodes unless /home is shared)
#  qsub -l nodes=1:ppn=1,walltime=10:00:00 –I
qsub: job 27.tomato.ibm.com ready

----------------------------------------
Begin PBS Prologue Thu Jul 15 14:26:59 PDT 2004
```

```
Job ID:         27.tomato.ibm.com
Username:       ibm
Group:          users
Nodes:          pepper
End PBS Prologue Thu Jul 15 14:26:59 PDT 2004
----------------------------------------
ibm@pepper:~> which lamboot
/usr/local/lam-7.0.5/gnu/ssh/bin/lamboot
ibm@pepper:~> lamboot -v $PBS_NODEFILE
LAM 7.0.5/MPI 2 C++/ROMIO - Indiana University

n-1<7601> ssi:boot:base:linear: booting n0 (pepper)
n-1<7601> ssi:boot:base:linear: finished

ibm@pepper:~> mpirun C cpi
Process 0 on pepper
pi is approximately 3.1416009869231254, Error is 0.0000083333333323
wall clock time = 0.000098
ibm@pepper:~> lamclean
ibm@pepper:~> logout

qsub: job 31.tomato.ibm.com completed
```

Installing IBM Compilers

If you do not already have the VisualAge® C++ and XL Fortran compilers you can find out more at:

http://www-306.ibm.com/software/awdtools/vacpp/
http://www-306.ibm.com/software/awdtools/fortran/xlfortran/

Note: The IBM compilers are not open source.

Prerequisite RPMs

There are prerequisite packages and they are as follows:

(for SLES 8)
cpp
libelf
gcc
libstdc++
gcc-c++
libstdc++-devel
glibc-devel

Cross Compilers. These are required. Check the version numbers since these may not be the same that came with your SUSE SLES 8 media.
cross-ppc64-gdb-5.2-115
cross-ppc64-linux-2.4.19-139
cross-ppc64-binutils-2.13.90.0.4-84
cross-ppc64-libs_and_headers-8.1-157
cross-ppc64-gcc-3.2.2-50
cross-ppc64-glibc-2.2.5-140

Make sure you copy these RPMs to /csminstall/…/install

XL Fortran

Assuming XL Fortran tar file (c4983NA.tar) exists in /tmp
cd /tmp
mkdir xlf
cd xlf
tar xvf ../c4983NA.tar
cd RPMS
rpm –i xls*
rpm –i xlf*

Now configure it…
cd /opt/ibmcmp/xlf/8.1/bin
./new_install

Note: To use the compilers, you have to link the executables in /opt/ibmcmp/xlf/8.1/bin bin to the /usr/bin directory OR add the executable directories to the $PATH statement.

VisualAge C++

cd /tmp
mkdir vac
cd vac
tar xvf ../C498FML.tar
cd rpms
rpm –i xls* (if you didn't install them with XL Fortran)
rpm –i vac*

Now configure it…
cd /opt/ibmcmp/vac/6.0/bin
./new_install

Note: To use the compilers, you can link the executables you need in /opt/ibmcmp/vac/6.0/bin and /opt/ibmcmp/vacpp/6.0/bin to the /usr/bin directory OR add the executable directories to the $PATH statement.

Deploying IBM Compilers

It is assumed that we will want to deploy the run-time environment for the IBM compilers. We will do this using SMS. Make sure prereqs are met via autoyast or /csminstall/Linux/SLES/8.1/ppc64/install.

cp /tmp/vac/rpms/*rte*rpm /csminstall/Linux/SLES/8.1/ppc64/install/.
cp /tmp/xlf/RPMS/*rte*rpm / csminstall/Linux/SLES/8.1/ppc64/install/.
cfmupdatenode -a

ESSL (Engineering and Scientific Subroutine Library)

The Engineering and Scientific Subroutine Library (ESSL) is a set of high-performance mathematical subroutines. ESSL is provided as two run-time libraries and can be used with Fortran, C, and C++ programs.

If you do not already have ESSL and would like to learn more

http://techsupport.services.ibm.com/server/cluster/fixes/esslfixhome.html

Note: ESSL is not open source

Install ESSL on MS from CD

```
# mount /media/cdrom
# cd /media/cdrom
# cd SLES
# rpm –i essl.license-4.1.1-0.ppc64.rpm
# /opt/ibmmath/essl/4.1/bin/install_essl -y -d .
```

To verify this installed correctly, check the directories

```
# l /usr/lib/*essl*
lrwxrwxrwx    1 root     root          21 Jul 15 07:46 /usr/lib/libessl.so -> /usr/lib/libessl.so.1
lrwxrwxrwx    1 root     root          14 Jul 15 07:46 /usr/lib/libessl.so.1 -> libessl.so.1.1
-rw-r--r--    1 bin      bin     11055701 Aug  1  2003 /usr/lib/libessl.so.1.1
lrwxrwxrwx    1 root     root          24 Jul 15 07:46 /usr/lib/libesslsmp.so -> /usr/lib/libesslsmp.so.1
lrwxrwxrwx    1 root     root          17 Jul 15 07:46 /usr/lib/libesslsmp.so.1 -> libesslsmp.so.1.1
-rw-r--r--    1 bin      bin     12588368 Aug  1  2003 /usr/lib/libesslsmp.so.1.1
# l /usr/lib64/*essl*
lrwxrwxrwx    1 root     root          23 Jul 15 07:46 /usr/lib64/libessl.so -> /usr/lib64/libessl.so.1
lrwxrwxrwx    1 root     root          14 Jul 15 07:46 /usr/lib64/libessl.so.1 -> libessl.so.1.1
-rw-r--r--    1 bin      bin     11837908 Aug  1  2003 /usr/lib64/libessl.so.1.1
lrwxrwxrwx    1 root     root          26 Jul 15 07:46 /usr/lib64/libesslsmp.so ->
/usr/lib64/libesslsmp.so.1
lrwxrwxrwx    1 root     root          17 Jul 15 07:46 /usr/lib64/libesslsmp.so.1 -> libesslsmp.so.1.1
-rw-r--r--    1 bin      bin     13727868 Aug  1  2003 /usr/lib64/libesslsmp.so.1.1
```

Deploy ESSL to Compute Nodes

Copy the RPMs to /cfmroot, create a post install script then deploy using CFM.

cp /media/cdrom/SLES/*rpm /cfmroot/tmp/.

Create file /cfmroot/tmp/essl.license-4.1.1-0.ppc64.rpm.post (755 perms) that runs the following commands

cd /tmp
rpm –i essl.license-4.1.1-0.ppc64.rpm
/opt/ibmmath/essl/4.1/bin/install_essl -y -d .

Install with CFM
cfmupdatenode –a

Linpack (Hpl)

For the purpose of this document, Linpack was built using the MPICH and MPICH-WRAPPER scripts RPMs.

Download Linpack (filename: hpl.tgz) from:
http://www.netlib.org/benchmark/hpl/

Linpack Prerequisites

Before attempting to install Linpack make sure you have the following rpms installed in addition to the HPC components.
Cross Compilers
IBM VisualAge C++ compiler
IBM XL Fortran compiler
ESSL
Make sure IBM compilers are in $PATH. Specifically, make sure xlc_r and xlf_r are in $PATH.
Use MPICH and MPICH Wrapper Scripts

Install Linpack

Set environment variables for mpi-wrapper-scripts before you run make. You will want these environment variable setting permanent for your run-time cluster users.

```
# export MP_RSH=ssh
# export MP_EUILIB=ip
# export OBJECT_MODE=64
# export P4_GLOBMEMSIZE=512000000
```

Make sure hpl.tgz is in your home directory. The make scripts look for it there. To make it look somewhere else, you will have to modify the TOPdir variable in Make.PWRPC_FBLAS.

```
# cd /root
# tar zxvf hpl.tgz
# cd hpl
# cp setup/Make.PWRPC_FBLAS .
```

Now edit the file Make.PWRPC_FBLAS: =
84
MPdir =
MPinc =
MPlib =

97
LAlib = -lesslsmp -lxlsmp

169
CC = mpicc -cc=xlc_r -q64

171,174
CCFLAGS = $(HPL_DEFS) -O5 -qtune=pwr4 -qarch=pwr4 -qmaxmem=-1 -qsmp=noauto
FC = mpif90 -f90=xlf_r -q64
FCFLAGS = $(HPL_DEFS) -O5 -qtune=pwr4 -qarch=pwr4 -qmaxmem=-1 -qsmp=noauto
LINKER = mpif90 -f90=xlf_r -q64 -qsmp=noauto
LINKFLAGS =

Build Linpack
make arch=PWRPC_FBLAS

Deploy Linpack

Make sure you make the required environment variable settings permanent for cluster users.

Copy the Linpack executable into /cfmroot and deploy…

cp /root/hpl/bin/PWRPC_FBLAS/xhpl /cfmroot/root/hpl/PWRPC_FBLAS/.
cfmupdatenode -a

Test Linpack

Once built, the Linpack executable (xhpl) should sit in ~/hpl/bin/PWRPC_FBLAS. Let's try a simple test with the following HPL.dat values…

```
HPLinpack benchmark input file
Innovative Computing Laboratory, University of Tennessee
HPL.out      output file name (if any)
6            device out (6=stdout,7=stderr,file)
1            # of problems sizes (N)
4000          Ns
1            # of NBs
400           NBs
0            PMAP process mapping (0=Row-,1=Column-major)
1            # of process grids (P x Q)
1            Ps
2            Qs
16.0          threshold
1            # of panel fact
1            PFACTs (0=left, 1=Crout, 2=Right)
1            # of recursive stopping criterium
4            NBMINs (>= 1)
1            # of panels in recursion
2            NDIVs
1            # of recursive panel fact.
2             RFACTs (0=left, 1=Crout, 2=Right)
1            # of broadcast
2            BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
1            # of lookahead depth
1            DEPTHs (>=0)
2            SWAP (0=bin-exch,1=long,2=mix)
64            swapping threshold
0            L1 in (0=transposed,1=no-transposed) form
0            U  in (0=transposed,1=no-transposed) form
1            Equilibration (0=no,1=yes)
16            memory alignment in double (> 0)
```

Run the test, note that host.list contains a list in the format (hostname:#cpus).

```
# mpirun -np 2  -machinefile host.list xhpl
```
(For more processors, increase the "np" value)

Lots of output, then finally….

```
- Computational tests pass if scaled residuals are less than          16.0

================================================================================
T/V                N    NB     P     Q               Time             Gflops
--------------------------------------------------------------------------------
WR12R2C4        4000   400     1     2              95.53          4.469e-01
--------------------------------------------------------------------------------
||Ax-b||_oo / ( eps * ||A||_1  * N        ) =        0.0152776 ...... PASSED
||Ax-b||_oo / ( eps * ||A||_1  * ||x||_1  ) =        0.0141086 ...... PASSED
||Ax-b||_oo / ( eps * ||A||_oo * ||x||_oo ) =        0.0032067 ...... PASSED
================================================================================

Finished      1 tests with the following results:
              1 tests completed and passed residual checks,
              0 tests completed and failed residual checks,
              0 tests skipped because of illegal input values.
--------------------------------------------------------------------------------

End of Tests.
```

Additional Resources

Links and Mailing lists

Torque and Maui
http://www.supercluster.org/mailing.shtml

MPICH
http://www-unix.mcs.anl.gov/mpi/mpich/

LAM-MPI
http://www.lam-mpi.org/MailArchives/

Linpack
http://www.netlib.org/benchmark/hpl/
http://ppclinux.ncsa.uiuc.edu/

IBM Corporation
Marketing Communications
Systems and Technology Group
Route 100
Somers, New York 10589

Produced in the United States of America
September 2004