



Blueprints

Setting up an HPC cluster with Red Hat Enterprise Linux





Blueprints

Setting up an HPC cluster with Red Hat Enterprise Linux

Note

Before using this information and the product it supports, read the information in “Notices” on page 87.

First Edition (December 2010)

© Copyright IBM Corporation 2010.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Chapter 1. Scope, requirements, and support	1
--	----------

Chapter 2. Open source-based HPC compute cluster on POWER	5
--	----------

Chapter 3. Setting up the cluster	9
Setting up the hpc user	10
Installing Open MPI and InfiniBand packages	10
Configuring InfiniBand	12
Testing InfiniBand connectivity	14

Chapter 4. Installing the Management Node	17
Installing Torque resource manager	17
Installing Maui cluster scheduler	18
Preparing the TORQUE packages for the compute nodes	20
Configuring TORQUE on the compute nodes	22
Validating job submissions	23

Chapter 5. Installing additional advanced software	27
Installing libhugetlbfs	27
Installing IBM Advance Toolchain for PowerLinux	28
Installing the IBM XL C/C++ and Fortran compilers	29
Installing ATLAS math libraries	32

Chapter 6. Building specialized versions of Open MPI	35
Building Open MPI with TORQUE	35
Configuring Open MPI for use with IBM Advance Toolchain for PowerLinux	37
Configuring Open MPI for use with IBM XL compilers	38
Using Open MPI with huge pages.	39

Chapter 7. Testing the InfiniBand interconnect performance with Intel MPI Benchmark	41
Building IMB.	41
Running IMB.	42

Chapter 8. Related information	45
---	-----------

Chapter 9. Setting up an HPC cluster with Red Hat Enterprise Linux	47
---	-----------

Scope, requirements, and support	47
Open source-based HPC compute cluster on POWER	49
Setting up the cluster	51
Setting up the hpc user	52
Installing Open MPI and InfiniBand packages	53
Configuring InfiniBand	55
Testing InfiniBand connectivity	57
Installing the Management Node	58
Installing Torque resource manager	59
Installing Maui cluster scheduler	60
Preparing the TORQUE packages for the compute nodes	62
Configuring TORQUE on the compute nodes	64
Validating job submissions	65
Installing additional advanced software	67
Installing libhugetlbfs	67
Installing IBM Advance Toolchain for PowerLinux	69
Installing the IBM XL C/C++ and Fortran compilers	70
Installing ATLAS math libraries	72
Building specialized versions of Open MPI	73
Building Open MPI with TORQUE	74
Configuring Open MPI for use with IBM Advance Toolchain for PowerLinux	76
Configuring Open MPI for use with IBM XL compilers	76
Using Open MPI with huge pages.	78
Testing the InfiniBand interconnect performance with Intel MPI Benchmark	79
Building IMB.	80
Running IMB.	80
Related information	83

Appendix. Related information	85
--	-----------

Notices	87
Trademarks	88

Chapter 1. Scope, requirements, and support

This blueprint applies to PowerLinux™ (POWER6® and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Systems to which this information applies

PowerLinux (POWER6 and later)

Intended audience

This blueprint is intended for advanced Linux system administrators and programmers who are already familiar with basic HPC concepts, terminology, products, including InfiniBand[®] interconnects. The instructions assume that you have a general understanding of the usage requirements for connecting nodes with InfiniBand and with MPI-based applications.

Scope and purpose

This document is based in part on the collaborative experiences of the IBM® Linux Technology Center, IBM's systems development teams, and Rice University in deploying and managing the POWER7-based BlueBioU cluster, at <http://bluebiou.rice.edu/>.

This document discusses the following software that you can use to deploy a simple cluster with POWER-based systems:

- Basic InfiniBand setup
- Open MPI
- Torque
- IBM Advance Toolchain for PowerLinux
- libhugetlbfs

The following topics are not covered:

- Advanced cluster management products, such as:
 - Moab (Cluster Resources)
 - IBM Parallel Environment
 - Extreme Cloud Administration Toolkit (xCAT)
- Detailed problem determination
- In-depth technology discussion

Resources for in-depth discussion of these and other topics are listed in Related Information

Test environment

The instructions in this blueprint were tested on IBM Power® 575 systems with POWER6 processors running Red Hat Enterprise Linux 5.5. The BlueBioU team at Rice University has deployed a similar software stack with POWER7-based IBM Power 750 systems.

Hardware, software, and other prerequisites

In testing, one server was used as a “management node” and two other servers were the “compute nodes”. Each server had an InfiniBand adapter supporting physical connections to a shared InfiniBand switch.

The examples in this demonstration are focused on eHCA with a single port.

The following software resources are used to perform the steps in this blueprint:

- A Red Hat Enterprise Linux 5.5 YUM repository
- The IBM Installation Toolkit for Linux ISO image from <http://www14.software.ibm.com/webapp/set2/sas/f/lopdiags/installtools/download/>
- Torque Resource Manager software from <http://www.clusterresources.com/torquedocs21/1.1installation.shtml>
- Maui Cluster Scheduler software from <http://www.adaptivecomputing.com/support-old/download-center/maui-cluster-scheduler>

This blueprint focuses on Red Hat Enterprise Linux 5.5, but you can apply the concepts easily to other Linux distributions that are supported on POWER-based systems.

The Red Hat createrepo RPM package must be installed on all nodes of the cluster.

The Red Hat Enterprise Linux 5.5 YUM repository must be available on each of the nodes.

Author names

Brad Benton
Bill Buros
Ric Hendrickson
Jenifer Hopper


Other contributors

Heather Crognale
Jeff George
Monza Lui


IBM Services

Linux offers flexibility, options, and competitive total cost of ownership with a world class enterprise operating system. Community innovation integrates leading-edge technologies and best practices into Linux.

IBM is a leader in the Linux community with over 600 developers in the IBM Linux Technology Center working on over 100 open source projects in the community. IBM supports Linux on all IBM servers, storage, and middleware, offering the broadest flexibility to match your business needs.

For more information about IBM and Linux, go to ibm.com/linux  (<https://www.ibm.com/linux>)

IBM Support

Questions and comments regarding this documentation can be posted on the developerWorks® HPC Central Technical Forum: <http://www.ibm.com/developerworks/forums/forum.jspa?forumID=1056> 

The IBM developerWorks discussion forums let you ask questions, share knowledge, ideas, and opinions about technologies and programming techniques with other developerWorks users. Use the forum content at your own risk. While IBM attempts to provide a timely response to all postings, the use of this developerWorks forum does not guarantee a response to every question that is posted, nor do we validate the answers or the code that are offered.

You can also find additional information in the following forums:

- The developerWorks Linux for Power Architecture® forum: <http://www.ibm.com/developerworks/forums/forum.jspa?forumID=375>
- The developerWorks IBM Installation Toolkit for PowerLinux on POWER® Support Forum: <http://www.ibm.com/developerworks/forums/forum.jspa?forumID=937>

Chapter 2. Open source-based HPC compute cluster on POWER

You can learn about basic concepts of HPC clusters, and get a brief overview of some of the open source software you can use to build an HPC cluster on POWER processor-based systems.

HPC compute clusters

This blueprint is focused on the class of servers which are used primarily for compute-intensive work like computational simulations of weather, bioinformatics, and so on. In this class of compute-intensive workloads, a single compute job may execute on multiple servers simultaneously and often requires extensive communication between each of the servers. In these server clusters, the servers typically share a dedicated high-speed network (often based on InfiniBand), are densely and closely located relative to each other, and are homogeneous.

Management node

A *management node*, typically a separate server, is used for running the cluster's scheduler and resource managers. This blueprint shows how to install a Maui Cluster Scheduler and the Torque Resource Manager which collectively manage and control the compute nodes.

Compute node

With a large number of compute nodes, there is usually a requirement for schedulers and server resource managers to prioritize and manage which jobs can be executed on the cluster. In this blueprint, we focus on just a handful of compute nodes, with the notion that the compute node setup is a repeatable incremental process. As you need to increase the number of compute nodes in your cluster, you can refer to these examples to ensure that the additional nodes are properly prepared.

InfiniBand interconnect

The example cluster in this blueprint uses InfiniBand to connect the nodes of the cluster. InfiniBand is a scalable, low-latency, high performance communication mechanism frequently used in HPC environments.

The software components

These are the essential components of the HPC software stack, including compilers, math libraries, communication stack, cluster management and scheduling software, and software used for installation of key software components for POWER processor-based systems.

IBM Installation Toolkit for PowerLinux

The IBM Installation Toolkit for PowerLinux provides a set of tools that simplifies the installation of IBM value-added software necessary to fully use the capabilities of the Power platform. In particular, for this cluster we are interested in the two 64-bit Open MPI packages included in the IBM Installation Toolkit Version 4.1.

InfiniBand software stack

The InfiniBand software stack provides cluster-knowledgeable middleware, such as MPI libraries, access to the low-latency, high bandwidth InfiniBand fabric for cluster communications. Red Hat Enterprise Linux provides an InfiniBand software stack.

Open MPI

The Open MPI project provides an open source implementation of the MPI-2 standard. It is developed and maintained by a consortium of academic, research, and industry partners. Open MPI is therefore able to combine the expertise, technologies, and resources from across the High Performance Computing community in order to build reliable and high performance MPI library.

IBM Advance Toolchain for PowerLinux

The IBM Advance Toolchain for PowerLinux is a collection of free software application development products specifically tailored to make use of IBM POWER processor features. The package includes GCC (GNU Compiler Collection), GLIBC (GNU C Libraries), GNU binary utilities, GDB (GNU debugger), and the performance analysis tools (QProfile and Valgrind) in a self contained Toolchain. The IBM Advance Toolchain for PowerLinux duplicates the function of packages already delivered by Linux distributors but provides additional features for the POWER6 and POWER7[®] processors as indicated in the IBM Advance Toolchain for PowerLinux Release Notes[®] 'Features' section. The IBM Advance Toolchain for PowerLinux is built and distributed by the University of Illinois.

TORQUE resource manager

Tera-scale Open-source Resource and QUEUE Manager (TORQUE) is a workload and resource-management system that manages batch jobs and compute nodes and schedules the execution of those jobs. The TORQUE system includes three pieces: the server, the scheduler, and the job executor. The server process, called `pbs_server`, creates and manages jobs and queues and dispatches jobs for execution to compute nodes, where the job executor, called `pbs_mom`, starts the job.

Maui Cluster Scheduler

The Maui Cluster Scheduler is a policy engine used to improve the manageability and efficiency of various sized clusters, from small clusters with just a few processors to very large supercomputers. The Maui Cluster Scheduler is intended to support a large array of scheduling policies, dynamic priorities, extensive reservations, and fairshare.

IBM XL C/C++ and XL Fortran compilers

XL C/C++ for Linux V11.1 is a highly advanced optimizing compiler for the selected Linux distributions. The compiler runs on IBM Power Systems[™] and is an industry standards-based professional programming tool that can be used for developing large, complex, computationally intensive 32-bit and 64-bit applications in the C and C++ programming languages.

XL Fortran is a standards-based Fortran compiler with extensive features to help you create high performance applications. A highly advanced optimizing compiler for the Linux operating system, runs on IBM Power Systems.

libhugetlbfs

The libhugetlbfs library interacts with the Linux hugetlbfs to make large pages available to applications in a transparent manner. On POWER processor-based systems, this library provides applications easy access to the 16 MB huge pages.

ATLAS math libraries

Automatically Tuned Linear Algebra Software (ATLAS) provides highly optimized Linear Algebra kernels for arbitrary cache-based architectures. ATLAS provides ANSI C and Fortran77 interfaces for the entire BLAS API, and a small portion of the LAPACK AP.

Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Chapter 3. Setting up the cluster

You can see an overview of the cluster component setup for this example, including whether components are set up on the management node or compute nodes, and which user does the installation or configuration tasks.

The cluster is dependent on each of the compute nodes being connected with InfiniBand. Although it is common that all nodes have an InfiniBand connection, it is not required for the the management node. However, the InfiniBand packages should still be installed on the management node so that other package builds can use the InfiniBand components. Therefore, we start by installing the latest Open MPI and InfiniBand packages on all nodes - the management node and all compute nodes.

The installation and configuration tasks are performed by the root user except where noted in the following tables.

Table 1. Cluster setup tasks

Cluster setup		
Task	Management node	Compute node
Installing OpenMPI and InfiniBand packages	Yes	Yes
Configuring InfiniBand	Not required	Yes
Installing Torque Resource Manager	Yes	No
Installing Maui Cluster Scheduler	Yes	No
Configuring Torque	Yes	Yes

Table 2. Advanced software setup tasks

Advanced software setup		
Task	Management node	Compute node
Installing libhugetlbfs	Yes	Yes
Installing IBM Advance Toolchain for PowerLinux	Yes	Yes
Installing the IBM XL C/C++ and Fortran compilers	Yes	Yes
Installing ATLAS math libraries	Yes	Yes
Building Open MPI with Torque	Yes	Yes
Configuring Open MPI for use with the Advance Toolchain	Yes, as hpc user	Yes, as hpc user
Configuring Open MPI for use with the IBM XL compilers	Yes, as root when installing and then as hpc user when verifying	Yes, as root when installing and then as hpc user when verifying
Using Open MPI with huge pages	Yes, as hpc user	Yes, as hpc user
Testing the InfiniBand interconnect performance with Intel MPI Benchmark (IMB)	Yes, as hpc user when installing and submitting jobs	Yes, as hpc user when installing

Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Setting up the hpc user

The hpc user configures and uses various optional software components on the cluster, including Open MPI, the IBM XL compilers, and the Intel MPI Benchmark (IMB).

About this task

You must ensure that the hpc user exists on all the nodes of the cluster in order to perform some of the configuration and testing tasks described in this blueprint.

The hpc user needs password-less connectivity among all the nodes in order to enable Open MPI applications to pass messages between nodes. Without password-less connectivity, you would be required to enter the password for each node whenever a job was run. The stage needs to be set just so.

Procedure

1. Create the hpc user on any nodes where it does not already exist. If the hpc user does not exist on a node, you can create the hpc user with the following commands:

```
# useradd -m hpc
# passwd hpc
```

2. Set up password-less connectivity for the hpc user among all the nodes in the cluster:

On the management node and all compute nodes, check if hpc user has password-less connectivity. You must be able to use ssh to connect to and from all the nodes without a password request. (There are several methods available to set up password-less connectivity, outside the scope of this document.)

When you have successfully configured ssh for the hpc user, you will be able to use ssh to connect between any two nodes in the cluster without a password. The following example shows a password-less ssh connection from the host named `compute_node1` to the host named `compute_node2`:

```
[compute_node1] # su - hpc
[compute_node1] $ ssh compute_node2

Warning: Permanently added 'compute_node2' (RSA) to the list of known hosts.
[compute_node2] $
```

Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Installing Open MPI and InfiniBand packages

You can obtain and use the IBM Installation Toolkit for PowerLinux to install the Open MPI and InfiniBand software packages.

Before you begin

1. Ensure that the createrepo RPM package is installed on each node of the cluster.

2. Download the IBM Installation Toolkit media (ISO image) from <http://www14.software.ibm.com/webapp/set2/sas/f/lopdiags/installtools/download/>

Note:

- You must register and sign in before you can download the software.
- As of the time of writing of this blueprint, the current IBM Installation Toolkit version is 4.1.

About this task

In these examples we assume the operating system (Red Hat Enterprise Linux 5.5) has already been installed on all of the servers – in this case the management node and the two compute nodes.

Note: The IBM Installation Toolkit can be used to install the Linux distribution for servers as well, but installing the operating system is outside the scope of this discussion.

The root user performs all the steps of this procedure.

Perform these steps for each node of the cluster:

Procedure

1. Create a local IBM Installation Toolkit YUM repository:
 - a. Mount the IBM Installation Toolkit ISO media and copy the Red Hat directory with all of the Red Hat-related RPM packages to a directory of your choice. The following example uses `/opt/ibmit41` as the destination directory:

```
# mkdir /mnt/ibmtoolkit
# mkdir /opt/ibmit41
# mount -o loop -t iso9660 IBM_Installation_Toolkit_41.iso /mnt/ibmtoolkit
# cp -aR /mnt/ibmtoolkit/* /opt/ibmit41
# umount /mnt/ibmtoolkit
# rmdir /mnt/ibmtoolkit
```

Note: This example, uses the `cp` command's `-L` flag order to dereference and copy files that are symbolic links in the `/RedHat` directory on the IBM Installation Toolkit ISO image.

- b. Use the **createrepo** command to create the repository, as in the following example:

```
# createrepo /opt/ibmit41/RedHat
33/33 - RPMS/xconfig-0.1-1.ppc.rpm
Saving Primary metadata
Saving file lists metadata
Saving other metadata
```

- c. Create the definition file for this new repository.

The `/etc/yum.repos.d` directory is the default YUM repository location. Therefore, in this example, the definition file is created as `/etc/yum.repos.d/ibmit41.repo`. The content of the definition file should look like the following example.

Note: The **baseurl** parameter is set to point to the local directory where the **createrepo** command was issued against in the previous step and where the Red Hat RPM packages reside.

```
[ibmit41]
name=ibmit41
baseurl=file:///opt/ibmit41/RedHat
enabled=1
gpgcheck=0
```

- d. Verify that the local repository is set up correctly, as in the following example:

```
# yum repolist
Loaded plugins: rhnplugin, security
This system is not registered with RHN.
RHN support will be disabled.
```

repo id	repo name	status
core-0	core-0	enabled: 32
core-1	core-1	enabled: 33
core-2	core-2	enabled: 3,089
ibmit41	ibmit41	enabled: 33
repolist: 3,187		

2. Install the Open MPI and InfiniBand-related packages from the local IBM Installation Toolkit YUM repository and the Red Hat Enterprise Linux 5.5 YUM repository.

```
# yum install sysstat \
  openmpi-1.4.2-1.ppc64 \
  openib mpi-selector infiniband-diags perftest ofed-docs \
  libibverbs* librdmacm* libehca*
```

Note:

- "sysstat" is a package for extended performance tools on a Power system.
 - The Open MPI package is provided on the IBM Installation Toolkit. This is a newer package of Open MPI built for 64-bit applications which has been packaged and provided for customers.
 - The remainder of the packages are standard packages necessary for MPI applications using InfiniBand connectivity.
 - "ofed" is from the OpenFabrics Alliance – <https://www.openfabrics.org/index.php>. The OpenFabrics Enterprise Distribution (OFED) is a standardized set of enabling software generally targeted at high-performance low-latency computing.
3. Configure the PATH and LD_LIBRARY_PATH environment variables for the hpc user to include the paths for Open MPI and software installed in /usr/local/sbin for this HPC cluster configuration by modifying the /home/hpc/.bashrc file as follows:
 - a. Add the /usr/local/sbin directory and the /opt/openmpi/1.4.2/bin directory to the beginning of the PATH environment variable.
 - b. Add the /opt/openmpi/1.4.2/lib directory to the beginning of the LD_LIBRARY_PATH environment variable.

The following example shows a simple method to append these modifications to the environment variables in the .bashrc file for the hpc user:

```
cat >> /home/hpc/.bashrc <<EOF
export PATH=/usr/local/sbin:/opt/openmpi/1.4.2/bin:$PATH
export LD_LIBRARY_PATH=/opt/openmpi/1.4.2/lib:$LD_LIBRARY_PATH
EOF
```

Results

After you have completed this installation process on each node of the cluster, every system will have the required Open MPI and InfiniBand software installed.

What to do next

You are now ready to configure and validate the InfiniBand connectivity between all the InfiniBand-connected nodes.

Related concepts:

Chapter 1, "Scope, requirements, and support," on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Configuring InfiniBand

You can learn about the steps needed to configure and confirm InfiniBand connectivity.

Before you begin

The Open MPI and InfiniBand packages must already be installed on all nodes.

The allowable amount of locked (pinned) memory needs to be set to unlimited. To do this, edit `/etc/security/limits.conf` and add the following lines:

```
#<domain> <type> <item> <value>
#
*          soft memlock unlimited
*          hard memlock unlimited
```

About this task

These steps assume your servers have InfiniBand adapters installed and are connected to an InfiniBand switch. In this example, all nodes are assumed to be connected to the same InfiniBand fabric.

This procedure is required for all nodes that will have InfiniBand traffic.

Note: You will only perform this process on the management node if the management node has an InfiniBand connection. The root user performs all the steps of this procedure.

Procedure

1. Specify the **nr_ports** and **port_act_time** options for the IBM InfiniBand eHCA kernel module, `ib_ehca`, by creating the `/etc/modprobe.d/ib_ehca.conf` file containing the following line:

```
options ib_ehca nr_ports=-1 port_act_time=120
```

Note:

- The `nr_ports` option is set to -1. This causes the module to autodetect the number of active ports.
 - `port_act_time` option is set to 120. This causes the module to wait for 120 seconds for port activation.
2. Edit `/etc/ofed/openib.conf` and change the values of **SDP_LOAD**, **RDS_LOAD**, **SRP_LOAD**, and **ISER_LOAD** to `no`. This changes the default setting to load only the drivers that are needed for this setup. The resulting file will be similar to the following example:

```
# Load IPoIB
IPOIB_LOAD=yes
# Load SDP module
SDP_LOAD=no
# Load the RDS protocol driver
RDS_LOAD=no
# Load SRP module
SRP_LOAD=no
# Load iSER module
ISER_LOAD=no
# Should we modify the system mtrr registers? We may need to do this if you
# get messages from the ib_ipath driver saying that it couldn't enable
# write combining for the PIO buffs on the card.
FIXUP_MTRR_REGS=no
```

3. Configure the `openibd` service:

- a. Run `chkconfig` to configure the `openibd` service to start automatically whenever the operating system starts, as shown in the following example:

```
# chkconfig openibd on
# chkconfig --list openibd
openibd          0:off  1:off  2:on   3:on   4:on   5:on   6:off
```

Note: The output from `chkconfig --list openibd` in this example shows whether the `openibd` service will start for each runlevel.

- b. Run the **kudzu** command to detect the InfiniBand hardware.

Note: The `/etc/sysconfig/hwconf` file will also be created which is referenced by the `openibd` service.

- c. Start the `openibd` service as shown in the following example:

```
# service openibd start
Loading OpenIB kernel modules: [ OK ]
```

4. Validate InfiniBand device configuration by displaying InfiniBand devices on the node:

- a. To list InfiniBand devices on the node, run the **`ibv_devices`** command. This step validates that the previous steps have been completed and the InfiniBand adapters are correctly detected. If your InfiniBand devices are correctly configured, the output of the **`ibv_devices`** command will look similar to the following example:

```
# ibv_devices
device          node GUID
-----
ehca0           0002550010e56a00
ehca1           0002550010e56c00
```

- b. Use `ibv_devinfo` to query the InfiniBand devices. Verify that the port status is active and that `sm_lid` is not 0 for all connected ports. The following example shows typical output from the `ibv_devinfo` command:

```
# ibv_devinfo

hca_id: ehca0
  node_guid:          0002:5500:7018:9600
  sys_image_guid:     0000:0000:0000:0000
  vendor_id:          0x5076
  vendor_part_id:     1
  hw_ver:              0x2000021
  phys_port_cnt:      2
    port: 1
      state:           PORT_ACTIVE (4)
      max_mtu:          4096 (5)
      active_mtu:       4096 (5)
      sm_lid:           1
      port_lid:         29
      port_lmc:         0x00
    port: 2
      state:           PORT_ACTIVE (4)
      max_mtu:          4096 (5)
      active_mtu:       4096 (5)
      sm_lid:           1
      port_lid:         41
      port_lmc:         0x00
max_mtu:
```

Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Testing InfiniBand connectivity

This series of tests assures connectivity, reasonable bandwidth, and latency performance between a pair of InfiniBand-connected nodes.

Before you begin

Configure the InfiniBand devices on all nodes, as described in “Configuring InfiniBand” on page 12.

About this task

Test the connectivity for each node with at least one other node in the cluster before you continue setting up the cluster.

These tests run between two nodes, one node acting as the server and the other node acting as the client for purposes of the test. The server node initiates a listening server and waits to receive requests from the client. The client node sends the request to the specified server.

The root user performs all the steps of this procedure.

Procedure

1. Test InfiniBand connectivity with the **ibv_rc_pingpong** command:

- a. Run the **ibv_rc_pingpong** command on the server node, as shown in the following example:

```
compute_node1 # ibv_rc_pingpong
local address: LID 0x001c, QPN 0x000033, PSN 0x150335
```

- b. Run the **ibv_rc_pingpong** command on the client node with the hostname of the server node as the parameter. In the following example, the hostname of the server node is `compute_node1`:

```
compute_node2 # ibv_rc_pingpong compute_node1

local address: LID 0x0016, QPN 0x000075, PSN 0x1fa58b
remote address: LID 0x001c, QPN 0x000033, PSN 0x150335
8192000 bytes in 0.02 seconds = 3601.08 Mbit/sec
1000 iters in 0.02 seconds = 18.20 usec/iter
```

If the connection is successful, the **ibv_rc_pingpong** process on the server will produce output indicating the connection from the client, and then exit. The following example shows output on the server that results from a successful connection:

```
remote address: LID 0x0016, QPN 0x000075, PSN 0x1fa58b
8192000 bytes in 0.02 seconds = 3584.34 Mbit/sec
1000 iters in 0.02 seconds = 18.28 usec/iter
```

2. Test the bidirectional InfiniBand bandwidth with the **rdma_bw** command:

- a. On the server node, run **rdma_bw -b** as shown in the following example:

Note: The **-b** flag specifies bidirectional message sending.

```
# rdma_bw -b
```

- b. On the client node, run **rdma_bw -b** with the hostname of the server as the parameter. In the following example, the hostname of the listening server is `compute_node1`:

```
# rdma_bw -b compute_node1
```

```
15151: | port=18515 | ib_port=1 | size=65536 | tx_depth=100 | sl=0 | iters=1000 | \
duplex=1 | cma=0 |
15151: Local address: LID 0x12, QPN 0x4377, PSN 0xb05218 RKey 0x679129 VAddr 0x00040000340000
15151: Remote address: LID 0x11, QPN 0x447b, PSN 0x8b85f6, RKey 0xa81105 VAddr 0x000400001a0000
```

Warning: measured timestamp frequency 511.951 differs from nominal 4704 MHz

```
15151: Bandwidth peak (#0 to #786): 3410.28 MB/sec
15151: Bandwidth average: 3408.49 MB/sec
15151: Service Demand peak (#0 to #786): 146 cycles/KB
15151: Service Demand Avg : 146 cycles/KB
```

The important item to check in this output is the Bandwidth average: value. Throughput greater than 3000 MB/sec is desirable. A bandwidth average value of less than 3000 MB/sec may indicate a performance issue.

Note: The other output and warnings can be safely ignored for the purposes of this test.

3. Test bidirectional InfiniBand latency with the **rdma_lat** command:

- a. On the server node, run the **rdma_lat** command.

```
# rdma_lat
```

- b. On the client node, run the **rdma_lat** command with hostname of the server as the parameter. The hostname of the listening server in the following example is `compute_node1`:

```
# rdma_lat compute_node1
local address: LID 0x12 QPN 0x43b2 PSN 0xb766cf RKey 0x67912e VAddr 0x00000010030001
remote address: LID 0x11 QPN 0x46f6 PSN 0xaa5ce6 RKey 0xa8110b VAddr 0x00000010030001
Warning: measured timestamp frequency 511.938 differs from nominal 4704 MHz
Latency typical: 3.61665 usec
Latency best   : 3.22695 usec
Latency worst  : 28.9488 usec
```

The important item to check in this output is the `Latency typical:` value. A Latency typical value of less than 4.0 usec is desirable.

Note: The other output and warnings can be safely ignored.

Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Chapter 4. Installing the Management Node

Install the TORQUE resource manager and the Maui Cluster Scheduler on the management node in order to control and track jobs on the compute nodes. Build special torque client and Machine Oriented Mini-server (MOM) packages to install on the compute nodes after the you install the Torque and Maui software on the management node.

Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Installing Torque resource manager

You install the TORQUE resource manager only on the management node. These instructions show how to obtain, build and install the newest version of the TORQUE software.

Before you begin

- Verify that the InfiniBand configuration is complete for all nodes by completing the tests in “Testing InfiniBand connectivity” on page 14.
- Ensure that the hpc user and the /home/hpc directory is set up on all nodes as described in “Setting up the hpc user” on page 10.

About this task

The root user on the management node performs all the steps of this procedure.

Procedure

1. As root user on the management node, download the newest version of TORQUE resource manager from:<http://www.clusterresources.com/downloads/torque>

In this example, torque.2.4.8 is the latest available package.

2. Extract, configure, and build the torque software package in the /usr/local/src directory. The following example shows this process and the appropriate flags to use when you configure for the 64-bit Power Architecture:

```
# tar xzf torque-2.4.8.tar.gz -C /usr/local
# cd /usr/local/torque-2.4.8

# ./configure --enable-docs --with-scp --enable-syslog CFLAGS=-m64 --libdir=/usr/local/lib64
# make
# make install
# ldconfig
```

3. Initialize the torque server by running the setup script and indicating the root user:

```
# ./torque.setup root
initializing TORQUE (admin: root@compute_node1.example.com)
Max open servers: 4
Max open servers: 4
```

4. Verify that the /var/spool/torque/server_name file contains the fully qualified host name of the management node.

Note:

a. A short host name will also work if all the nodes in the cluster share the same name server. This should be created during the initialization step.

b. The `/var/spool/torque/server_name` file was created by the **torque.setup** command.

```
# cat /var/spool/torque/server_name  
mgmt_node.example.com
```

5. Build the Torque self-extracting packages for compute nodes. The following example shows the root user running make packages:

```
# make packages
```

6. Copy the self-extracting packages you built for the compute nodes into the `/home/hpc/comput-node-files` directory, as shown in the following example:

Note: You must create the `/home/hpc/compute-node-files` directory if it doesn't already exist.

```
# mkdir /home/hpc/compute-node-files  
# cp *.sh /home/hpc/compute-node-files/
```

7. Run the **libtool** command as shown in the following example to configure the libraries for the management node.

```
# libtool --finish /usr/local/lib64
```

Results

You will use these packages in “Preparing the TORQUE packages for the compute nodes” on page 20.

Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Installing Maui cluster scheduler

The Maui server controls job scheduling across the cluster. These instructions show how to obtain, build, and install the newest version of the Maui cluster scheduler on the management node of the cluster.

Before you begin

Ensure that the TORQUE server is installed on the management node, as described in Installing the TORQUE resource manager.

About this task

The root user on the management node performs all the steps of this procedure.

Procedure

1. Download Maui from the following site: <http://www.clusterresources.com/product/maui/index.php>

Note: you must register in order to download the software

For our example, we selected the Maui 3.3 version, the newest version as of the writing of this document.

2. Extract, configure, and build the Maui server software using the following commands:

- ```
tar -zxf maui-3.3.tar.gz -C /usr/local/src
cd /usr/local/src/maui-3.3
chmod +x /usr/local/torque-2.4.8/pbs-config
./configure --with-pbs=/usr/local CFLAGS=-m64 LDFLAGS=-m64
make
make install
```
3. Add the `/usr/local/maui/bin` directory to the **PATH** environment variable for the `hpc` user by appending the contents of the `/usr/local/src/maui-3.3/etc/maui.sh` file to the `/home/hpc/.bashrc` file.
 

```
cat /usr/local/src/maui-3.3/etc/maui.sh >> /home/hpc/.bashrc
cp /usr/local/src/maui-3.3/etc/maui.sh /home/hpc/compute-node-files/
```
  4. Set up the Maui daemon as a service on the management node server:
    - a. Edit the `/usr/local/src/maui-3.3/etc/maui.d` file and change the value of the **MAUI\_PREFIX** variable to `/usr/local/maui` as shown in the following example:
 

```
MAUI_PREFIX=/usr/local/maui
```
    - b. Copy the `/usr/local/src/maui-3.3/etc/maui.d` file to `/etc/init.d/maui.d`, as shown in the following example:
 

```
cp /usr/local/src/maui-3.3/etc/maui.d /etc/init.d/maui.d
```
  5. Edit the `/usr/local/maui/maui.cfg` file, which was created by the build process, to include the information needed for the environment of your `hpc` cluster:
    - a. Make sure the following lines are included as follows:
 

```
SERVERPORT 42599
ADMIN1 root hpc
ADMIN3 ALL
```
    - b. Modify the `SERVERHOST` and `ADMINHOST` lines to contain the fully qualified host name for the management node. For example, if the fully qualified host name of your management node is `mgmt_node.example.com`, edit these lines as shown in the following example:
 

```
SERVERHOST mgmt_node.example.com
ADMINHOST mgmt_node.example.com
```
    - c. Verify that the `RMCFG` line contains the short host name in brackets, as shown in the following example:
 

```
RMCFG[mgmt_node] TYPE=PBS
```
  6. Create the `/var/spool/torque/server_priv/nodes` file to contain a list of all of the compute nodes in the cluster:
 

```
cd /var/spool/torque/server_priv
vi nodes
```

    - a. Use a text editor to add one line to the `/var/spool/torque/server_priv/nodes` file for each compute node in your cluster. Use the **np** parameter to define the number of CPU threads that are available for use on each node. For example, we have two compute nodes we are defining, each with 64 threads available:
 

```
compute_node1.example.com np=64
compute_node2.example.com np=64
```
    - b. Copy the `/var/spool/torque/server_priv/nodes` file to the `/home/hpc/compute-node-files/` directory.
 

```
cp /var/spool/torque/server_priv/nodes /home/hpc/compute-node-files
```
  7. Set up Maui as a service on the management node using the **chkconfig** command, as shown in the following example:
 

```
chkconfig --add maui.d
chkconfig --level 3456 maui.d on
chkconfig --list maui.d
maui.d 0:off 1:off 2:off 3:on 4:on 5:on 6:on
```
  8. Start the Maui service on the management node as shown in the following example:

```
service maui.d start
Starting MAUI Scheduler: [OK]
```

9. Set up the TORQUE server on the management node by copying the `pbs_server` file to the `/etc/init.d/` directory, as shown in the following example:

```
cd /usr/local/torque-2.4.8
cp -p contrib/init.d/pbs_server /etc/init.d/
```

10. Add the `pbs_server` service to the management node with the **chkconfig** command, as shown in the following example:

**Note:** The `pbs_server` service is the TORQUE resource manager service.

```
chkconfig --add pbs_server
chkconfig --level 3456 pbs_server on
chkconfig --list pbs_server
pbs_server 0:off 1:off 2:off 3:on 4:on 5:on 6:on
```

11. Restart the `pbs_server` service on the management node in order to allow changes to take effect. The `pbs_server` service starts the batch server on the management node that will pick up any jobs. You can start the `pbs_server` service as shown in the following example:

```
service pbs_server restart
Starting TORQUE Server: [OK]
```

## Results

The TORQUE and Maui services are installed and running on the management node. You can query the TORQUE queue and server attributes with the `qmgr -c 'p s'` command.

### Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

---

## Preparing the TORQUE packages for the compute nodes

ou must create Torque Resource Manager installation packages for the compute nodes before you can configure the Machine Oriented Mini-server on the compute nodes.

### Before you begin

Ensure that you have completed all of these prerequisite tasks:

- Build and install the Torque and Maui packages on the management node.
- Build the self-extracting installation packages and copy them to the `/home/hpc/compute-node-files` directory on the management node.
- Create the `hpc` user on the compute nodes as described in “Setting up the `hpc` user” on page 10.

### About this task

The root user on the management node performs these steps.

### Procedure

1. Create the `/usr/local/torque-2.4.8/mom_priv-config` file on the management node:

**Note:** ou will copy this temporary file to the `/home/hpc/compute-node-files` directory.

- a. Provide the following information in the `/usr/local/torque-2.4.8/mom_priv-config` file:
  - IP address of the `pbsserver`, (the same as the IP address of the management node)

- IP address of the clienthost, (the same as the IP address of the management node)
- Settings for important directories that are used by the Machine Oriented Mini-server

The following example of the `mom_priv-config` file content shows how to format this information.

**Note:** The IP address for your management node is specific to your environment. The rest of the file must be the same as this example if you have configured your `hpc` user as described in this document.

```
$pbsserver x.x.x.x # (IP address of management node)
$clienthost x.x.x.x # (IP address of management node)
$usecp */home/hpc /home/hpc
$usecp */home /home
$usecp */root /root
$usecp */tmp.scratch /tmp.scratch
$tmpdir /tmp
```

- b. Copy the `/usr/local/torque-2.4.8/mom_priv-config` file to the `/home/hpc/compute-node-files` directory.
2. Prepare for the configuration of the `pbs_mom` service on each compute node. The **`pbs_mom`** command starts a batch Machine Oriented Mini-server that will control job execution as directed by the TORQUE resource manager service, usage limits, and notifications.
  - a. Update `/usr/local/torque-2.4.8/contrib/init.d/pbs_mom` file on the management node by adding `ulimit -l unlimited` as shown in the following example `pbs_mom` file:
 

```
#!/bin/sh
#
pbs_mom This script will start and stop the PBS Mom
#
chkconfig: 345 95 5
description: TORQUE/PBS is a versatile batch system for SMPs and clusters
#
ulimit -n 32768
ulimit -l unlimited # <--- add this line
Source the library functions
. /etc/rc.d/init.d/functions
```
  - b. Copy the `/usr/local/torque-2.4.8/contrib/init.d/pbs_mom` to the `/home/hpc/compute-node-files` directory.
  - c. Use the `cp -p` command to preserve mode and ownership of the file as shown in the following example:
 

```
cp -p contrib/init.d/pbs_mom /home/hpc/compute-node-files/
```
3. Copy the `/home/hpc/compute-node-files` directory as an archive file from the management node onto each of the compute nodes:
  - a. Create a tar archive of the contents of the `/home/hpc/compute-node-files` directory, as shown in the following example:

**Note:** All of the files shown in this example must be included in the tar file.

```
cd /home/hpc
tar zcvf compute-node-files.tar.gz compute-node-files
compute-node-files/
compute-node-files/torque-package-clients-linux-powerpc64.sh
compute-node-files/torque-package-devel-linux-powerpc64.sh
compute-node-files/torque-package-doc-linux-powerpc64.sh
compute-node-files/torque-package-mom-linux-powerpc64.sh
compute-node-files/torque-package-server-linux-powerpc64.sh
compute-node-files/nodes
compute-node-files/maui.sh
compute-node-files/mom_priv-config
compute-node-files/pbs_mom
```

- b. Copy the archive to the /home/hpc directory on each compute node. In the following example there are two compute nodes in the cluster, with the host names compute\_node1 and compute\_node2:

```
scp compute-node-files.tar.gz compute_node1:/home/hpc
scp compute-node-files.tar.gz compute_node2:/home/hpc
```

## Results

You are ready to extract and install the TORQUE packages on each compute node.

### Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

---

## Configuring TORQUE on the compute nodes

In order to use the Torque Resource Manager to control jobs in the cluster, you need to configure the Machine Oriented Mini-server on each compute node.

### Before you begin

Complete the procedure described in “Preparing the TORQUE packages for the compute nodes” on page 20.

### About this task

You need to follow these steps on each compute node.

The root user on the compute node performs this procedure.

### Procedure

1. Extract the compute-node-files.tar.gz archive and install each TORQUE package as shown in the following example:

```
cd /home/hpc

tar xzf compute-node-files.tar.gz

cd compute-node-files/

./torque-package-clients-linux-powerpc64.sh --install
Installing TORQUE archive...

Done.

./torque-package-mom-linux-powerpc64.sh --install
```

```
Installing TORQUE archive...

Done.
```

2. Copy the Machine Oriented Mini-server files to the appropriate directories with the following commands:

```
cp /home/hpc/compute-node-files/mom_priv-config /var/spool/torque/mom_priv/config
cp /home/hpc/compute-node-files/pbs_mom /etc/init.d/
```

3. Prepare the pbs\_mom service with the following commands:

```
chkconfig --add pbs_mom
chkconfig --level 3456 pbs_mom on
chkconfig --list pbs_mom
```

4. Start the pbs\_mom service with the following command:

```
service pbs_mom start
```

### Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

---

## Validating job submissions

Verify that TORQUE resource manager and Maui Cluster Scheduler are properly configured by submitting a basic test job and a multi-node test job.

### Before you begin

Ensure that you have completed the following tasks:

- “Installing Maui cluster scheduler” on page 18
- “Installing Torque resource manager” on page 17
- “Preparing the TORQUE packages for the compute nodes” on page 20
- “Configuring TORQUE on the compute nodes” on page 22

Restart the operating system on each node.

### About this task

The hpc user submits the test jobs on the cluster.

**Note:** The root user cannot submit the jobs on the cluster because of the TORQUE security controls.

The job basic job test and multi-node job test verify resource handling and job scheduling on your nodes.

### Procedure

1. Verify that the Maui and TORQUE services are running with the following steps. Because the torque and maui software have been configured as services, the processes should automatically start after rebooting.

- a. On the management node, verify that the maui service and the pbs\_server service are running. When these services are running, you can see output from the ps command similar to the output shown in the following example:

```
ps -ef | grep -E "pbs|maui"
root 4091 1 0 11:16 ? 00:00:00 /usr/local/maui-3.3/sbin/maui
root 4525 1 0 11:16 ? 00:00:00 /usr/local/sbin/pbs_server
root 7253 7077 0 11:28 pts/1 00:00:00 grep -E pbs|maui
```

- b. On the compute nodes, verify that the pbs\_mom service is running. When this service is running, you can see output from the ps command similar to the output shown in the following example:

```
ps -ef | grep pbs
root 4467 1 0 11:16 ? 00:00:00 /usr/local/sbin/pbs_mom -p
root 7144 7046 0 11:26 pts/1 00:00:00 grep pbs
```

2. Perform a basic job test on the management node with the following steps:

- a. Submit the job using the qsub command and view the status of the job using the qstat command as the hpc user on the management node, as shown in the following example:

```
$ echo "sleep 30" | qsub
1.mgmt_node.example.com
```

```
$ qstat
Job id Name User Time Use S Queue

1.mgmt_node STDIN hpc 0 R batch
```

The output of the `qstat` command shows the following characteristics of the job:

**Job id** The Job id field shows the ID associated with the job and the node where the job is running. The ID of the job running on the `mgmt_node` node in this example is 1.

**Name** The Name field shows the name defined on the `#PBS -N` line in the PBS job file. If no job name is defined in the PBS job file, the default value of the Name field is the file name of the job. In this example, the job is from standard input, so the job name is `STDIN`.

**User** The User field shows the name of the user that submitted the job.

#### Time Use

The Time Use field shows the amount of CPU time used by the job.

**S** The S field shows the status of the job. In this example, the value of the S field is R, indicating that the job is running. When the job is complete, the value of the status field is C.

**Note:** If the value of the status column is Q (queued) when you run this test job, there is probably a configuration issue. If this occurs, review the installation and configuration steps to ensure they were completed correctly.

#### Queue

The Queue field shows the name of the queue that is handling the job. In this example, the job is the queue named `batch`.

3. To verify that the cluster is configured correctly, use the `mpirun` command to run the `hostname` command on each node to ensure that processes run successfully on each node. The parameters used in the following example with the `mpirun` command are:

- The `-np` flag specifies the total number of copies of the process to run on the nodes.
- The `-host` flag specifies the host names where the process will run.
- The last parameter of the `mpirun` command is the command to run on the nodes.

The following example shows how to use the `mpirun` command to run a simple process on all the nodes of your cluster.

```
$ mpirun -np 2 --host compute_node1.example.com,compute_node2.example.com hostname
compute_node1.example.com
compute_node2.example.com
```

If the `hostname` command runs successfully on each node, the host name of each node appears in the output of the `mpirun` command shown in this example. In this example, the output shows the host names of the two compute nodes in the test cluster, `compute_node1`, and `compute_node2`.

4. Perform a simple multi-node test using a PBS script with the following steps:
  - a. On the management node, create a PBS script similar to the following example. Note the following in the example:
    - The lines beginning with the number sign (`#`) in the PBS script are TORQUE directives and are used to set TORQUE environment variables for the job.
    - In the following example, the file name of the PBS script is `multinode.pbs`.
    - In the following example, the host names of the compute nodes are `compute_node1` and `compute_node2`.
    - In the following example, the `-np` flag of the `mpirun` command is used to specify that two copies of the `hostname` command will run.

```
#PBS -N node_test
#PBS -q batch
#PBS -l nodes=2,pmem=100m,walltime=00:30:00
#PBS -M admin@example.com
#PBS -m e
#PBS -V
```

```
mpirun -np 2 --host compute_node1,compute_node2 hostname
```

- b. Submit the PBS script as the hpc user on the management node, and check the status of the job as shown in the following example:

```
$ qsub multinode.pbs
11.mgmt_node.example.com
[hpc@mgmt_node ~]$ qstat
```

| Job id      | Name      | User | Time Use | S | Queue |
|-------------|-----------|------|----------|---|-------|
| 1.mgmt_node | node_test | hpc  | 00:00:00 | C | batch |

This example shows a job submission, the returned job number (1), and the qstat output which indicates whether the job is running or has already completed.

### Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.





---

## Chapter 5. Installing additional advanced software

There are several advanced software packages you can use to enhance the capability of compute nodes for a POWER processor-based cluster. This discussion includes basic installation and configuration instructions for libhugetlbfs, the IBM Advance Toolchain for PowerLinux, the ATLAS math libraries, and the IBM XL C/C++ and IBM XL Fortran compilers

### Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

---

### Installing libhugetlbfs

You can boost performance for some HPC workloads by using 16MB huge pages provided by the libhugetlbfs library.

### About this task

You must install and configure all nodes (including the management node) because the libhugetlbfs environment variables are applied on the management node for jobs that run on the compute nodes.

The root user performs all the steps of this task.

### Procedure

1. Download the newest libhugetlbfs library from the project home, at the following URL:  
`http://sourceforge.net/projects/libhugetlbfs/files/`
2. Prepare and install the library. Note that libhugetlbfs build process will create both the 32-bit and the 64-bit versions on the Power architecture.
  - a. Extract the archive file, as shown in the following example:  

```
tar -C /usr/local/src -zxf libhugetlbfs-2.9.tar.gz
```
  - b. Change to the top-level directory of the extracted source, as shown in the following example:  

```
cd /usr/local/src/libhugetlbfs-2.9/
```
  - c. Build and install the software package, as shown in the following example:  

```
make
make install PREFIX=/usr/local
```
3. Create and mount the /libhugetlbfs file system.  

```
mkdir /libhugetlbfs
mount -t hugetlbfs hugetlbfs /libhugetlbfs
```
4. Give the hpc user group privileges to use /libhugetlbfs.
  - a. Create the libhuge group, as shown in the following example:  

```
groupadd libhuge
```
  - b. Make the hpc user a member of the libhuge group, as shown in the following example:  

```
usermod -a -G libhuge hpc
```
  - c. Give the libhuge group privileges to use the /libhugetlbfs directory and set the permissions as shown in the following example:  

```
chgrp libhuge /libhugetlbfs
chmod 777 /libhugetlbfs
```

5. Reserve huge pages for compute jobs to use, by providing the number of desired pages in the `nr_hugepages` file. Clear any existing huge pages before you reserve a new number of huge pages, as shown in the following example:

```
echo 0 > /proc/sys/vm/nr_hugepages
echo 100 > /proc/sys/vm/nr_hugepages
```

6. Validate the configuration of `libhugetlbfs`.

- a. Verify that the permissions of the `/libhugetlbfs` file system are set correctly, as shown in the following example:

```
ls -ld /libhugetlbfs/
drwxrwxrwx 2 root libhuge 0 Oct 8 20:40 /libhugetlbfs/
```

- b. Verify that the `libhuge ld` scripts are located in the directory where `libhugetlbfs` is installed, as shown in the following example:

```
ls /usr/local/share/libhugetlbfs/
ld ld.hugetlbfs ldscripts
```

- c. Verify that the correct number of huge pages is allocated by checking `/proc/meminfo` as shown in the following example:

```
cat /proc/meminfo | grep Huge
```

7. Make the settings for `libhugetlbfs` persistent by having the configuration occur when the node starts up.

- a. Edit the `/etc/fstab` file to include the following line:

```
hugetlbfs /libhugetlbfs hugetlbfs mode=0777,gid=1000 0 0
```

**Note:** The group ID for the `libhuge` group in this example is 1000. Verify the `libhuge` group ID by running the `id` command as the `hpc` user.

- b. Edit the `/etc/sysctl.conf` file to include the following parameters:

```
vm.nr_hugepages=0
vm.nr_hugepages=100
```

- c. Run the **`sysctl -p`** command to load settings from the default `sysctl` configuration file, `/etc/sysctl.conf`.

## Results

When you have completed this process for each node, you can use 16MB huge pages on your cluster.

### Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

---

## Installing IBM Advance Toolchain for PowerLinux

The IBM Advance Toolchain for PowerLinux provides potential performance benefits by using newer open source compilers and runtime libraries than those that are provided in most Linux distributions.

### About this task

If you determine that your applications will benefit from the IBM Advance Toolchain for PowerLinux, perform this process on each node of the cluster.

The root user performs the installation process.

The `hpc` user must perform some of the of the validation steps.

## Procedure

1. Download the three IBM Advance Toolchain for PowerLinux 3.0 RPM packages for Red Hat Enterprise Linux from the following URL: [ftp://linuxpatch.ncsa.uiuc.edu/toolchain/at/at3.0/redhat/RHEL5](http://linuxpatch.ncsa.uiuc.edu/toolchain/at/at3.0/redhat/RHEL5)

The following list shows the names of the specific files to download:

- advance-toolchain-at3.0-runtime-3.0-0.ppc64.rpm
- advance-toolchain-at3.0-devel-3.0-0.ppc64.rpm
- advance-toolchain-at3.0-perf-3.0-0.ppc64.rpm

2. Install the RPM packages in the order shown in the following example:

```
rpm -i advance-toolchain-at3.0-runtime-3.0-0.ppc64.rpm
rpm -i advance-toolchain-at3.0-devel-3.0-0.ppc64.rpm
rpm -i advance-toolchain-at3.0-perf-3.0-0.ppc64.rpm
```

The default installation location for the IBM Advance Toolchain for PowerLinux is the `/opt/at3.0/` directory. The IBM Advance Toolchain for PowerLinux **gcc** compilers are installed in the `/opt/at3.0/bin` directory by default.

3. Run the **createldhuge** script to configure IBM Advance Toolchain for PowerLinux to work with libhugetlbfs, as shown in the following example:

```
/opt/at3.0/scripts/createldhuge.sh \
/usr/local/share/libhugetlbfs/ld.hugetlbfs /opt/at3.0/
```

4. Confirm the version of the **gcc** command installed by the Linux distribution is the **gcc** executable provided by the Linux distribution.

```
which gcc
/usr/bin/gcc
```

5. Make the IBM Advance Toolchain for PowerLinux gcc available for the hpc user.

- a. Add the IBM Advance Toolchain for PowerLinux bin directory in the PATH environment variable for the hpc user so that the hpc user will use this version of the **gcc** command by default, as shown in the following example:

```
su - hpc
$ export PATH=/opt/at3.0/bin:$PATH
```

- b. Verify that the IBM Advance Toolchain for PowerLinux version of gcc is now the default gcc command used by the hpc user, as shown in the following example:

```
$ which gcc
/opt/at3.0/bin/gcc
```

## Results

If you complete this process on all the nodes, the cluster is configured to use the IBM Advance Toolchain for PowerLinux with libhugetlbfs.

### Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

---

## Installing the IBM XL C/C++ and Fortran compilers

Users who want the advantages of optimized compilers may benefit from the IBM XL compilers.

### Before you begin

You can find more information about getting a license for these compilers at the following web sites:

- XL C/C++ for Linux, V11.1 : <http://www.ibm.com/software/awdtools/xlcpp/linux/>

- XL Fortran for Linux, V13.1: <http://www.ibm.com/software/awdtools/fortran/xlfortran/linux/>

**Note:** Users can also take advantage of the free 60-day trial for the IBM XL compilers. Here we provide instructions on how to download and install the trial versions.

You must ensure that the required compiler toolchain packages from the Linux distributor are installed on the system before you install the IBM XL compilers. You can do this with the **yum install** command as shown in the following example:

```
yum install gcc gcc-c++ glibc.ppc glibc.ppc64 glibc-devel.ppc glibc-devel.ppc64 \
libgcc.ppc libgcc.ppc64 libstdc++.ppc libstdc++.ppc64 libstdc++-devel.ppc \
libstdc++-devel.ppc64 perl
```

## About this task

If you choose to use the XL C/C++ and XL Fortran compilers, perform this process on each node of the cluster.

The root user performs the installation process.

## Procedure

1. Download the XL compiler trial versions from the following URLs:

- XL C/C++ for Linux, V11.1: [https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=swg-xlcc111&S\\_CMP=rnav](https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=swg-xlcc111&S_CMP=rnav)
- XL Fortran for Linux, V13.1: [https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=swg-xlf131&S\\_CMP=rnav](https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=swg-xlf131&S_CMP=rnav)

**Note:** You have to accept the license agreement for both compilers before you can go to the **Download using http** tab and click **Download now** to download the compiler archive files.

2. Install the XL C/C++ compiler.

- a. Prepare and install the XL C/C++ compiler from the directory where you extract the archive file, as shown in the following example. This process will install the compiler in the `/opt/ibmcmp/` directory by default:

```
mkdir /usr/local/src/xlc
tar -C /usr/local/src/xlc -zxf xlc.11.1.0.0.linux.eval.tar.gz
cd /usr/local/src/xlc/
./xlc_install -rpmloc images/rpms/
```

The following example output shows the portions of the **xlc\_install** command output that you need to interact with the installation process. The places where output was omitted from this example are indicated by the three periods ("..."):

Press Enter to continue viewing the license agreement, or, Enter "1" to accept the agreement, "2" to decline it or "99" to go back to the previous screen, "3" Print, "4" Read non-IBM terms.

1

...

| Link             | Source                             |
|------------------|------------------------------------|
| /usr/bin/gxlc    | /opt/ibmcmp/vacpp/11.1/bin/gxlc    |
| /usr/bin/gxlc++  | /opt/ibmcmp/vacpp/11.1/bin/gxlc++  |
| /usr/bin/gxlc    | /opt/ibmcmp/vacpp/11.1/bin/gxlc    |
| /usr/bin/xlc     | /opt/ibmcmp/vacpp/11.1/bin/xlc     |
| /usr/bin/xlc++   | /opt/ibmcmp/vacpp/11.1/bin/xlc++   |
| /usr/bin/xlc     | /opt/ibmcmp/vacpp/11.1/bin/xlc     |
| /usr/bin/xlc_r   | /opt/ibmcmp/vacpp/11.1/bin/xlc_r   |
| /usr/bin/xlc++_r | /opt/ibmcmp/vacpp/11.1/bin/xlc++_r |
| /usr/bin/xlc_r   | /opt/ibmcmp/vacpp/11.1/bin/xlc_r   |

```

=====
Do you want to proceed with the symbolic links?
Please type "yes" or "no": yes

```

...

Installation Completed.  
Filename of the installation log: /opt/ibmcomp/vacpp/11.1/xlc\_install.log

- b. Run the **xlc** command to verify that you see the compiler information page, as shown in the following example:

```

xlc
xlc(1) IBM XL C/C++ for Linux, V11.1 xlc(1)

NAME
 xlc, xlc++, xlc, cc, c89, c99 and related commands - invoke the IBM XL
 C/C++ compiler.

SYNTAX
 <invocation-command> [<option> | <inputfile>]

```

...

### 3. Install the XL Fortran compiler.

- a. Prepare and install the XL Fortran compiler from the directory where you extract the archive file, as shown in the following example. This process will install the compiler in the /opt/ibmcomp/ directory by default:

```

mkdir /usr/local/src/xlf
tar -C /usr/local/src/xlf -zxf xlf.13.1.0.0.linux.eval.tar.gz
cd /usr/local/src/xlf/
./xlf_install -rpmloc images/rpms/

```

The following example output shows the portions of the **xlf\_install** command output which you need to interact with the installation process. The places where output was omitted from this example are indicated by the three periods ("..."):

```

Type "yes" even if you have "IBM XL SMP Version 2.1.0.0-100630 for Linux",
"IBM XL MASS Version 6.1.0.0-100630 for Linux" already installed at the
default installation path.
Do you want to proceed with the installation?
Please type "yes" or "no": yes

```

...

Press Enter to continue viewing the license agreement, or, Enter "1" to accept the agreement, "2" to decline it or "99" to go back to the previous screen, "3" Print, "4" Read non-IBM terms.

1

...

```

Do you want to proceed with the symbolic links?
Please type "yes" or "no": yes

```

...

Installation Completed.  
Filename of the installation log: /opt/ibmcomp/xlf/13.1/xlf\_install.log

- b. Run the **xlf** command to verify that you see the compiler information page, as shown in the following example:

```

xlf
xlf(1) IBM XL Fortran for Linux, V13.1 xlf(1)

NAME
 xlf, xlf_r, f77, fort77, xlf90, xlf90_r, f90, xlf95, xlf95_r, f95,
 xlf2003, xlf2003_r, f2003 - invoke the IBM XL Fortran compiler.

SYNTAX
 <invocation-command> [<option> | <inputfile>]

```

...

## Results

If you complete this process on all the nodes of your cluster, the XL C/C++ and XL Fortran compilers are installed and ready to use.

### Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

---

## Installing ATLAS math libraries

Automatically Tuned Linear Algebra Software (ATLAS) is a math library that can be used to compile workloads such as Linpack and HPC Challenge.

### Before you begin

Download the newest stable version of ATLAS, which is 3.8.3 at the time this paper was written:  
<http://sourceforge.net/projects/math-atlas/files/>

**Note:** At the time this document was written, the newest stable version was ATLAS 3.8.3, which is used in the examples in this document.

### About this task

The build process takes several hours to complete, so you may prefer to compile the software on one node and copy the compiled versions to your other nodes with the identical processor architecture.

The root user performs all the steps of this process.

### Procedure

1. Extract the ATLAS tar archive file, as shown in the following example:

```
tar -jxf atlas3.8.3.tar.bz2 -C /usr/local/
```

2. Configure the ATLAS setup.

As root, go into the top level ATLAS directory and create a build directory where you will run the configure and make commands, as shown in the following example:

```
cd /usr/local/ATLAS/
mkdir obj64
cd obj64/
../configure -b 64
```

- The example uses a directory called obj64 as the 64-bit library build location.
- The -b 64 option tells the ATLAS **configure** command to build the 64-bit libraries.

3. Run the make step to build, test, and optimize the library.

**Note:** This step may take several hours to complete. The ATLAS math libraries employ a self-tuning automated optimization process of building, running, assessing the results, and using the best options. The following example shows the invocation of the **make** command, and modified output with the last several lines of message produced by the **make** command. The omitted lines of output are represented by a line containing three periods (“...”) in the example output.

```
make
...
ATLAS install complete. Examine
ATLAS/bin/<arch>/INSTALL_LOG/SUMMARY.LOG for details.
make[1]: Leaving directory `/usr/local//ATLAS/obj64'
```

```
make clean
make[1]: Entering directory `/usr/local/ATLAS/obj64'
rm -f *.o x* config?.out *core*
make[1]: Leaving directory `/usr/local/ATLAS/obj64'
```

4. After the **make** command completes, run the **make check** command to check the libraries. This command produces a report showing a list of tests that either pass or fail. Ensure that all of the tests pass.
5. Install the libraries. The following example shows a listing of the libraries that the **make install** command installs in the `/usr/local/ATLAS/obj64/lib/` directory:

```
make install
ls lib/
libatlas.a libf77blas.a libptcbblas.a libtstatlas.a Make.inc
libcblas.a liblapack.a libptf77blas.a Makefile
```

## Results

After you install the ATLAS libraries on the nodes of the cluster, the libraries are ready to use.

## Example

The order you use to link to the ATLAS libraries can affect how they function. The ATLAS FAQs provide more information about this subject. The following example shows a correct order to link the libraries:

```
-lc -L/usr/local/ATLAS/obj64/lib/ -llapack -lptcbblas -lptf77blas -latlas
```

### Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.





---

## Chapter 6. Building specialized versions of Open MPI

You can configure Open MPI to integrate better and make better use of other components in an HPC software stack.

Open MPI is highly configurable both at build time and at run time. Although Open MPI will function well on POWER-based systems as installed from the IBM Installation Toolkit, you can optimize Open MPI to work better with other software that is installed on the HPC cluster. This topic discusses methods to configure Open MPI to work better with various software components that are discussed in this blueprint. Specifically, this discussion covers integrating the following packages with Open MPI:

- TORQUE resource manager
- IBM Advance Toolchain for PowerLinux
- IBM XL C/C++ and Fortran compilers
- libhugetlbfs

### Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

---

## Building Open MPI with TORQUE

Configuring Open MPI to work directly with the TORQUE resource manager can simplify launching and managing MPI jobs across a set of resources.

### Before you begin

Ensure that the TORQUE resource manager is installed on the nodes in your cluster, as described in Chapter 4, “Installing the Management Node,” on page 17.

### About this task

The hpc user performs most of the steps in this task.

The root user performs the build and installation processes in this task.

This procedure can be used to install any new versions of Open MPI.

The process shows how to build and make the package on the management node. You can then create a tar archive file of the /usr/local/openmpi library and then copy the archive file to each compute node and then extract the files. After these steps, you must change the PATH and LD\_LIBRARY settings in the .bashrc file to use the new library.

### Procedure

1. On the management node, download the latest Open MPI source tar file in the 1.4.x release series from the following location: <http://www.open-mpi.org/software/ompi/v1.4/>

**Note:** This example, uses Open MPI version 1.4.2, which is the latest stable release of Open MPI at the time of this writing.

2. As the root user on the management node, extract the sources and go to the source directory, as shown in the following example:

```
tar xjf openmpi-1.4.2.tar.bz2 -C /usr/local/src
cd /usr/local/src/openmpi-1.4.2
```

3. On the management node, configure and build a Torque-knowledgeable version of Open MPI. Do this by using the **--with-tm** flag for the configure script. Specifically, set the value of the **--with-tm** flag to the directory location where Open MPI can reference include/tm.h. If you have followed the instructions in this blueprint so far, the directory to use is /usr/local/torque-2.4.8/src as shown in the following example. After the configure step, run the **make** command and the **make install** command.

```
#!/configure --prefix=/usr/local/openmpi-1.4.2-tm \
--with-platform=contrib/platform/ibm/optimized-ppc64-gcc --with-tm=/usr/local/torque-2.4.8/src
make
make install
```

4. On the management node, update the relevant environment variables of the hpc user to enable use of the TORQUE resource manager-aware version of Open MPI as shown in the following example:

```
su - hpc
$ export PATH=/usr/local/openmpi-1.4.2-tm/bin:$PATH
$ export LD_LIBRARY_PATH=/usr/local/openmpi-1.4.2-tm/lib:$LD_LIBRARY_PATH
```

**Note:** You may also remove the previous version of Open MPI from PATH and LD\_LIBRARY\_PATH. However the syntax in this example sets the updated library to be found first in the path, so any older library in the path has no effect.

You can use the following example to make these updates to the .bashrc file:

```
$ cat >> /home/hpc/.bashrc <<EOF
export PATH=/usr/local/openmpi-1.4.2-tm/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/openmpi-1.4.2-tm/lib:$LD_LIBRARY_PATH
EOF
```

5. On the management node, run **ompi\_info** and use the **grep** command to retrieve the lines containing "plm" to ensure that the new build of Open MPI has TORQUE integration. If the TORQUE manager component built correctly, you will see the tm entry listed as a component of the plm (Process Launch Module) framework, as shown in the following example:

```
$ ompi_info | egrep plm:
 MCA plm: rsh (MCA v2.0, API v2.0, Component v1.4.2)
 MCA plm: slurm (MCA v2.0, API v2.0, Component v1.4.2)
 MCA plm: tm (MCA v2.0, API v2.0, Component v1.4.2)
```

6. On the management node, as the root user, prepare the tar archive file of the openmpi-1.4.2-tm directory contains the updated Open MPI library, and distributed the archive to all of the compute nodes, as shown in the following example:

```
cd /usr/local
tar cvf openmpi-1.4.2-tm.tar openmpi-1.4.2-tm
scp openmpi-1.4.2-tm.tar compute_node1:/usr/local/
scp openmpi-1.4.2-tm.tar compute_node2:/usr/local/
```

**Note:** In this example, the compute nodes are named compute\_node1 and compute\_node2.

7. On each of the compute nodes, install the TORQUE resource manager-aware version of Open MPI, and set up the environment for the hpc user.
  - a. As the root user on each compute node, extract the tar archive, as shown in the following example

```
cd /usr/local
tar xvf openmpi-1.4.2-tm.tar
```

- b. As the hpc user on each compute node, configure the appropriate environment variables, as shown in the following example:

```
$ export PATH=/usr/local/openmpi-1.4.2-tm/bin:$PATH
$ export LD_LIBRARY_PATH=/usr/local/openmpi-1.4.2-tm/lib:$LD_LIBRARY_PATH
```

You can use the following example to make these updates to the .bashrc file:

```
$ cat >> /home/hpc/.bashrc <<EOF
export PATH=/usr/local/openmpi-1.4.2-tm/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/openmpi-1.4.2-tm/lib:$LD_LIBRARY_PATH
EOF
```

- c. On each compute node, run **ompi\_info** and use the **grep** command to retrieve the lines containing "plm" to ensure that the new build of Open MPI has TORQUE integration as shown in the following example:

```
$ ompi_info | egrep plm:
MCA plm: rsh (MCA v2.0, API v2.0, Component v1.4.2)
MCA plm: slurm (MCA v2.0, API v2.0, Component v1.4.2)
MCA plm: tm (MCA v2.0, API v2.0, Component v1.4.2)
```

---

## Configuring Open MPI for use with IBM Advance Toolchain for PowerLinux

You can configure Open MPI for use with IBM Advance Toolchain for PowerLinux with a single command string.

### Before you begin

Ensure that the Open MPI and the IBM Advance Toolchain for PowerLinux RPMs are installed on the system.

### About this task

The following procedure should be completed on all nodes.

### Procedure

1. Adjust the PATH environment variable so that the Advance Toolchain installation bin directory is at the head of the PATH list, as shown in the following example:

```
$ export PATH=/opt/at3.0/bin:$PATH
```

Once this is done, all builds done with the Open MPI wrapper compilers (such as mpicc, mpif90, and so on), regardless of which Open MPI build you are using, will use the compilers and runtime environment provided by the IBM Advance Toolchain for PowerLinux.

2. Validate the use of IBM Advance Toolchain for PowerLinux with the following steps:
  - a. Run the **which gcc** command and ensure that the **gcc** used is coming from the /opt/at3.0/bin directory, as shown in the following example:

```
$ which gcc
/opt/at3.0/bin/gcc
```
  - b. Build an MPI program using the mpicc compiler wrapper (now using IBM Advance Toolchain for PowerLinux) and run the **ldd** command on the resulting executable. Make sure that the libc.so.6 library (and other libraries) resolve to libraries in the /opt/at3.0/lib64 directory hierarchy:

#### Note:

```
$ mpicc -o hello_c hello_c.c
$ ldd hello_c
ldd hello_c
linux-vdso64.so.1 => (0x0000000000100000)
libmpi.so.0 => /usr/local/openmpi-1.4.2-tm/lib/libmpi.so.0 (0x0000040000040000)
libopen-rte.so.0 => /usr/local/openmpi-1.4.2-tm/lib/libopen-rte.so.0 (0x0000040000140000)
libopen-pal.so.0 => /usr/local/openmpi-1.4.2-tm/lib/libopen-pal.so.0 (0x00000400001c0000)
libdl.so.2 => /opt/at3.0/lib64/libdl.so.2 (0x0000040000250000)
libnsl.so.1 => /opt/at3.0/lib64/libnsl.so.1 (0x0000040000270000)
libutil.so.1 => /opt/at3.0/lib64/libutil.so.1 (0x00000400002a0000)
libm.so.6 => /opt/at3.0/lib64/power6/libm.so.6 (0x00000400002c0000)
```

```
libpthread.so.0 => /opt/at3.0/lib64/power6/libpthread.so.0 (0x00000400003a0000)
libc.so.6 => /opt/at3.0/lib64/power6/libc.so.6 (0x00000400003e0000)
/opt/at3.0/lib64/ld64.so.1 (0x0000040000000000)
```

#### Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

---

## Configuring Open MPI for use with IBM XL compilers

Put your short description here; used for first paragraph and abstract.

### Before you begin

Ensure that Open MPI and the IBM XL Compiler RPMs are installed on the system.

### About this task

This procedure will enable the use of Open MPI to build and run HPC applications with the IBM XL compilers. The use of the XL compilers can provide significant performance improvements over the distro-supplied compilers.

This procedure does not require that Open MPI be reconfigured or rebuilt. It relies solely on enhancing the Open MPI implementation to be knowledgeable of the XL compilers, therefore it can be done for either Open MPI build (the rpm installed version or the version integrated with torque).

The procedure given below is specifically for the xlc compiler. However, the same procedure can be followed to enable all of the languages and variants supported by the XL compilers to be used with Open MPI.

On all nodes, as the root user, follow the procedures below.

### Procedure

1. To create a wrapper compiler to integrate the use of xlc with Open MPI, complete the following steps:

a. Run the following command:

```
ompi_install_dir=/usr/local/openmpi-1.4.2-tm/
```

b. In the Open MPI binary directory, symbolically link opal\_wrapper to the desired wrapper compiler name (in this case, mpixlc):

```
ln -s ${ompi_install_dir}/bin/opal_wrapper ${ompi_install_dir}/bin/mpixlc
```

c. In the Open MPI share/openmpi directory, copy the mpicc-wrapper-data.txt file to the mpixlc-wrapper-data.txt file:

```
cd ${ompi_install_dir}/share/openmpi
cp mpicc-wrapper-data.txt mpixlc-wrapper-data.txt
```

d. Edit the mpixlc-wrapper.txt file as shown in the following example:

```
diff mpicc-wrapper-data.txt mpixlc-wrapper-data.txt
14c14
< compiler=gcc

> compiler=xlc
17,18c17,18
< compiler_flags=-pthread -m64
< linker_flags= -m64

> compiler_flags=-qthreaded -q64
> linker_flags= -q64
```

Once this is done, you should be able to build your applications with `mpixlc` and it will use the IBM `xlc` compiler.

2. To create a wrapper compiler to integrate the use of `xlf` with Open MPI, complete the following steps:

- a. Run the following command:

```
ompi_install_dir=/usr/local/openmpi-1.4.2-tm/
```

- b. In the Open MPI binary directory, symbolically link `opal_wrapper` to the desired wrapper compiler name (in this case, `mpixlf`):

```
ln -s ${ompi_install_dir}/bin/opal_wrapper ${ompi_install_dir}/bin/mpixlf
```

- c. In the Open MPI `share/openmpi` directory, copy the `mpif77-wrapper-data.txt` file to the `mpixlf-wrapper-data.txt` file.

```
cd ${ompi_install_dir}/share/openmpi
cp mpif77-wrapper-data.txt mpixlf-wrapper-data.txt
```

- d. Edit the `mpixlf-wrapper.txt` file as shown in the following example:

```
diff mpif77-wrapper-data.txt mpixlf-wrapper-data.txt
14c14
< compiler=gfortran

> compiler=xlf
17,18c17,18
< compiler_flags=-pthread -m64
< linker_flags= -m64

> compiler_flags=-qthreaded -q64
> linker_flags= -q64
```

Once this is done, you should be able to build your applications with `mpixlf` and it will use the IBM `xlf` compiler.

3. To verify the compiler options that are being used, run the compiler with the `--showme` option (this step can be done as `hpc` user):

```
$ mpixlc --showme
xlc -I/usr/local/openmpi-1.4.2/include -qthreaded -q64 -q64 -L/usr/local/openmpi-1.4.2/lib -lmpi \
-llopenrte -lopenpal -ldl -Wl,--export-dynamic -lnsl -lutil -lm -ldl

$ mpixlf --showme
xlf -I/opt/openmpi/1.4.2/include -qthreaded -q64 -q64 -L/opt/openmpi/1.4.2/lib -lmpi_f77 -lmpi \
-llopen-rte -lopen-pal -ldl -Wl,--export-dynamic -lnsl -lutil -lm -ldl
```

## Results

If `-I/usr/local/${ompi_install_dir}/include` and `-L/usr/local/${ompi_install_dir}/lib` are shown in the output, then you have successfully configured Open MPI to build and run HPC application with IBM XL compilers. Note that the other wrapper compilers for `mpif77`, `mpif90`, `mpic++`, and the `_r` variants can be handled similarly.

### Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

---

## Using Open MPI with huge pages

If you have `libhugetlbfs` installed, you can choose to use 16MB huge pages with Open MPI when you start a distributed program.

## Before you begin

Ensure that Open MPI is installed and that libhugetlbfs is installed.

## About this task

The standard Open MPI configuration includes support for optimal use of RDMA-based communications, such as InfiniBand. This requires the use of special memory management routines within the Open MPI library to manage pages used in RDMA transfers. However, this is incompatible with the use of an interposer memory management library, such as libhugetlbfs.

To enable use of libhugetlbfs with Open MPI, the IBM Installation Toolkit contains a version of Open MPI which does not use the embedded Open MPI memory manager. After Open MPI is installed from the IBM Installation Toolkit, and you have your environment setup to use it, you can use libhugetlbfs as an interposer library with the LD\_PRELOAD mechanism. This enables existing software to take advantage of libhugetlbfs without recompiling or relinking.

## Procedure

1. To make use of 16MB huge pages, start the distributed application with the environment variable LD\_PRELOAD set to libhugetlbfs.so and HUGETLB\_MORECORE set to yes. The following example shows how the hpc user can start an application with these settings:

```
$ mpirun -x LD_PRELOAD=libhugetlbfs.so -x HUGETLB_MORECORE=yes \
-x OMPI_MCA_memory_ptmalloc2_disable=1 -np <#> <program>
```

**Note:** In this example, the **-x** flag of the **mpirun** command tells **mpirun** to propagate the designated environment setting to all of the processes of the distributed job.

2. Verify that huge pages are used by the application.

You can verify the use of libhugetlbfs by running a job which does dynamic memory allocation and then observing the values of HugePages\_Total and HugePages\_Free in /proc/meminfo. If the job uses huge pages, when the application runs, the value of HugePages\_Free decreases from the value it shows before you run the program.

- a. Check the values of HugePages\_Total and HugePages\_Free before you run the program, as shown in the following example:

```
egrep 'HugePages_(Total|Free)' /proc/meminfo
HugePages_Total: 400
HugePages_Free: 400
```

- b. Check the values of HugePages\_Total and HugePages\_Free while the program is running, as shown in the following example:

```
egrep 'HugePages_(Total|Free)' /proc/meminfo
HugePages_Total: 400
HugePages_Free: 80
```

3. Check the values of HugePages\_Total and HugePages\_Free when the program is has completed, as shown in the following example:

```
egrep 'HugePages_(Total|Free)' /proc/meminfo
HugePages_Total: 400
HugePages_Free: 400
```

## Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

---

## Chapter 7. Testing the InfiniBand interconnect performance with Intel MPI Benchmark

Validate the InfiniBand interconnect using a commonly used benchmark called Intel MPI Benchmark (IMB).

This section focuses on verifying the performance of the InfiniBand interconnects. This is an important step in verifying the performance of your cluster because if the interconnects cannot perform well, they can become a bottleneck for multi-node runs. You can complete the steps in each of the following sections as the hpc user.

### Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

---

## Building IMB

IMB can be run on a single node, but for the purposes of testing InfiniBand performance, you will want to use at least two nodes for the test so that the messages will be passed across the InfiniBand link and not just within the shared memory of a single system.

### About this task

Perform the following procedure on all of the nodes.

### Procedure

1. Download IMB-3.2 from the following link: <http://software.intel.com/en-us/articles/intel-mpi-benchmarks/>

2. Untar the package and change directory into the /src directory with the following commands:

```
tar zxvf IMB_3.2.tgz -C /home/hpc
cd /home/hpc/imb/src/
```

3. Edit the make\_ict makefile to change the assignment of the **CC** value from mpiic to mpicc as shown:

```
LIB_PATH =
LIBS =
CC = mpicc
ifeq (, $(shell which ${CC}))
$(error ${CC} is not defined through the PATH <clipped>)
endif
OPTFLAGS =
CLINKER = ${CC}
LDFLAGS =
CPPFLAGS =
```

```
export CC LIB_PATH LIBS OPTFLAGS CLINKER LDFLAGS CPPFLAGS
include Makefile.base
```

4. Run the **make** command. When the **make** command completes, the IMB-MPI1 executable is shown with other object files and executables.



## Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

---

## Running IMB

You can run IMB to determine whether you need to affinitize the processes to increase performance.

### Procedure

1. Run IMB\_MPI1 pingpong from management node:

```
$ cd /home/hpc
$ mpirun -np 2 -host compute_node1,compute_node2 imb/src/IMB-MPI1 pingpong
```

Verify that you see the IMB output on your screen.

2. Run IMB in a pbs job with the following steps. This example job runs three IMB-MPI1 workloads: pingpong, sendrecv, and exchange across two nodes.

**Note:** Perform this step for every InfiniBand link.

- a. Create the job.pbs file as shown:

```
#PBS -N imb_2node
#PBS -q batch
#PBS -l nodes=2,pmem=100m,walltime=00:30:00
#PBS -M admin@example.com
#PBS -m e
#PBS -V
```

```
cd /home/hpc
mpirun -np 2 --host compute_node1,compute_node2 imb/src/IMB-MPI1 pingpong
```

3. Submit the job with the **qsub** command, as shown in the following example:

```
$ qsub job.pbs
3.mgmt_node.example.com
```

### Results

Two new output files are available after running the job: one containing job outputs and another containing any errors. They are named <job name>.o<job identifier> and <job name>.e<job identifier>, respectively. These output files are located on the compute node where the job ran.

The following output is the imb\_2node.o3 file for the job that ran in the test environment, including the output of expected performance for the pingpong, sendrecv, and exchange workloads on a single InfiniBand link:

```
#-----
Intel (R) MPI Benchmark Suite V3.2, MPI-1 part
#-----
Date : Wed Aug 25 22:28:31 2010
Machine : ppc64
System : Linux
Release : 2.6.32.12-0.7-ppc64
Version : #1 SMP 2010-05-20 11:14:20 +0200
MPI Version : 2.1
MPI Thread Environment: MPI_THREAD_SINGLE
```

```
New default behavior from Version 3.2 on:
```

```
the number of iterations per message size is cut down
```



```
dynamically when a certain run time (per message size sample)
is expected to be exceeded. Time limit is defined by variable
"SECS_PER_SAMPLE" (=> IMB_settings.h)
or through the flag => -time
```

```
Calling sequence was:
```

```
IMB-MPI1 pingpong sendrecv exchange
```

```
Minimum message length in bytes: 0
Maximum message length in bytes: 4194304
#
MPI_Datatype : MPI_BYTE
MPI_Datatype for reductions : MPI_FLOAT
MPI_Op : MPI_SUM
#
#
```

```
List of Benchmarks to run:
```

```
PingPong
Sendrecv
Exchange
```

```
#-----
Benchmarking PingPong
#processes = 2
#-----
```

| #bytes  | #repetitions | t[usec] | Mbytes/sec |
|---------|--------------|---------|------------|
| 0       | 1000         | 3.42    | 0.00       |
| 1       | 1000         | 3.61    | 0.26       |
| 2       | 1000         | 3.63    | 0.53       |
| 4       | 1000         | 3.58    | 1.06       |
| 8       | 1000         | 3.37    | 2.26       |
| 16      | 1000         | 3.28    | 4.65       |
| 32      | 1000         | 3.33    | 9.15       |
| 64      | 1000         | 3.37    | 18.11      |
| 128     | 1000         | 3.51    | 34.75      |
| 256     | 1000         | 3.86    | 63.23      |
| 512     | 1000         | 4.54    | 107.44     |
| 1024    | 1000         | 5.83    | 167.51     |
| 2048    | 1000         | 7.93    | 246.42     |
| 4096    | 1000         | 12.39   | 315.15     |
| 8192    | 1000         | 16.83   | 464.20     |
| 16384   | 1000         | 18.94   | 825.08     |
| 32768   | 1000         | 27.44   | 1138.76    |
| 65536   | 640          | 44.30   | 1410.69    |
| 131072  | 320          | 78.01   | 1602.41    |
| 262144  | 160          | 145.32  | 1720.40    |
| 524288  | 80           | 280.61  | 1781.86    |
| 1048576 | 40           | 549.60  | 1819.50    |
| 2097152 | 20           | 1088.37 | 1837.60    |
| 4194304 | 10           | 2162.96 | 1849.32    |

```
#-----
Benchmarking Sendrecv
#processes = 2
#-----
```

| #bytes | #repetitions | t_min[usec] | t_max[usec] | t_avg[usec] | Mbytes/sec |
|--------|--------------|-------------|-------------|-------------|------------|
| 0      | 1000         | 3.23        | 3.23        | 3.23        | 0.00       |
| 1      | 1000         | 3.33        | 3.33        | 3.33        | 0.57       |
| 2      | 1000         | 3.39        | 3.39        | 3.39        | 1.13       |
| 4      | 1000         | 3.33        | 3.33        | 3.33        | 2.29       |
| 8      | 1000         | 3.34        | 3.35        | 3.34        | 4.56       |
| 16     | 1000         | 3.37        | 3.38        | 3.37        | 9.04       |

|         |      |         |         |         |         |
|---------|------|---------|---------|---------|---------|
| 32      | 1000 | 3.39    | 3.39    | 3.39    | 18.01   |
| 64      | 1000 | 3.47    | 3.48    | 3.48    | 35.09   |
| 128     | 1000 | 3.66    | 3.66    | 3.66    | 66.71   |
| 256     | 1000 | 4.05    | 4.05    | 4.05    | 120.50  |
| 512     | 1000 | 4.90    | 4.91    | 4.90    | 199.01  |
| 1024    | 1000 | 6.07    | 6.07    | 6.07    | 321.81  |
| 2048    | 1000 | 8.26    | 8.26    | 8.26    | 473.14  |
| 4096    | 1000 | 12.46   | 12.46   | 12.46   | 627.01  |
| 8192    | 1000 | 17.12   | 17.13   | 17.13   | 912.20  |
| 16384   | 1000 | 23.07   | 23.07   | 23.07   | 1354.69 |
| 32768   | 1000 | 33.05   | 33.05   | 33.05   | 1891.07 |
| 65536   | 640  | 52.24   | 52.24   | 52.24   | 2392.58 |
| 131072  | 320  | 90.99   | 90.99   | 90.99   | 2747.44 |
| 262144  | 160  | 168.42  | 168.44  | 168.43  | 2968.34 |
| 524288  | 80   | 323.80  | 323.84  | 323.82  | 3087.96 |
| 1048576 | 40   | 637.67  | 637.78  | 637.72  | 3135.90 |
| 2097152 | 20   | 1261.31 | 1261.56 | 1261.43 | 3170.69 |
| 4194304 | 10   | 2502.89 | 2502.92 | 2502.91 | 3196.27 |

```
#-----
Benchmarking Exchange
#processes = 2
#-----
#bytes #repetitions t_min[usec] t_max[usec] t_avg[usec] Mbytes/sec
0 1000 4.62 4.62 4.62 0.00
1 1000 4.66 4.66 4.66 0.82
2 1000 4.67 4.68 4.67 1.63
4 1000 4.67 4.67 4.67 3.27
8 1000 4.67 4.67 4.67 6.53
16 1000 4.61 4.62 4.61 13.23
32 1000 4.77 4.77 4.77 25.59
64 1000 4.78 4.79 4.78 51.02
128 1000 5.38 5.38 5.38 90.78
256 1000 5.81 5.81 5.81 168.05
512 1000 6.42 6.42 6.42 304.37
1024 1000 8.03 8.03 8.03 486.39
2048 1000 10.59 10.59 10.59 737.44
4096 1000 16.51 16.52 16.52 945.71
8192 1000 23.46 23.47 23.47 1331.55
16384 1000 44.33 44.33 44.33 1409.82
32768 1000 64.44 64.44 64.44 1939.73
65536 640 104.86 104.86 104.86 2384.21
131072 320 182.61 182.61 182.61 2738.03
262144 160 338.96 338.98 338.97 2950.01
524288 80 650.80 650.80 650.80 3073.13
1048576 40 1272.23 1272.23 1272.23 3144.10
2097152 20 2518.40 2518.70 2518.55 3176.24
4194304 10 5003.00 5003.31 5003.15 3197.88
```

```
All processes entering MPI_Finalize
```

### Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

---

## Chapter 8. Related information

You can find additional information about the processes and tools described in these procedures.

IBM Installation

Toolkit for PowerLinux

<http://www14.software.ibm.com/webapp/set2/sas/f/lopdiags/installtools>

Open MPI

<http://www.open-mpi.org/>

IBM Advance

Toolchain for PowerLinux

<ftp://linuxpatch.ncsa.uiuc.edu/toolchain>

TORQUE Resource Manager

<http://www.clusterresources.com/torquedocs21/>

Maui Cluster Scheduler

<http://www.clusterresources.com/products/maui-cluster-scheduler.php>

IBM XL C/C++ and XL Fortran Compilers

<http://www.ibm.com/software/awdtools/xlcpp/linux/>

<http://www.ibm.com/software/awdtools/fortran/xlfortran/linux/>

libhugetlbfs

<http://sourceforge.net/projects/libhugetlbfs/>

ATLAS math libraries

<http://sourceforge.net/projects/math-atlas/>

### **Related concepts:**

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.



---

## Chapter 9. Setting up an HPC cluster with Red Hat Enterprise Linux

You can set up and manage a cluster of compute nodes using common open-source based cluster and resource management tools on POWER-based systems running Red Hat Enterprise Linux 5.5. This demonstration has examples of how to use the Open MPI, Torque, and Maui open source software to define and manage a small set of compute servers interconnected with InfiniBand in a classic High Performance Computing (HPC) cluster.

Key tools and technologies detailed in this demonstration include ATLAS, TORQUE, Maui, and Open MPI.

---

### Scope, requirements, and support

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

### Systems to which this information applies

PowerLinux (POWER6 and later)

### Intended audience

This blueprint is intended for advanced Linux system administrators and programmers who are already familiar with basic HPC concepts, terminology, products, including InfiniBand<sup>®</sup> interconnects. The instructions assume that you have a general understanding of the usage requirements for connecting nodes with InfiniBand and with MPI-based applications.

### Scope and purpose

This document is based in part on the collaborative experiences of the IBM Linux Technology Center, IBM's systems development teams, and Rice University in deploying and managing the POWER7-based BlueBioU cluster, at <http://bluebiou.rice.edu/>.

This document discusses the following software that you can use to deploy a simple cluster with POWER-based systems:

- Basic InfiniBand setup
- Open MPI
- Torque
- IBM Advance Toolchain for PowerLinux
- libhugetlbfs

The following topics are not covered:

- Advanced cluster management products, such as:
  - Moab (Cluster Resources)
  - IBM Parallel Environment
  - Extreme Cloud Administration Toolkit (xCAT)
- Detailed problem determination
- In-depth technology discussion

Resources for in-depth discussion of these and other topics are listed in Related Information

## Test environment

The instructions in this blueprint were tested on IBM Power 575 systems with POWER6 processors running Red Hat Enterprise Linux 5.5. The BlueBioU team at Rice University has deployed a similar software stack with POWER7-based IBM Power 750 systems.

## Hardware, software, and other prerequisites

In testing, one server was used as a “management node” and two other servers were the “compute nodes”. Each server had an InfiniBand adapter supporting physical connections to a shared InfiniBand switch.

The examples in this demonstration are focused on eHCA with a single port.

The following software resources are used to perform the steps in this blueprint:

- A Red Hat Enterprise Linux 5.5 YUM repository
- The IBM Installation Toolkit for Linux ISO image from <http://www14.software.ibm.com/webapp/set2/sas/f/lopdiags/installtools/download/>
- Torque Resource Manager software from <http://www.clusterresources.com/torquedocs21/1.1installation.shtml>
- Maui Cluster Scheduler software from <http://www.adaptivecomputing.com/support-old/download-center/maui-cluster-scheduler>

This blueprint focuses on Red Hat Enterprise Linux 5.5, but you can apply the concepts easily to other Linux distributions that are supported on POWER-based systems.

The Red Hat createrepo RPM package must be installed on all nodes of the cluster.

The Red Hat Enterprise Linux 5.5 YUM repository must be available on each of the nodes.

## Author names

Brad Benton  
Bill Buros  
Ric Hendrickson  
Jenifer Hopper


## Other contributors

Heather Crognale  
Jeff George  
Monza Lui

## IBM Services

Linux offers flexibility, options, and competitive total cost of ownership with a world class enterprise operating system. Community innovation integrates leading-edge technologies and best practices into Linux.

IBM is a leader in the Linux community with over 600 developers in the IBM Linux Technology Center working on over 100 open source projects in the community. IBM supports Linux on all IBM servers, storage, and middleware, offering the broadest flexibility to match your business needs.

For more information about IBM and Linux, go to [ibm.com/linux](https://www.ibm.com/linux)  (<https://www.ibm.com/linux>)

## IBM Support

Questions and comments regarding this documentation can be posted on the developerWorks HPC

Central Technical Forum: <http://www.ibm.com/developerworks/forums/forum.jspa?forumID=1056> 

The IBM developerWorks discussion forums let you ask questions, share knowledge, ideas, and opinions about technologies and programming techniques with other developerWorks users. Use the forum content at your own risk. While IBM attempts to provide a timely response to all postings, the use of this developerWorks forum does not guarantee a response to every question that is posted, nor do we validate the answers or the code that are offered.

You can also find additional information in the following forums:

- The developerWorks Linux for Power Architecture forum: <http://www.ibm.com/developerworks/forums/forum.jspa?forumID=375>
- The developerWorks IBM Installation Toolkit for PowerLinux on POWER Support Forum: <http://www.ibm.com/developerworks/forums/forum.jspa?forumID=937>

---

## Open source-based HPC compute cluster on POWER

You can learn about basic concepts of HPC clusters, and get a brief overview of some of the open source software you can use to build an HPC cluster on POWER processor-based systems.

### HPC compute clusters

This blueprint is focused on the class of servers which are used primarily for compute-intensive work like computational simulations of weather, bioinformatics, and so on. In this class of compute-intensive workloads, a single compute job may execute on multiple servers simultaneously and often requires extensive communication between each of the servers. In these server clusters, the servers typically share a dedicated high-speed network (often based on InfiniBand), are densely and closely located relative to each other, and are homogeneous.

### Management node

A *management node*, typically a separate server, is used for running the cluster's scheduler and resource managers. This blueprint shows how to install a Maui Cluster Scheduler and the Torque Resource Manager which collectively manage and control the compute nodes.

### Compute node

With a large number of compute nodes, there is usually a requirement for schedulers and server resource managers to prioritize and manage which jobs can be executed on the cluster. In this blueprint, we focus on just a handful of compute nodes, with the notion that the compute node setup is a repeatable incremental process. As you need to increase the number of compute nodes in your cluster, you can refer to these examples to ensure that the additional nodes are properly prepared.

### InfiniBand interconnect

The example cluster in this blueprint uses InfiniBand to connect the nodes of the cluster. InfiniBand is a scalable, low-latency, high performance communication mechanism frequently used in HPC environments.

## **The software components**

These are the essential components of the HPC software stack, including compilers, math libraries, communication stack, cluster management and scheduling software, and software used for installation of key software components for POWER processor-based systems.

### **IBM Installation Toolkit for PowerLinux**

The IBM Installation Toolkit for PowerLinux provides a set of tools that simplifies the installation of IBM value-added software necessary to fully use the capabilities of the Power platform. In particular, for this cluster we are interested in the two 64-bit Open MPI packages included in the IBM Installation Toolkit Version 4.1.

### **InfiniBand software stack**

The InfiniBand software stack provides cluster-knowledgeable middleware, such as MPI libraries, access to the low-latency, high bandwidth InfiniBand fabric for cluster communications. Red Hat Enterprise Linux provides an InfiniBand software stack.

### **Open MPI**

The Open MPI project provides an open source implementation of the MPI-2 standard. It is developed and maintained by a consortium of academic, research, and industry partners. Open MPI is therefore able to combine the expertise, technologies, and resources from across the High Performance Computing community in order to build reliable and high performance MPI library.

### **IBM Advance Toolchain for PowerLinux**

The IBM Advance Toolchain for PowerLinux is a collection of free software application development products specifically tailored to make use of IBM POWER processor features. The package includes GCC (GNU Compiler Collection), GLIBC (GNU C Libraries), GNU binary utilities, GDB (GNU debugger), and the performance analysis tools (QProfile and Valgrind) in a self contained Toolchain. The IBM Advance Toolchain for PowerLinux duplicates the function of packages already delivered by Linux distributors but provides additional features for the POWER6 and POWER7 processors as indicated in the IBM Advance Toolchain for PowerLinux Release Notes 'Features' section. The IBM Advance Toolchain for PowerLinux is built and distributed by the University of Illinois.

### **TORQUE resource manager**

Tera-scale Open-source Resource and QUEue Manager (TORQUE) is a workload and resource-management system that manages batch jobs and compute nodes and schedules the execution of those jobs. The TORQUE system includes three pieces: the server, the scheduler, and the job executor. The server process, called `pbs_server`, creates and manages jobs and queues and dispatches jobs for execution to compute nodes, where the job executor, called `pbs_mom`, starts the job.

### **Maui Cluster Scheduler**

The Maui Cluster Scheduler is a policy engine used to improve the manageability and efficiency of various sized clusters, from small clusters with just a few processors to very large supercomputers. The Maui Cluster Scheduler is intended to support a large array of scheduling policies, dynamic priorities, extensive reservations, and fairshare.

### **IBM XL C/C++ and XL Fortran compilers**

XL C/C++ for Linux V11.1 is a highly advanced optimizing compiler for the selected Linux distributions. The compiler runs on IBM Power Systems and is an industry standards-based professional programming



tool that can be used for developing large, complex, computationally intensive 32-bit and 64-bit applications in the C and C++ programming languages.

XL Fortran is a standards-based Fortran compiler with extensive features to help you create high performance applications. A highly advanced optimizing compiler for the Linux operating system, runs on IBM Power Systems.

## libhugetlbfs

The libhugetlbfs library interacts with the Linux hugetlbfs to make large pages available to applications in a transparent manner. On POWER processor-based systems, this library provides applications easy access to the 16 MB huge pages.

## ATLAS math libraries

Automatically Tuned Linear Algebra Software (ATLAS) provides highly optimized Linear Algebra kernels for arbitrary cache-based architectures. ATLAS provides ANSI C and Fortran77 interfaces for the entire BLAS API, and a small portion of the LAPACK AP.

### Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

---

## Setting up the cluster

You can see an overview of the cluster component setup for this example, including whether components are set up on the management node or compute nodes, and which user does the installation or configuration tasks.

The cluster is dependent on each of the compute nodes being connected with InfiniBand. Although it is common that all nodes have an InfiniBand connection, it is not required for the the management node. However, the InfiniBand packages should still be installed on the management node so that other package builds can use the InfiniBand components. Therefore, we start by installing the latest Open MPI and InfiniBand packages on all nodes - the management node and all compute nodes.

The installation and configuration tasks are performed by the root user except where noted in the following tables.

*Table 3. Cluster setup tasks*

| Cluster setup                              |                 |              |
|--------------------------------------------|-----------------|--------------|
| Task                                       | Management node | Compute node |
| Installing OpenMPI and InfiniBand packages | Yes             | Yes          |
| Configuring InfiniBand                     | Not required    | Yes          |
| Installing Torque Resource Manager         | Yes             | No           |
| Installing Maui Cluster Scheduler          | Yes             | No           |
| Configuring Torque                         | Yes             | Yes          |

*Table 4. Advanced software setup tasks*

| Advanced software setup |                 |              |
|-------------------------|-----------------|--------------|
| Task                    | Management node | Compute node |

Table 4. Advanced software setup tasks (continued)

| Advanced software setup                                                        |                                                                  |                                                                  |
|--------------------------------------------------------------------------------|------------------------------------------------------------------|------------------------------------------------------------------|
| Installing libhugetlbfs                                                        | Yes                                                              | Yes                                                              |
| Installing IBM Advance Toolchain for PowerLinux                                | Yes                                                              | Yes                                                              |
| Installing the IBM XL C/C++ and Fortran compilers                              | Yes                                                              | Yes                                                              |
| Installing ATLAS math libraries                                                | Yes                                                              | Yes                                                              |
| Building Open MPI with Torque                                                  | Yes                                                              | Yes                                                              |
| Configuring Open MPI for use with the Advance Toolchain                        | Yes, as hpc user                                                 | Yes, as hpc user                                                 |
| Configuring Open MPI for use with the IBM XL compilers                         | Yes, as root when installing and then as hpc user when verifying | Yes, as root when installing and then as hpc user when verifying |
| Using Open MPI with huge pages                                                 | Yes, as hpc user                                                 | Yes, as hpc user                                                 |
| Testing the InfiniBand interconnect performance with Intel MPI Benchmark (IMB) | Yes, as hpc user when installing and submitting jobs             | Yes, as hpc user when installing                                 |

#### Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

## Setting up the hpc user

The hpc user configures and uses various optional software components on the cluster, including Open MPI, the IBM XL compilers, and the Intel MPI Benchmark (IMB).

### About this task

You must ensure that the hpc user exists on all the nodes of the cluster in order to perform some of the configuration and testing tasks described in this blueprint.

The hpc user needs password-less connectivity among all the nodes in order to enable Open MPI applications to pass messages between nodes. Without password-less connectivity, you would be required to enter the password for each node whenever a job was run. The stage needs to be set just so.

### Procedure

1. Create the hpc user on any nodes where it does not already exist. If the hpc user does not exist on a node, you can create the hpc user with the following commands:

```
useradd -m hpc
passwd hpc
```

2. Set up password-less connectivity for the hpc user among all the nodes in the cluster:

On the management node and all compute nodes, check if hpc user has password-less connectivity. You must be able to use ssh to connect to and from all the nodes without a password request. (There are several methods available to set up password-less connectivity, outside the scope of this document.)

When you have successfully configured ssh for the hpc user, you will be able to use ssh to connect between any two nodes in the cluster without a password. The following example shows a password-less ssh connection from the host named `compute_node1` to the host named `compute_node2`:

```
[compute_node1] # su - hpc
[compute_node1] $ ssh compute_node2
```

```
Warning: Permanently added 'compute_node2' (RSA) to the list of known hosts.
[compute_node2] $
```

### Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

## Installing Open MPI and InfiniBand packages

You can obtain and use the IBM Installation Toolkit for PowerLinux to install the Open MPI and InfiniBand software packages.

### Before you begin

1. Ensure that the createrepo RPM package is installed on each node of the cluster.
2. Download the IBM Installation Toolkit media (ISO image) from <http://www14.software.ibm.com/webapp/set2/sas/f/lopdiags/installtools/download/>

#### Note:

- You must register and sign in before you can download the software.
- As of the time of writing of this blueprint, the current IBM Installation Toolkit version is 4.1.

### About this task

In these examples we assume the operating system (Red Hat Enterprise Linux 5.5) has already been installed on all of the servers – in this case the management node and the two compute nodes.

**Note:** The IBM Installation Toolkit can be used to install the Linux distribution for servers as well, but installing the operating system is outside the scope of this discussion.

The root user performs all the steps of this procedure.

Perform these steps for each node of the cluster:

### Procedure

1. Create a local IBM Installation Toolkit YUM repository:
  - a. Mount the IBM Installation Toolkit ISO media and copy the Red Hat directory with all of the Red Hat-related RPM packages to a directory of your choice. The following example uses /opt/ibmit41 as the destination directory:

```
mkdir /mnt/ibmtoolkit
mkdir /opt/ibmit41
mount -o loop -t iso9660 IBM_Installation_Toolkit_41.iso /mnt/ibmtoolkit
cp -aR /mnt/ibmtoolkit/* /opt/ibmit41
umount /mnt/ibmtoolkit
rmdir /mnt/ibmtoolkit
```

**Note:** This example, uses the **cp** command's **-L** flag order to dereference and copy files that are symbolic links in the /RedHat directory on the IBM Installation Toolkit ISO image.

- b. Use the **createrepo** command to create the repository, as in the following example:

```
createrepo /opt/ibmit41/RedHat
33/33 - RPMS/xconfig-0.1-1.ppc.rpm
Saving Primary metadata
Saving file lists metadata
Saving other metadata
```

- c. Create the definition file for this new repository.

The `/etc/yum.repos.d` directory is the default YUM repository location. Therefore, in this example, the definition file is created as `/etc/yum.repos.d/ibmit41.repo`. The content of the definition file should look like the following example.

**Note:** The `baseurl` parameter is set to point to the local directory where the `createrepo` command was issued against in the previous step and where the Red Hat RPM packages reside.

```
[ibmit41]
name=ibmit41
baseurl=file:///opt/ibmit41/RedHat
enabled=1
gpgcheck=0
```

- d. Verify that the local repository is set up correctly, as in the following example:

```
yum repolist
Loaded plugins: rhnplugin, security
This system is not registered with RHN.
RHN support will be disabled.
```

| repo id | repo name | status         |
|---------|-----------|----------------|
| core-0  | core-0    | enabled: 32    |
| core-1  | core-1    | enabled: 33    |
| core-2  | core-2    | enabled: 3,089 |
| ibmit41 | ibmit41   | enabled: 33    |

```
repolist: 3,187
```

2. Install the Open MPI and InfiniBand-related packages from the local IBM Installation Toolkit YUM repository and the Red Hat Enterprise Linux 5.5 YUM repository.

```
yum install sysstat \
 openmpi-1.4.2-1.ppc64 \
 openib mpi-selector infiniband-diags perftest ofed-docs \
 libibverbs* librdmacm* libehca*
```

**Note:**

- "sysstat" is a package for extended performance tools on a Power system.
- The Open MPI package is provided on the IBM Installation Toolkit. This is a newer package of Open MPI built for 64-bit applications which has been packaged and provided for customers.
- The remainder of the packages are standard packages necessary for MPI applications using InfiniBand connectivity.
- "ofed" is from the OpenFabrics Alliance – <https://www.openfabrics.org/index.php>. The OpenFabrics Enterprise Distribution (OFED) is a standardized set of enabling software generally targeted at high-performance low-latency computing.

3. Configure the `PATH` and `LD_LIBRARY_PATH` environment variables for the `hpc` user to include the paths for Open MPI and software installed in `/usr/local/sbin` for this HPC cluster configuration by modifying the `/home/hpc/.bashrc` file as follows:
- Add the `/usr/local/sbin` directory and the `/opt/openmpi/1.4.2/bin` directory to the beginning of the `PATH` environment variable.
  - Add the `/opt/openmpi/1.4.2/lib` directory to the beginning of the `LD_LIBRARY_PATH` environment variable.

The following example shows a simple method to append these modifications to the environment variables in the `.bashrc` file for the `hpc` user:

```
cat >> /home/hpc/.bashrc <<EOF
export PATH=/usr/local/sbin:/opt/openmpi/1.4.2/bin:$PATH
export LD_LIBRARY_PATH=/opt/openmpi/1.4.2/lib:$LD_LIBRARY_PATH
EOF
```

## Results

After you have completed this installation process on each node of the cluster, every system will have the required Open MPI and InfiniBand software installed.

## What to do next

You are now ready to configure and validate the InfiniBand connectivity between all the InfiniBand-connected nodes.

### Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

## Configuring InfiniBand

You can learn about the steps needed to configure and confirm InfiniBand connectivity.

## Before you begin

The Open MPI and InfiniBand packages must already be installed on all nodes.

The allowable amount of locked (pinned) memory needs to be set to unlimited. To do this, edit `/etc/security/limits.conf` and add the following lines:

```
#<domain> <type> <item> <value>
#
* soft memlock unlimited
* hard memlock unlimited
```

## About this task

These steps assume your servers have InfiniBand adapters installed and are connected to an InfiniBand switch. In this example, all nodes are assumed to be connected to the same InfiniBand fabric.

This procedure is required for all nodes that will have InfiniBand traffic.

**Note:** You will only perform this process on the management node if the management node has an InfiniBand connection. The root user performs all the steps of this procedure.

## Procedure

1. Specify the `nr_ports` and `port_act_time` options for the IBM InfiniBand eHCA kernel module, `ib_ehca`, by creating the `/etc/modprobe.d/ib_ehca.conf` file containing the following line:
 

```
options ib_ehca nr_ports=-1 port_act_time=120
```

### Note:

- The `nr_ports` option is set to -1. This causes the module to autodetect the number of active ports.
- `port_act_time` option is set to 120. This causes the module to wait for 120 seconds for port activation.

2. Edit `/etc/ofed/openib.conf` and change the values of **SDP\_LOAD**, **RDS\_LOAD**, **SRP\_LOAD**, and **ISER\_LOAD** to `no`. This changes the default setting to load only the drivers that are needed for this setup. The resulting file will be similar to the following example:

```
Load IPoIB
IPOIB_LOAD=yes
Load SDP module
SDP_LOAD=no
Load the RDS protocol driver
RDS_LOAD=no
Load SRP module
SRP_LOAD=no
Load iSER module
ISER_LOAD=no
Should we modify the system mtrr registers? We may need to do this if you
get messages from the ib_ipath driver saying that it couldn't enable
write combining for the PIO buffs on the card.
FIXUP_MTRR_REGS=no
```

3. Configure the `openibd` service:

- a. Run `chkconfig` to configure the `openibd` service to start automatically whenever the operating system starts, as shown in the following example:

```
chkconfig openibd on
chkconfig --list openibd
openibd 0:off 1:off 2:on 3:on 4:on 5:on 6:off
```

**Note:** The output from `chkconfig --list openibd` in this example shows whether the `openibd` service will start for each runlevel.

- b. Run the **kudzu** command to detect the InfiniBand hardware.

**Note:** The `/etc/sysconfig/hwconf` file will also be created which is referenced by the `openibd` service.

- c. Start the `openibd` service as shown in the following example:

```
service openibd start
Loading OpenIB kernel modules: [OK]
```

4. Validate InfiniBand device configuration by displaying InfiniBand devices on the node:

- a. To list InfiniBand devices on the node, run the **ibv\_devices** command. This step validates that the previous steps have been completed and the InfiniBand adapters are correctly detected. If your InfiniBand devices are correctly configured, the output of the **ibv\_devices** command will look similar to the following example:

```
ibv_devices
device node GUID

ehca0 0002550010e56a00
ehca1 0002550010e56c00
```

- b. Use `ibv_devinfo` to query the InfiniBand devices. Verify that the port status is active and that `sm_lid` is not 0 for all connected ports. The following example shows typical output from the `ibv_devinfo` command:

```
ibv_devinfo

hca_id: ehca0
node_guid: 0002:5500:7018:9600
sys_image_guid: 0000:0000:0000:0000
vendor_id: 0x5076
vendor_part_id: 1
hw_ver: 0x2000021
phys_port_cnt: 2
 port: 1
 state: PORT_ACTIVE (4)
 max_mtu: 4096 (5)
```

```

 active_mtu: 4096 (5)
 sm_lid: 1
 port_lid: 29
 port_lmc: 0x00
max_mtu:
port: 2
 state: PORT_ACTIVE (4)
 active_mtu: 4096 (5)
 sm_lid: 1
 port_lid: 41
 port_lmc: 0x00

```

#### Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

## Testing InfiniBand connectivity

This series of tests assures connectivity, reasonable bandwidth, and latency performance between a pair of InfiniBand-connected nodes.

### Before you begin

Configure the InfiniBand devices on all nodes, as described in “Configuring InfiniBand” on page 12.

### About this task

Test the connectivity for each node with at least one other node in the cluster before you continue setting up the cluster.

These tests run between two nodes, one node acting as the server and the other node acting as the client for purposes of the test. The server node initiates a listening server and waits to receive requests from the client. The client node sends the request to the specified server.

The root user performs all the steps of this procedure.

### Procedure

1. Test InfiniBand connectivity with the **ibv\_rc\_pingpong** command:

- a. Run the **ibv\_rc\_pingpong** command on the server node, as shown in the following example:

```
compute_node1 # ibv_rc_pingpong
local address: LID 0x001c, QPN 0x000033, PSN 0x150335
```

- b. Run the **ibv\_rc\_pingpong** command on the client node with the hostname of the server node as the parameter. In the following example, the hostname of the server node is `compute_node1`:

```
compute_node2 # ibv_rc_pingpong compute_node1

local address: LID 0x0016, QPN 0x000075, PSN 0x1fa58b
remote address: LID 0x001c, QPN 0x000033, PSN 0x150335
8192000 bytes in 0.02 seconds = 3601.08 Mbit/sec
1000 iters in 0.02 seconds = 18.20 usec/iter
```

If the connection is successful, the **ibv\_rc\_pingpong** process on the server will produce output indicating the connection from the client, and then exit. The following example shows output on the server that results from a successful connection:

```
remote address: LID 0x0016, QPN 0x000075, PSN 0x1fa58b
8192000 bytes in 0.02 seconds = 3584.34 Mbit/sec
1000 iters in 0.02 seconds = 18.28 usec/iter
```



2. Test the bidirectional InfiniBand bandwidth with the **rdma\_bw** command:
  - a. On the server node, run **rdma\_bw -b** as shown in the following example:

**Note:** The **-b** flag specifies bidirectional message sending.

```
rdma_bw -b
```

- b. On the client node, run **rdma\_bw -b** with the hostname of the server as the parameter. In the following example, the hostname of the listening server is `compute_node1`:

```
rdma_bw -b compute_node1
```

```
15151: | port=18515 | ib_port=1 | size=65536 | tx_depth=100 | sl=0 | iters=1000 | \
duplex=1 | cma=0 |
15151: Local address: LID 0x12, QPN 0x4377, PSN 0xb05218 RKey 0x679129 VAddr 0x00040000340000
15151: Remote address: LID 0x11, QPN 0x447b, PSN 0x8b85f6, RKey 0xa81105 VAddr 0x000400001a0000
```

```
Warning: measured timestamp frequency 511.951 differs from nominal 4704 MHz
```

```
15151: Bandwidth peak (#0 to #786): 3410.28 MB/sec
15151: Bandwidth average: 3408.49 MB/sec
15151: Service Demand peak (#0 to #786): 146 cycles/KB
15151: Service Demand Avg : 146 cycles/KB
```

The important item to check in this output is the Bandwidth average: value. Throughput greater than 3000 MB/sec is desirable. A bandwidth average value of less than 3000 MB/sec may indicate a performance issue.

**Note:** The other output and warnings can be safely ignored for the purposes of this test.

3. Test bidirectional InfiniBand latency with the **rdma\_lat** command:

- a. On the server node, run the **rdma\_lat** command.

```
rdma_lat
```

- b. On the client node, run the **rdma\_lat** command with hostname of the server as the parameter. The hostname of the listening server in the following example is `compute_node1`:

```
rdma_lat compute_node1
local address: LID 0x12 QPN 0x43b2 PSN 0xb766cf RKey 0x67912e VAddr 0x00000010030001
remote address: LID 0x11 QPN 0x46f6 PSN 0xaa5ce6 RKey 0xa8110b VAddr 0x00000010030001
```

```
Warning: measured timestamp frequency 511.938 differs from nominal 4704 MHz
```

```
Latency typical: 3.61665 usec
Latency best : 3.22695 usec
Latency worst : 28.9488 usec
```

The important item to check in this output is the Latency typical: value. A Latency typical value of less than 4.0 usec is desirable.

**Note:** The other output and warnings can be safely ignored.

### Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

---

## Installing the Management Node

Install the TORQUE resource manager and the Maui Cluster Scheduler on the management node in order to control and track jobs on the compute nodes. Build special torque client and Machine Oriented Mini-server (MOM) packages to install on the compute nodes after the you install the Torque and Maui software on the management node.



## Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

## Installing Torque resource manager

You install the TORQUE resource manager only on the management node. These instructions show how to obtain, build and install the newest version of the TORQUE software.

### Before you begin

- Verify that the InfiniBand configuration is complete for all nodes by completing the tests in “Testing InfiniBand connectivity” on page 14.
- Ensure that the hpc user and the /home/hpc directory is set up on all nodes as described in “Setting up the hpc user” on page 10.

### About this task

The root user on the management node performs all the steps of this procedure.

### Procedure

1. As root user on the management node, download the newest version of TORQUE resource manager from: <http://www.clusterresources.com/downloads/torque>  
In this example, torque.2.4.8 is the latest available package.

2. Extract, configure, and build the torque software package in the /usr/local/src directory. The following example shows this process and the appropriate flags to use when you configure for the 64-bit Power Architecture:

```
tar xzf torque-2.4.8.tar.gz -C /usr/local
cd /usr/local/torque-2.4.8

./configure --enable-docs --with-scp --enable-syslog CFLAGS=-m64 --libdir=/usr/local/lib64
make
make install
ldconfig
```

3. Initialize the torque server by running the setup script and indicating the root user:

```
./torque.setup root
initializing TORQUE (admin: root@compute_node1.example.com)
Max open servers: 4
Max open servers: 4
```

4. Verify that the /var/spool/torque/server\_name file contains the fully qualified host name of the management node.

#### Note:

- a. A short host name will also work if all the nodes in the cluster share the same name server. This should be created during the initialization step.
- b. The /var/spool/torque/server\_name file was created by the **torque.setup** command.

```
cat /var/spool/torque/server_name
mgmt_node.example.com
```

5. Build the Torque self-extracting packages for compute nodes. The following example shows the root user running make packages:  

```
make packages
```
6. Copy the self-extracting packages you built for the compute nodes into the /home/hpc/comput-node-files directory, as shown in the following example:

**Note:** You must create the `/home/hpc/compute-node-files` directory if it doesn't already exist.

```
mkdir /home/hpc/compute-node-files
cp *.sh /home/hpc/compute-node-files/
```

7. Run the **libtool** command as shown in the following example to configure the libraries for the management node.

```
libtool --finish /usr/local/lib64
```

## Results

You will use these packages in “Preparing the TORQUE packages for the compute nodes” on page 20.

### Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

## Installing Maui cluster scheduler

The Maui server controls job scheduling across the cluster. These instructions show how to obtain, build, and install the newest version of the Maui cluster scheduler on the management node of the cluster.

### Before you begin

Ensure that the TORQUE server is installed on the management node, as described in Installing the TORQUE resource manager.

### About this task

The root user on the management node performs all the steps of this procedure.

### Procedure

1. Download Maui from the following site: <http://www.clusterresources.com/product/maui/index.php>

**Note:** you must register in order to download the software

For our example, we selected the Maui 3.3 version, the newest version as of the writing of this document.

2. Extract, configure, and build the Maui server software using the following commands:

```
tar -zxf maui-3.3.tar.gz -C /usr/local/src
cd /usr/local/src/maui-3.3
chmod +x /usr/local/torque-2.4.8/pbs-config
./configure --with-pbs=/usr/local CFLAGS=-m64 LDFLAGS=-m64
make
make install
```

3. Add the `/usr/local/maui/bin` directory to the **PATH** environment variable for the hpc user by appending the contents of the `/usr/local/src/maui-3.3/etc/maui.sh` file to the `/home/hpc/.bashrc` file.

```
cat /usr/local/src/maui-3.3/etc/maui.sh >> /home/hpc/.bashrc
cp /usr/local/src/maui-3.3/etc/maui.sh /home/hpc/compute-node-files/
```

4. Set up the Maui daemon as a service on the management node server:

- a. Edit the `/usr/local/src/maui-3.3/etc/maui.d` file and change the value of the **MAUI\_PREFIX** variable to `/usr/local/maui` as shown in the following example:

```
MAUI_PREFIX=/usr/local/maui
```

- b. Copy the `/usr/local/src/maui-3.3/etc/maui.d` file to `/etc/init.d/maui.d`, as shown in the following example:
 

```
cp /usr/local/src/maui-3.3/etc/maui.d /etc/init.d/maui.d
```
5. Edit the `/usr/local/maui/maui.cfg` file, which was created by the build process, to include the information needed for the environment of your hpc cluster:
  - a. Make sure the following lines are included as follows:
 

```
SERVERPORT 42599
ADMIN1 root hpc
ADMIN3 ALL
```
  - b. Modify the `SERVERHOST` and `ADMINHOST` lines to contain the fully qualified host name for the management node. For example, if the fully qualified host name of your management node is `mgt_node.example.com`, edit these lines as shown in the following example:
 

```
SERVERHOST mgt_node.example.com
ADMINHOST mgt_node.example.com
```
  - c. Verify that the `RMCFG` line contains the short host name in brackets, as shown in the following example:
 

```
RMCFG[mgt_node] TYPE=PBS
```
6. Create the `/var/spool/torque/server_priv/nodes` file to contain a list of all of the compute nodes in the cluster:
 

```
cd /var/spool/torque/server_priv
vi nodes
```

  - a. Use a text editor to add one line to the `/var/spool/torque/server_priv/nodes` file for each compute node in your cluster. Use the `np` parameter to define the number of CPU threads that are available for use on each node. For example, we have two compute nodes we are defining, each with 64 threads available:
 

```
compute_node1.example.com np=64
compute_node2.example.com np=64
```
  - b. Copy the `/var/spool/torque/server_priv/nodes` file to the `/home/hpc/compute-node-files/` directory.
 

```
cp /var/spool/torque/server_priv/nodes /home/hpc/compute-node-files
```
7. Set up Maui as a service on the management node using the **chkconfig** command, as shown in the following example:
 

```
chkconfig --add maui.d
chkconfig --level 3456 maui.d on
chkconfig --list maui.d
maui.d 0:off 1:off 2:off 3:on 4:on 5:on 6:on
```
8. Start the Maui service on the management node as shown in the following example:
 

```
service maui.d start
Starting MAUI Scheduler: [OK]
```
9. Set up the TORQUE server on the management node by copying the `pbs_server` file to the `/etc/init.d/` directory, as shown in the following example:
 

```
cd /usr/local/torque-2.4.8
cp -p contrib/init.d/pbs_server /etc/init.d/
```
10. Add the `pbs_server` service to the management node with the **chkconfig** command, as shown in the following example:

**Note:** The `pbs_server` service is the TORQUE resource manager service.

```
chkconfig --add pbs_server
chkconfig --level 3456 pbs_server on
chkconfig --list pbs_server
pbs_server 0:off 1:off 2:off 3:on 4:on 5:on 6:on
```

- Restart the `pbs_server` service on the management node in order to allow changes to take effect. The `pbs_server` service starts the batch server on the management node that will pick up any jobs. You can start the `pbs_server` service as shown in the following example:

```
service pbs_server restart
Starting TORQUE Server: [OK]
```

## Results

The TORQUE and Maui services are installed and running on the management node. You can query the TORQUE queue and server attributes with the `qmgr -c 'p s'` command.

### Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

## Preparing the TORQUE packages for the compute nodes

ou must create Torque Resource Manager installation packages for the compute nodes before you can configure the Machine Oriented Mini-server on the compute nodes.

### Before you begin

Ensure that you have completed all of these prerequisite tasks:

- Build and install the Torque and Maui packages on the management node.
- Build the self-extracting installation packages and copy them to the `/home/hpc/compute-node-files` directory on the management node.
- Create the `hpc` user on the compute nodes as described in “Setting up the `hpc` user” on page 10.

### About this task

The root user on the management node performs these steps.

### Procedure

- Create the `/usr/local/torque-2.4.8/mom_priv-config` file on the management node:

**Note:** ou will copy this temporary file to the `/home/hpc/compute-node-files` directory.

- Provide the following information in the `/usr/local/torque-2.4.8/mom_priv-config` file:
  - IP address of the `pbsserver`, (the same as the IP address of the management node)
  - IP address of the `clienthost`, (the same as the IP address of the management node)
  - Settings for important directories that are used by the Machine Oriented Mini-server

The following example of the `mom_priv-config` file content shows how to format this information.

**Note:** The IP address for your management node is specific to your environment. The rest of the file must be the same as this example if you have configured your `hpc` user as described in this document.

```
$pbsserver x.x.x.x # (IP address of management node)
$clienthost x.x.x.x # (IP address of management node)
$usecp */home/hpc /home/hpc
$usecp */home /home
$usecp */root /root
$usecp */tmp.scratch /tmp.scratch
$tmpdir /tmp
```

- b. Copy the `/usr/local/torque-2.4.8/mom_priv-config` file to the `/home/hpc/compute-node-files` directory.
2. Prepare for the configuration of the `pbs_mom` service on each compute node. The **`pbs_mom`** command starts a batch Machine Oriented Mini-server that will control job execution as directed by the TORQUE resource manager service, usage limits, and notifications.
  - a. Update `/usr/local/torque-2.4.8/contrib/init.d/pbs_mom` file on the management node by adding `ulimit -l unlimited` as shown in the following example `pbs_mom` file:
 

```
#!/bin/sh
#
pbs_mom This script will start and stop the PBS Mom
#
chkconfig: 345 95 5
description: TORQUE/PBS is a versatile batch system for SMPs and clusters
#
ulimit -n 32768
ulimit -l unlimited # <--- add this line
Source the library functions
. /etc/rc.d/init.d/functions
```
  - b. Copy the `/usr/local/torque-2.4.8/contrib/init.d/pbs_mom` to the `/home/hpc/compute-node-files` directory.
  - c. Use the `cp -p` command to preserve mode and ownership of the file as shown in the following example:
 

```
cp -p contrib/init.d/pbs_mom /home/hpc/compute-node-files/
```
3. Copy the `/home/hpc/compute-node-files` directory as an archive file from the management node onto each of the compute nodes:
  - a. Create a tar archive of the contents of the `/home/hpc/compute-node-files` directory, as shown in the following example:

**Note:** All of the files shown in this example must be included in the tar file.

```
cd /home/hpc
tar zcvf compute-node-files.tar.gz compute-node-files
compute-node-files/
compute-node-files/torque-package-clients-linux-powerpc64.sh
compute-node-files/torque-package-devel-linux-powerpc64.sh
compute-node-files/torque-package-doc-linux-powerpc64.sh
compute-node-files/torque-package-mom-linux-powerpc64.sh
compute-node-files/torque-package-server-linux-powerpc64.sh
compute-node-files/nodes
compute-node-files/maui.sh
compute-node-files/mom_priv-config
compute-node-files/pbs_mom
```

- b. Copy the archive to the `/home/hpc` directory on each compute node. In the following example there are two compute nodes in the cluster, with the host names `compute_node1` and `compute_node2`:
 

```
scp compute-node-files.tar.gz compute_node1:/home/hpc
scp compute-node-files.tar.gz compute_node2:/home/hpc
```

## Results

You are ready to extract and install the TORQUE packages on each compute node.

### Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

## Configuring TORQUE on the compute nodes

In order to use the Torque Resource Manager to control jobs in the cluster, you need to configure the Machine Oriented Mini-server on each compute node.

### Before you begin

Complete the procedure described in “Preparing the TORQUE packages for the compute nodes” on page 20.

### About this task

You need to follow these steps on each compute node.

The root user on the compute node performs this procedure.

### Procedure

1. Extract the `compute-node-files.tar.gz` archive and install each TORQUE package as shown in the following example:

```
cd /home/hpc

tar xzf compute-node-files.tar.gz

cd compute-node-files/

./torque-package-clients-linux-powerpc64.sh --install
Installing TORQUE archive...

Done.

./torque-package-mom-linux-powerpc64.sh --install
```

```
Installing TORQUE archive...
```

```
Done.
```

2. Copy the Machine Oriented Mini-server files to the appropriate directories with the following commands:

```
cp /home/hpc/compute-node-files/mom_priv-config /var/spool/torque/mom_priv/config
cp /home/hpc/compute-node-files/pbs_mom /etc/init.d/
```

3. Prepare the `pbs_mom` service with the following commands:

```
chkconfig --add pbs_mom
chkconfig --level 3456 pbs_mom on
chkconfig --list pbs_mom
```

4. Start the `pbs_mom` service with the following command:

```
service pbs_mom start
```

## Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

## Validating job submissions

Verify that TORQUE resource manager and Maui Cluster Scheduler are properly configured by submitting a basic test job and a multi-node test job.

### Before you begin

Ensure that you have completed the following tasks:

- “Installing Maui cluster scheduler” on page 18
- “Installing Torque resource manager” on page 17
- “Preparing the TORQUE packages for the compute nodes” on page 20
- “Configuring TORQUE on the compute nodes” on page 22

Restart the operating system on each node.

### About this task

The hpc user submits the test jobs on the cluster.

**Note:** The root user cannot submit the jobs on the cluster because of the TORQUE security controls.

The job basic job test and multi-node job test verify resource handling and job scheduling on your nodes.

### Procedure

1. Verify that the Maui and TORQUE services are running with the following steps. Because the torque and maui software have been configured as services, the processes should automatically start after rebooting.

- a. On the management node, verify that the maui service and the pbs\_server service are running. When these services are running, you can see output from the ps command similar to the output shown in the following example:

```
ps -ef | grep -E "pbs|maui"
root 4091 1 0 11:16 ? 00:00:00 /usr/local/maui-3.3/sbin/maui
root 4525 1 0 11:16 ? 00:00:00 /usr/local/sbin/pbs_server
root 7253 7077 0 11:28 pts/1 00:00:00 grep -E pbs|maui
```

- b. On the compute nodes, verify that the pbs\_mom service is running. When this service is running, you can see output from the ps command similar to the output shown in the following example:

```
ps -ef | grep pbs
root 4467 1 0 11:16 ? 00:00:00 /usr/local/sbin/pbs_mom -p
root 7144 7046 0 11:26 pts/1 00:00:00 grep pbs
```

2. Perform a basic job test on the management node with the following steps:

- a. Submit the job using the qsub command and view the status of the job using the qstat command as the hpc user on the management node, as shown in the following example:

```
$ echo "sleep 30" | qsub
1.mgmt_node.example.com
```

```
$ qstat
Job id Name User Time Use S Queue

1.mgmt_node STDIN hpc 0 R batch
```



The output of the `qstat` command shows the following characteristics of the job:

**Job id** The Job id field shows the ID associated with the job and the node where the job is running. The ID of the job running on the `mgmt_node` node in this example is 1.

**Name** The Name field shows the name defined on the `#PBS -N` line in the PBS job file. If no job name is defined in the PBS job file, the default value of the Name field is the file name of the job. In this example, the job is from standard input, so the job name is `STDIN`.

**User** The User field shows the name of the user that submitted the job.

**Time Use**

The Time Use field shows the amount of CPU time used by the job.

**S** The S field shows the status of the job. In this example, the value of the S field is `R`, indicating that the job is running. When the job is complete, the value of the status field is `C`.

**Note:** If the value of the status column is `Q` (queued) when you run this test job, there is probably a configuration issue. If this occurs, review the installation and configuration steps to ensure they were completed correctly.

**Queue**

The Queue field shows the name of the queue that is handling the job. In this example, the job is the queue named `batch`.

3. To verify that the cluster is configured correctly, use the `mpirun` command to run the `hostname` command on each node to ensure that processes run successfully on each node. The parameters used in the following example with the `mpirun` command are:

- The `-np` flag specifies the total number of copies of the process to run on the nodes.
- The `-host` flag specifies the host names where the process will run.
- The last parameter of the `mpirun` command is the command to run on the nodes.

The following example shows how to use the `mpirun` command to run a simple process on all the nodes of your cluster.

```
$ mpirun -np 2 --host compute_node1.example.com,compute_node2.example.com hostname
compute_node1.example.com
compute_node2.example.com
```

If the `hostname` command runs successfully on each node, the host name of each node appears in the output of the `mpirun` command shown in this example. In this example, the output shows the host names of the two compute nodes in the test cluster, `compute_node1`, and `compute_node2`.

4. Perform a simple multi-node test using a PBS script with the following steps:
  - a. On the management node, create a PBS script similar to the following example. Note the following in the example:
    - The lines beginning with the number sign (`#`) in the PBS script are TORQUE directives and are used to set TORQUE environment variables for the job.
    - In the following example, the file name of the PBS script is `multinode.pbs`.
    - In the following example, the host names of the compute nodes are `compute_node1` and `compute_node2`.
    - In the following example, the `-np` flag of the `mpirun` command is used to specify that two copies of the `hostname` command will run.

```
#PBS -N node_test
#PBS -q batch
#PBS -l nodes=2,pmem=100m,walltime=00:30:00
#PBS -M admin@example.com
#PBS -m e
#PBS -V

mpirun -np 2 --host compute_node1,compute_node2 hostname
```



- b. Submit the PBS script as the hpc user on the management node, and check the status of the job as shown in the following example:

```
$ qsub multinode.pbs
11.mgmt_node.example.com
[hpc@mgmt_node ~]$ qstat
```

| Job id      | Name      | User | Time Use | S | Queue |
|-------------|-----------|------|----------|---|-------|
| 1.mgmt_node | node_test | hpc  | 00:00:00 | C | batch |

This example shows a job submission, the returned job number (1), and the qstat output which indicates whether the job is running or has already completed.

#### Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

---

## Installing additional advanced software

There are several advanced software packages you can use to enhance the capability of compute nodes for a POWER processor-based cluster. This discussion includes basic installation and configuration instructions for libhugetlbfs, the IBM Advance Toolchain for PowerLinux, the ATLAS math libraries, and the IBM XL C/C++ and IBM XL Fortran compilers

#### Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

## Installing libhugetlbfs

You can boost performance for some HPC workloads by using 16MB huge pages provided by the libhugetlbfs library.

### About this task

You must install and configure all nodes (including the management node) because the libhugetlbfs environment variables are applied on the management node for jobs that run on the compute nodes.

The root user performs all the steps of this task.

### Procedure

1. Download the newest libhugetlbfs library from the project home, at the following URL:  
<http://sourceforge.net/projects/libhugetlbfs/files/>
2. Prepare and install the library. Note that libhugetlbfs build process will create both the 32-bit and the 64-bit versions on the Power architecture.
  - a. Extract the archive file, as shown in the following example:  

```
tar -C /usr/local/src -zxf libhugetlbfs-2.9.tar.gz
```
  - b. Change to the top-level directory of the extracted source, as shown in the following example:  

```
cd /usr/local/src/libhugetlbfs-2.9/
```
  - c. Build and install the software package, as shown in the following example:  

```
make
make install PREFIX=/usr/local
```
3. Create and mount the /libhugetlbfs file system.

- ```
# mkdir /libhugetlbfs
# mount -t hugetlbfs hugetlbfs /libhugetlbfs
```
4. Give the hpc user group privileges to use /libhugetlbfs.
 - a. Create the libhuge group, as shown in the following example:


```
# groupadd libhuge
```
 - b. Make the hpc user a member of the libhuge group, as shown in the following example:


```
# usermod -a -G libhuge hpc
```
 - c. Give the libhuge group privileges to use the /libhugetlbfs directory and set the permissions as shown in the following example:


```
# chgrp libhuge /libhugetlbfs
# chmod 777 /libhugetlbfs
```
 5. Reserve huge pages for compute jobs to use, by providing the number of desired pages in the nr_hugepages file. Clear any existing huge pages before you reserve a new number of huge pages, as shown in the following example:


```
# echo 0 > /proc/sys/vm/nr_hugepages
# echo 100 > /proc/sys/vm/nr_hugepages
```
 6. Validate the configuration of libhugetlbfs.
 - a. Verify that the permissions of the /libhugetlbfs file system are set correctly, as shown in the following example:


```
# ls -ld /libhugetlbfs/
drwxrwxrwx 2 root libhuge 0 Oct 8 20:40 /libhugetlbfs/
```
 - b. Verify that the libhuge ld scripts are located in the directory where libhugetlbfs is installed, as shown in the following example:


```
ls /usr/local/share/libhugetlbfs/
ld ld.hugetlbfs ldscripts
```
 - c. Verify that the correct number of huge pages is allocated by checking /proc/meminfo as shown in the following example:


```
# cat /proc/meminfo | grep Huge
```
 7. Make the settings for libhugetlbfs persistent by having the configuration occur when the node starts up.
 - a. Edit the /etc/fstab file to include the following line:


```
hugetlbfs    /libhugetlbfs    hugetlbfs    mode=0777,gid=1000    0 0
```

Note: The group ID for the libhuge group in this example is 1000. Verify the libhuge group ID by running the **id** command as the hpc user.
 - b. Edit the /etc/sysctl.conf file to include the following parameters:


```
vm.nr_hugepages=0
vm.nr_hugepages=100
```
 - c. Run the **sysctl -p** command to load settings from the default sysctl configuration file, /etc/sysctl.conf.

Results

When you have completed this process for each node, you can use 16MB huge pages on your cluster.

Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Installing IBM Advance Toolchain for PowerLinux

The IBM Advance Toolchain for PowerLinux provides potential performance benefits by using newer open source compilers and runtime libraries than those that are provided in most Linux distributions.

About this task

If you determine that your applications will benefit from the IBM Advance Toolchain for PowerLinux, perform this process on each node of the cluster.

The root user performs the installation process.

The hpc user must perform some of the of the validation steps.

Procedure

1. Download the three IBM Advance Toolchain for PowerLinux 3.0 RPM packages for Red Hat Enterprise Linux from the following URL: <ftp://linuxpatch.ncsa.uiuc.edu/toolchain/at/at3.0/redhat/RHEL5>

The following list shows the names of the specific files to download:

- advance-toolchain-at3.0-runtime-3.0-0.ppc64.rpm
- advance-toolchain-at3.0-devel-3.0-0.ppc64.rpm
- advance-toolchain-at3.0-perf-3.0-0.ppc64.rpm

2. Install the RPM packages in the order shown in the following example:

```
# rpm -i advance-toolchain-at3.0-runtime-3.0-0.ppc64.rpm
# rpm -i advance-toolchain-at3.0-devel-3.0-0.ppc64.rpm
# rpm -i advance-toolchain-at3.0-perf-3.0-0.ppc64.rpm
```

The default installation location for the IBM Advance Toolchain for PowerLinux is the /opt/at3.0/ directory. The IBM Advance Toolchain for PowerLinux **gcc** compilers are installed in the /opt/at3.0/bin directory by default.

3. Run the **createldhuge** script to configure IBM Advance Toolchain for PowerLinux to work with libhugetlbfs, as shown in the following example:

```
# /opt/at3.0/scripts/createldhuge.sh \
/usr/local/share/libhugetlbfs/ld.hugetlbfs /opt/at3.0/
```

4. Confirm the version of the **gcc** command installed by the Linux distribution is the **gcc** executable provided by the Linux distribution.

```
# which gcc
/usr/bin/gcc
```

5. Make the IBM Advance Toolchain for PowerLinux gcc available for the hpc user.

- a. Add the IBM Advance Toolchain for PowerLinux bin directory in the PATH environment variable for the hpc user so that the hpc user will use this version of the **gcc** command by default, as shown in the following example:

```
# su - hpc
$ export PATH=/opt/at3.0/bin:$PATH
```

- b. Verify that the IBM Advance Toolchain for PowerLinux version of gcc is now the default gcc command used by the hpc user, as shown in the following example:

```
$ which gcc
/opt/at3.0/bin/gcc
```

Results

If you complete this process on all the nodes, the cluster is configured to use the IBM Advance Toolchain for PowerLinux with libhugetlbfs.

Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Installing the IBM XL C/C++ and Fortran compilers

Users who want the advantages of optimized compilers may benefit from the IBM XL compilers.

Before you begin

You can find more information about getting a license for these compilers at the following web sites:

- XL C/C++ for Linux, V11.1 : <http://www.ibm.com/software/awdtools/xlcpp/linux/>
- XL Fortran for Linux, V13.1: <http://www.ibm.com/software/awdtools/fortran/xlfortran/linux/>

Note: Users can also take advantage of the free 60-day trial for the IBM XL compilers. Here we provide instructions on how to download and install the trial versions.

You must ensure that the required compiler toolchain packages from the Linux distributor are installed on the system before you install the IBM XL compilers. You can do this with the **yum install** command as shown in the following example:

```
# yum install gcc gcc-c++ glibc.ppc glibc.ppc64 glibc-devel.ppc glibc-devel.ppc64 \
libgcc.ppc libgcc.ppc64 libstdc++.ppc libstdc++.ppc64 libstdc++-devel.ppc \
libstdc++-devel.ppc64 perl
```

About this task

If you choose to use the XL C/C++ and XL Fortran compilers, perform this process on each node of the cluster.

The root user performs the installation process.

Procedure

1. Download the XL compiler trial versions from the following URLs:

- XL C/C++ for Linux, V11.1: https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=swg-xlcc111&S_CMP=rnav
- XL Fortran for Linux, V13.1: https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=swg-xf131&S_CMP=rnav

Note: You have to accept the license agreement for both compilers before you can go to the **Download using http** tab and click **Download now** to download the compiler archive files.

2. Install the XL C/C++ compiler.

- a. Prepare and install the XL C/C++ compiler from the directory where you extract the archive file, as shown in the following example. This process will install the compiler in the `/opt/ibmcomp/` directory by default:

```
mkdir /usr/local/src/xlc
# tar -C /usr/local/src/xlc -zxf xlc.11.1.0.0.linux.eval.tar.gz
# cd /usr/local/src/xlc/
# ./xlc_install -rpmloc images/rpms/
```

The following example output shows the portions of the **xlc_install** command output that you need to interact with the installation process. The places where output was omitted from this example are indicated by the three periods ("..."):

Press Enter to continue viewing the license agreement, or, Enter "1" to accept the agreement, "2" to decline it or "99" to go back to the previous screen, "3" Print, "4" Read non-IBM terms.

1

...

```

Link          Source
=====
/usr/bin/gxlc  /opt/ibmcmp/vacpp/11.1/bin/gxlc
/usr/bin/gxlc++ /opt/ibmcmp/vacpp/11.1/bin/gxlc++
/usr/bin/gxlc  /opt/ibmcmp/vacpp/11.1/bin/gxlc
/usr/bin/xlc   /opt/ibmcmp/vacpp/11.1/bin/xlc
/usr/bin/xlc++ /opt/ibmcmp/vacpp/11.1/bin/xlc++
/usr/bin/xlc   /opt/ibmcmp/vacpp/11.1/bin/xlc
/usr/bin/xlc_r /opt/ibmcmp/vacpp/11.1/bin/xlc_r
/usr/bin/xlc++_r /opt/ibmcmp/vacpp/11.1/bin/xlc++_r
/usr/bin/xlc_r /opt/ibmcmp/vacpp/11.1/bin/xlc_r
=====
Do you want to proceed with the symbolic links?
Please type "yes" or "no": yes

```

...

Installation Completed.
Filename of the installation log: /opt/ibmcmp/vacpp/11.1/xlc_install.log

- b. Run the **xlc** command to verify that you see the compiler information page, as shown in the following example:

```

# xlc
xlc(1)                IBM XL C/C++ for Linux, V11.1                xlc(1)

NAME
    xlc, xlc++, xlc, cc, c89, c99 and related commands - invoke the IBM XL
    C/C++ compiler.

SYNTAX
    <invocation-command> [ <option> | <inputfile> ]

```

...

3. Install the XL Fortran compiler.

- a. Prepare and install the XL Fortran compiler from the directory where you extract the archive file, as shown in the following example. This process will install the compiler in the /opt/ibmcmp/ directory by default:

```

# mkdir /usr/local/src/xlf
# tar -C /usr/local/src/xlf -zxf xlf.13.1.0.0.linux.eval.tar.gz
# cd /usr/local/src/xlf/
# ./xlf_install -rpmloc images/rpms/

```

The following example output shows the portions of the **xlf_install** command output which you need to interact with the installation process. The places where output was omitted from this example are indicated by the three periods ("..."):

```

Type "yes" even if you have "IBM XL SMP Version 2.1.0.0-100630 for Linux",
"IBM XL MASS Version 6.1.0.0-100630 for Linux" already installed at the
default installation path.
Do you want to proceed with the installation?
Please type "yes" or "no": yes

```

...

Press Enter to continue viewing the license agreement, or, Enter "1" to accept the agreement, "2" to decline it or "99" to go back to the previous screen,

"3" Print, "4" Read non-IBM terms.

1

...

Do you want to proceed with the symbolic links?
Please type "yes" or "no": yes

...

Installation Completed.

Filename of the installation log: /opt/ibmcmp/xlf/13.1/xlf_install.log

- b. Run the **xlf** command to verify that you see the compiler information page, as shown in the following example:

```
# xlf
xlf(1)                IBM XL Fortran for Linux, V13.1                xlf(1)

NAME
    xlf, xlf_r, f77, fort77, xlf90, xlf90_r, f90, xlf95, xlf95_r, f95,
    xlf2003, xlf2003_r, f2003 - invoke the IBM XL Fortran compiler.

SYNTAX
    <invocation-command> [ <option> | <inputfile> ]
...
```

Results

If you complete this process on all the nodes of your cluster, the XL C/C++ and XL Fortran compilers are installed and ready to use.

Related concepts:

Chapter 1, "Scope, requirements, and support," on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Installing ATLAS math libraries

Automatically Tuned Linear Algebra Software (ATLAS) is a math library that can be used to compile workloads such as Linpack and HPC Challenge.

Before you begin

Download the newest stable version of ATLAS. which is 3.8.3 at the time this paper was written:
<http://sourceforge.net/projects/math-atlas/files/>

Note: At the time this document was written, the newest stable version was ATLAS 3.8.3, which is used in the examples in this document.

About this task

The build process takes several hours to complete, so you may prefer to compile the software on one node and copy the compiled versions to your other nodes with the identical processor architecture.

The root user performs all the steps of this process.

Procedure

1. Extract the ATLAS tar archive file, as shown in the following example:

```
# tar -jxf atlas3.8.3.tar.bz2 -C /usr/local/
```

2. Configure the ATLAS setup.

As root, go into the top level ATLAS directory and create a build directory where you will run the configure and make commands, as shown in the following example:

```
# cd /usr/local/ATLAS/
# mkdir obj64
# cd obj64/
# ../configure -b 64
```

- The example uses a directory called obj64 as the 64-bit library build location.
- The -b 64 option tells the ATLAS **configure** command to build the 64-bit libraries.

3. Run the make step to build, test, and optimize the library.

Note: This step may take several hours to complete. The ATLAS math libraries employ a self-tuning automated optimization process of building, running, assessing the results, and using the best options. The following example shows the invocation of the **make** command, and modified output with the last several lines of message produced by the **make** command. The omitted lines of output are represented by a line containing three periods ("...") in the example output.

```
# make
...
ATLAS install complete. Examine
ATLAS/bin/<arch>/INSTALL_LOG/SUMMARY.LOG for details.
make[1]: Leaving directory `/usr/local/ATLAS/obj64'
make clean
make[1]: Entering directory `/usr/local/ATLAS/obj64'
rm -f *.o x* config?.out *core*
make[1]: Leaving directory `/usr/local/ATLAS/obj64'
```

4. After the **make** command completes, run the **make check** command to check the libraries. This command produces a report showing a list of tests that either pass or fail. Ensure that all of the tests pass.
5. Install the libraries. The following example shows a listing of the libraries that the **make install** command installs in the /usr/local/ATLAS/obj64/lib/ directory:

```
# make install
# ls lib/
libatlas.a libf77blas.a libptcblas.a libtstatlas.a Make.inc
libcblas.a liblapack.a libptf77blas.a Makefile
```

Results

After you install the ATLAS libraries on the nodes of the cluster, the libraries are ready to use.

Example

The order you use to link to the ATLAS libraries can affect how they function. The ATLAS FAQs provide more information about this subject. The following example shows a correct order to link the libraries:

```
-lc -L/usr/local/ATLAS/obj64/lib/ -llapack -lptcblas -lptf77blas -latlas
```

Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Building specialized versions of Open MPI

You can configure Open MPI to integrate better and make better use of other components in an HPC software stack.

Open MPI is highly configurable both at build time and at run time. Although Open MPI will function well on POWER-based systems as installed from the IBM Installation Toolkit, you can optimize Open MPI to work better with other software that is installed on the HPC cluster. This topic discusses methods

to configure Open MPI to work better with various software components that are discussed in this blueprint. Specifically, this discussion covers integrating the following packages with Open MPI:

- TORQUE resource manager
- IBM Advance Toolchain for PowerLinux
- IBM XL C/C++ and Fortran compilers
- libhugetlbfs

Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Building Open MPI with TORQUE

Configuring Open MPI to work directly with the TORQUE resource manager can simplify launching and managing MPI jobs across a set of resources.

Before you begin

Ensure that the TORQUE resource manager is installed on the nodes in your cluster, as described in Chapter 4, “Installing the Management Node,” on page 17.

About this task

The hpc user performs most of the steps in this task.

The root user performs the build and installation processes in this task.

This procedure can be used to install any new versions of Open MPI.

The process shows how to build and make the package on the management node. You can then create a tar archive file of the /usr/local/openmpi library and then copy the archive file to each compute node and then extract the files. After these steps, you must change the PATH and LD_LIBRARY settings in the .bashrc file to use the new library.

Procedure

1. On the management node, download the latest Open MPI source tar file in the 1.4.x release series from the following location: <http://www.open-mpi.org/software/ompi/v1.4/>

Note: This example, uses Open MPI version 1.4.2, which is the latest stable release of Open MPI at the time of this writing.

2. As the root user on the management node, extract the sources and go to the source directory, as shown in the following example:

```
# tar xjf openmpi-1.4.2.tar.bz2 -C /usr/local/src
# cd /usr/local/src/openmpi-1.4.2
```

3. On the management node, configure and build a Torque-knowlegeable version of Open MPI. Do this by using the **--with-tm** flag for the configure script. Specifically, set the value of the **--with-tm** flag to the directory location where Open MPI can reference include/tm.h. If you have followed the instructions in this blueprint so far, the directory to use is /usr/local/torque-2.4.8/src as shown in the following example. After the configure step, run the **make** command and the **make install** command.

```
#!/configure --prefix=/usr/local/openmpi-1.4.2-tm \
--with-platform=contrib/platform/ibm/optimized-ppc64-gcc --with-tm=/usr/local/torque-2.4.8/src
# make
# make install
```


4. On the management node, update the relevant environment variables of the hpc user to enable use of the TORQUE resource manager-aware version of Open MPI as shown in the following example:

```
# su - hpc
$ export PATH=/usr/local/openmpi-1.4.2-tm/bin:$PATH
$ export LD_LIBRARY_PATH=/usr/local/openmpi-1.4.2-tm/lib:$LD_LIBRARY_PATH
```

Note: You may also remove the previous version of Open MPI from PATH and LD_LIBRARY_PATH. However the syntax in this example sets the updated library to be found first in the path, so any older library in the path has no effect.

You can use the following example to make these updates to the .bashrc file:

```
$ cat >> /home/hpc/.bashrc <<EOF
export PATH=/usr/local/openmpi-1.4.2-tm/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/openmpi-1.4.2-tm/lib:$LD_LIBRARY_PATH
EOF
```

5. On the management node, run **ompi_info** and use the **grep** command to retrieve the lines containing "plm" to ensure that the new build of Open MPI has TORQUE integration. If the TORQUE manager component built correctly, you will see the tm entry listed as a component of the plm (Process Launch Module) framework, as shown in the following example:

```
$ ompi_info | egrep plm:
MCA plm: rsh (MCA v2.0, API v2.0, Component v1.4.2)
MCA plm: slurm (MCA v2.0, API v2.0, Component v1.4.2)
MCA plm: tm (MCA v2.0, API v2.0, Component v1.4.2)
```

6. On the management node, as the root user, prepare the tar archive file of the openmpi-1.4.2-tm directory contains the updated Open MPI library, and distributed the archive to all of the compute nodes, as shown in the following example:

```
# cd /usr/local
# tar cvf openmpi-1.4.2-tm.tar openmpi-1.4.2-tm
# scp openmpi-1.4.2-tm.tar compute_node1:/usr/local/
# scp openmpi-1.4.2-tm.tar compute_node2:/usr/local/
```

Note: In this example, the compute nodes are named compute_node1 and compute_node2.

7. On each of the compute nodes, install the TORQUE resource manager-aware version of Open MPI, and set up the environment for the hpc user.
 - a. As the root user on each compute node, extract the tar archive, as shown in the following example

```
# cd /usr/local
# tar xvf openmpi-1.4.2-tm.tar
```

- b. As the hpc user on each compute node, configure the appropriate environment variables, as shown in the following example:

```
$ export PATH=/usr/local/openmpi-1.4.2-tm/bin:$PATH
$ export LD_LIBRARY_PATH=/usr/local/openmpi-1.4.2-tm/lib:$LD_LIBRARY_PATH
```

You can use the following example to make these updates to the .bashrc file:

```
$ cat >> /home/hpc/.bashrc <<EOF
export PATH=/usr/local/openmpi-1.4.2-tm/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/openmpi-1.4.2-tm/lib:$LD_LIBRARY_PATH
EOF
```

- c. On each compute node, run **ompi_info** and use the **grep** command to retrieve the lines containing "plm" to ensure that the new build of Open MPI has TORQUE integration as shown in the following example:

```
$ ompi_info | egrep plm:
MCA plm: rsh (MCA v2.0, API v2.0, Component v1.4.2)
MCA plm: slurm (MCA v2.0, API v2.0, Component v1.4.2)
MCA plm: tm (MCA v2.0, API v2.0, Component v1.4.2)
```

Configuring Open MPI for use with IBM Advance Toolchain for PowerLinux

You can configure Open MPI for use with IBM Advance Toolchain for PowerLinux with a single command string.

Before you begin

Ensure that the Open MPI and the IBM Advance Toolchain for PowerLinux RPMs are installed on the system.

About this task

The following procedure should be completed on all nodes.

Procedure

1. Adjust the PATH environment variable so that the Advance Toolchain installation bin directory is at the head of the PATH list, as shown in the following example:

```
$ export PATH=/opt/at3.0/bin:$PATH
```

Once this is done, all builds done with the Open MPI wrapper compilers (such as mpicc, mpif90, and so on), regardless of which Open MPI build you are using, will use the compilers and runtime environment provided by the IBM Advance Toolchain for PowerLinux.

2. Validate the use of IBM Advance Toolchain for PowerLinux with the following steps:
 - a. Run the **which gcc** command and ensure that the **gcc** used is coming from the /opt/at3.0/bin directory, as shown in the following example:

```
$ which gcc
/opt/at3.0/bin/gcc
```
 - b. Build an MPI program using the mpicc compiler wrapper (now using IBM Advance Toolchain for PowerLinux) and run the **ldd** command on the resulting executable. Make sure that the libc.so.6 library (and other libraries) resolve to libraries in the /opt/at3.0/lib64 directory hierarchy:

Note:

```
$ mpicc -o hello_c hello_c.c
$ ldd hello_c
    ldd hello_c
    linux-vdso64.so.1 => (0x0000000000100000)
    libmpi.so.0 => /usr/local/openmpi-1.4.2-tm/lib/libmpi.so.0 (0x0000040000040000)
    libopen-rte.so.0 => /usr/local/openmpi-1.4.2-tm/lib/libopen-rte.so.0 (0x0000040000140000)
    libopen-pal.so.0 => /usr/local/openmpi-1.4.2-tm/lib/libopen-pal.so.0 (0x00000400001c0000)
    libdl.so.2 => /opt/at3.0/lib64/libdl.so.2 (0x0000040000250000)
    libnsl.so.1 => /opt/at3.0/lib64/libnsl.so.1 (0x0000040000270000)
    libutil.so.1 => /opt/at3.0/lib64/libutil.so.1 (0x00000400002a0000)
    libm.so.6 => /opt/at3.0/lib64/power6/libm.so.6 (0x00000400002c0000)
    libpthread.so.0 => /opt/at3.0/lib64/power6/libpthread.so.0 (0x00000400003a0000)
    libc.so.6 => /opt/at3.0/lib64/power6/libc.so.6 (0x00000400003e0000)
    /opt/at3.0/lib64/ld64.so.1 (0x0000040000000000)
```

Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Configuring Open MPI for use with IBM XL compilers

Put your short description here; used for first paragraph and abstract.

Before you begin

Ensure that Open MPI and the IBM XL Compiler RPMs are installed on the system.

About this task

This procedure will enable the use of Open MPI to build and run HPC applications with the IBM XL compilers. The use of the XL compilers can provide significant performance improvements over the distro-supplied compilers.

This procedure does not require that Open MPI be reconfigured or rebuilt. It relies solely on enhancing the Open MPI implementation to be knowledgeable of the XL compilers, therefore it can be done for either Open MPI build (the rpm installed version or the version integrated with torque).

The procedure given below is specifically for the xlc compiler. However, the same procedure can be followed to enable all of the languages and variants supported by the XL compilers to be used with Open MPI.

On all nodes, as the root user, follow the procedures below.

Procedure

1. To create a wrapper compiler to integrate the use of xlc with Open MPI, complete the following steps:

- a. Run the following command:

```
# ompi_install_dir=/usr/local/openmpi-1.4.2-tm/
```

- b. In the Open MPI binary directory, symbolically link opal_wrapper to the desired wrapper compiler name (in this case, mpixlc):

```
# ln -s ${ompi_install_dir}/bin/opal_wrapper ${ompi_install_dir}/bin/mpixlc
```

- c. In the Open MPI share/openmpi directory, copy the mpicc-wrapper-data.txt file to the mpixlc-wrapper-data.txt file:

```
# cd ${ompi_install_dir}/share/openmpi
# cp mpicc-wrapper-data.txt mpixlc-wrapper-data.txt
```

- d. Edit the mpixlc-wrapper.txt file as shown in the following example:

```
# diff mpicc-wrapper-data.txt mpixlc-wrapper-data.txt
14c14
< compiler=gcc
---
> compiler=xlc
17,18c17,18
< compiler_flags=-pthread -m64
< linker_flags=      -m64
---
> compiler_flags=-qthreaded -q64
> linker_flags=      -q64
```

Once this is done, you should be able to build your applications with mpixlc and it will use the IBM xlc compiler.

2. To create a wrapper compiler to integrate the use of xlf with Open MPI, complete the following steps:

- a. Run the following command:

```
# ompi_install_dir=/usr/local/openmpi-1.4.2-tm/
```

- b. In the Open MPI binary directory, symbolically link opal_wrapper to the desired wrapper compiler name (in this case, mpixlf):

```
# ln -s ${ompi_install_dir}/bin/opal_wrapper ${ompi_install_dir}/bin/mpixlf
```

- c. In the Open MPI share/openmpi directory, copy the mpi77-wrapper-data.txt file to the mpixlf-wrapper-data.txt file.

```
# cd ${ompi_install_dir}/share/openmpi
# cp mpif77-wrapper-data.txt mpixlf-wrapper-data.txt
```

- d. Edit the `mpixlf-wrapper.txt` file as shown in the following example:

```
# diff mpif77-wrapper-data.txt mpixlf-wrapper-data.txt
14c14
< compiler=gfortran
---
> compiler=xlf
17,18c17,18
< compiler_flags=-pthread -m64
< linker_flags=      -m64
---
> compiler_flags=-qthreaded -q64
> linker_flags=      -q64
```

Once this is done, you should be able to build your applications with `mpixlf` and it will use the IBM `xlf` compiler.

3. To verify the compiler options that are being used, run the compiler with the `--showme` option (this step can be done as `hpc` user):

```
$ mpixlc --showme
xlc -I/usr/local/openmpi-1.4.2/include -qthreaded -q64 -q64 -L/usr/local/openmpi-1.4.2/lib -lmpi \
-lopenrte -lopenpal -ldl -Wl,--export-dynamic -lnsl -lutil -lm -ldl

$ mpixlf --showme
xlf -I/opt/openmpi/1.4.2/include -qthreaded -q64 -q64 -L/opt/openmpi/1.4.2/lib -lmpi_f77 -lmpi \
-lopen-rte -lopen-pal -ldl -Wl,--export-dynamic -lnsl -lutil -lm -ldl
```

Results

If `-I/usr/local/${ompi_install_dir}/include` and `-L/usr/local/${ompi_install_dir}/lib` are shown in the output, then you have successfully configured Open MPI to build and run HPC application with IBM XL compilers. Note that the other wrapper compilers for `mpif77`, `mpif90`, `mpic++`, and the `_r` variants can be handled similarly.

Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Using Open MPI with huge pages

If you have `libhugetlbfs` installed, you can choose to use 16MB huge pages with Open MPI when you start a distributed program.

Before you begin

Ensure that Open MPI is installed and that `libhugetlbfs` is installed.

About this task

The standard Open MPI configuration includes support for optimal use of RDMA-based communications, such as InfiniBand. This requires the use of special memory management routines within the Open MPI library to manage pages used in RDMA transfers. However, this is incompatible with the use of an interposer memory management library, such as `libhugetlbfs`.

To enable use of `libhugetlbfs` with Open MPI, the IBM Installation Toolkit contains a version of Open MPI which does not use the embedded Open MPI memory manager. After Open MPI is installed from the IBM Installation Toolkit, and you have your environment setup to use it, you can use `libhugetlbfs` as an

interposer library with the LD_PRELOAD mechanism. This enables existing software to take advantage of libhugetlbfs without recompiling or relinking.

Procedure

1. To make use of 16MB huge pages, start the distributed application with the environment variable LD_PRELOAD set to libhugetlbfs.so and HUGETLB_MORECORE set to yes. The following example shows how the hpc user can start an application with these settings:

```
$ mpirun -x LD_PRELOAD=libhugetlbfs.so -x HUGETLB_MORECORE=yes \  
-x OMPI_MCA_memory_ptmalloc2_disable=1 -np <#> <program>
```

Note: In this example, the **-x** flag of the **mpirun** command tells **mpirun** to propagate the designated environment setting to all of the processes of the distributed job.

2. Verify that huge pages are used by the application.

You can verify the use of libhugetlbfs by running a job which does dynamic memory allocation and then observing the values of HugePages_Total and HugePages_Free in /proc/meminfo. If the job uses huge pages, when the application runs, the value of HugePages_Free decreases from the value it shows before you run the program.

- a. Check the values of HugePages_Total and HugePages_Free before you run the program, as shown in the following example:

```
# egrep 'HugePages_(Total|Free)' /proc/meminfo  
HugePages_Total:    400  
HugePages_Free:     400
```

- b. Check the values of HugePages_Total and HugePages_Free while the program is running, as shown in the following example:

```
# egrep 'HugePages_(Total|Free)' /proc/meminfo  
HugePages_Total:    400  
HugePages_Free:      80
```

3. Check the values of HugePages_Total and HugePages_Free when the program is has completed, as shown in the following example:

```
# egrep 'HugePages_(Total|Free)' /proc/meminfo  
HugePages_Total:    400  
HugePages_Free:     400
```

Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Testing the InfiniBand interconnect performance with Intel MPI Benchmark

Validate the InfiniBand interconnect using a commonly used benchmark called Intel MPI Benchmark (IMB).

This section focuses on verifying the performance of the InfiniBand interconnects. This is an important step in verifying the performance of your cluster because if the interconnects cannot perform well, they can become a bottleneck for multi-node runs. You can complete the steps in each of the following sections as the hpc user.

Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Building IMB

IMB can be run on a single node, but for the purposes of testing InfiniBand performance, you will want to use at least two nodes for the test so that the messages will be passed across the InfiniBand link and not just within the shared memory of a single system.

About this task

Perform the following procedure on all of the nodes.

Procedure

1. Download IMB-3.2 from the following link: <http://software.intel.com/en-us/articles/intel-mpi-benchmarks/>
2. Untar the package and change directory into the /src directory with the following commands:

```
tar zxvf IMB_3.2.tgz -C /home/hpc
cd /home/hpc/imb/src/
```
3. Edit the `make_ict` makefile to change the assignment of the `CC` value from `mpiic` to `mpicc` as shown:

```
LIB_PATH      =
LIBS           =
CC             = mpicc
ifeq (, $(shell which ${CC}))
$(error ${CC} is not defined through the PATH <clipped> )
endif
OPTFLAGS      =
CLINKER        = ${CC}
LDFLAGS       =
CPPFLAGS      =

export CC LIB_PATH LIBS OPTFLAGS CLINKER LDFLAGS CPPFLAGS
include Makefile.base
```
4. Run the **make** command. When the **make** command completes, the IMB-MPI1 executable is shown with other object files and executables.

Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Running IMB

You can run IBM to determine whether you need to affinitize the processes to increase performance.

Procedure

1. Run IMB-MPI1 pingpong from management node:

```
$ cd /home/hpc
$ mpirun -np 2 -host compute_node1,compute_node2 imb/src/IMB-MPI1 pingpong
```

Verify that you see the IMB output on your screen.
2. Run IMB in a pbs job with the following steps. This example job runs three IMB-MPI1 workloads: pingpong, sendrecv, and exchange across two nodes.

Note: Perform this step for every InfiniBand link.

a. Create the `job.pbs` file as shown:

```
#PBS -N imb_2node
#PBS -q batch
#PBS -l nodes=2,pmem=100m,walltime=00:30:00
#PBS -M admin@example.com
#PBS -m e
#PBS -V
```

```
cd /home/hpc
mpirun -np 2 --host compute_node1,compute_node2 imb/src/IBM-MPI1 pingpong
```

3. Submit the job with the **qsub** command, as shown in the following example:

```
$ qsub job.pbs
3.mgmt_node.example.com
```

Results

Two new output files are available after running the job: one containing job outputs and another containing any errors. They are named `<job name>.o<job identifier>` and `<job name>.e<job identifier>`, respectively. These output files are located on the compute node where the job ran.

The following output is the `imb_2node.o3` file for the job that ran in the test environment, including the output of expected performance for the pingpong, sendrecv, and exchange workloads on a single InfiniBand link:

```
#-----
#   Intel (R) MPI Benchmark Suite V3.2, MPI-1 part
#-----
# Date           : Wed Aug 25 22:28:31 2010
# Machine        : ppc64
# System         : Linux
# Release        : 2.6.32.12-0.7-ppc64
# Version        : #1 SMP 2010-05-20 11:14:20 +0200
# MPI Version    : 2.1
# MPI Thread Environment: MPI_THREAD_SINGLE

# New default behavior from Version 3.2 on:

# the number of iterations per message size is cut down
# dynamically when a certain run time (per message size sample)
# is expected to be exceeded. Time limit is defined by variable
# "SECS_PER_SAMPLE" (=> IMB_settings.h)
# or through the flag => -time

# Calling sequence was:

# IMB-MPI1 pingpong sendrecv exchange

# Minimum message length in bytes:  0
# Maximum message length in bytes:  4194304
#
# MPI_Datatype           :  MPI_BYTE
# MPI_Datatype for reductions :  MPI_FLOAT
# MPI_Op                 :  MPI_SUM
#
#
# List of Benchmarks to run:

# PingPong
```

```
# Sendrecv
# Exchange
```

```
#-----
# Benchmarking PingPong
# #processes = 2
#-----
```

#bytes	#repetitions	t[usec]	Mbytes/sec
0	1000	3.42	0.00
1	1000	3.61	0.26
2	1000	3.63	0.53
4	1000	3.58	1.06
8	1000	3.37	2.26
16	1000	3.28	4.65
32	1000	3.33	9.15
64	1000	3.37	18.11
128	1000	3.51	34.75
256	1000	3.86	63.23
512	1000	4.54	107.44
1024	1000	5.83	167.51
2048	1000	7.93	246.42
4096	1000	12.39	315.15
8192	1000	16.83	464.20
16384	1000	18.94	825.08
32768	1000	27.44	1138.76
65536	640	44.30	1410.69
131072	320	78.01	1602.41
262144	160	145.32	1720.40
524288	80	280.61	1781.86
1048576	40	549.60	1819.50
2097152	20	1088.37	1837.60
4194304	10	2162.96	1849.32

```
#-----
# Benchmarking Sendrecv
# #processes = 2
#-----
```

#bytes	#repetitions	t_min[usec]	t_max[usec]	t_avg[usec]	Mbytes/sec
0	1000	3.23	3.23	3.23	0.00
1	1000	3.33	3.33	3.33	0.57
2	1000	3.39	3.39	3.39	1.13
4	1000	3.33	3.33	3.33	2.29
8	1000	3.34	3.35	3.34	4.56
16	1000	3.37	3.38	3.37	9.04
32	1000	3.39	3.39	3.39	18.01
64	1000	3.47	3.48	3.48	35.09
128	1000	3.66	3.66	3.66	66.71
256	1000	4.05	4.05	4.05	120.50
512	1000	4.90	4.91	4.90	199.01
1024	1000	6.07	6.07	6.07	321.81
2048	1000	8.26	8.26	8.26	473.14
4096	1000	12.46	12.46	12.46	627.01
8192	1000	17.12	17.13	17.13	912.20
16384	1000	23.07	23.07	23.07	1354.69
32768	1000	33.05	33.05	33.05	1891.07
65536	640	52.24	52.24	52.24	2392.58
131072	320	90.99	90.99	90.99	2747.44
262144	160	168.42	168.44	168.43	2968.34
524288	80	323.80	323.84	323.82	3087.96
1048576	40	637.67	637.78	637.72	3135.90
2097152	20	1261.31	1261.56	1261.43	3170.69
4194304	10	2502.89	2502.92	2502.91	3196.27

```
#-----
# Benchmarking Exchange
# #processes = 2
#-----
```


#bytes	#repetitions	t_min[usec]	t_max[usec]	t_avg[usec]	Mbytes/sec
0	1000	4.62	4.62	4.62	0.00
1	1000	4.66	4.66	4.66	0.82
2	1000	4.67	4.68	4.67	1.63
4	1000	4.67	4.67	4.67	3.27
8	1000	4.67	4.67	4.67	6.53
16	1000	4.61	4.62	4.61	13.23
32	1000	4.77	4.77	4.77	25.59
64	1000	4.78	4.79	4.78	51.02
128	1000	5.38	5.38	5.38	90.78
256	1000	5.81	5.81	5.81	168.05
512	1000	6.42	6.42	6.42	304.37
1024	1000	8.03	8.03	8.03	486.39
2048	1000	10.59	10.59	10.59	737.44
4096	1000	16.51	16.52	16.52	945.71
8192	1000	23.46	23.47	23.47	1331.55
16384	1000	44.33	44.33	44.33	1409.82
32768	1000	64.44	64.44	64.44	1939.73
65536	640	104.86	104.86	104.86	2384.21
131072	320	182.61	182.61	182.61	2738.03
262144	160	338.96	338.98	338.97	2950.01
524288	80	650.80	650.80	650.80	3073.13
1048576	40	1272.23	1272.23	1272.23	3144.10
2097152	20	2518.40	2518.70	2518.55	3176.24
4194304	10	5003.00	5003.31	5003.15	3197.88

All processes entering MPI_Finalize

Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Related information

You can find additional information about the processes and tools described in these procedures.

IBM Installation

Toolkit for PowerLinux

<http://www14.software.ibm.com/webapp/set2/sas/f/lopdiags/installtools>

Open MPI

<http://www.open-mpi.org/>

IBM Advance

Toolchain for PowerLinux

<ftp://linuxpatch.ncsa.uiuc.edu/toolchain>

TORQUE Resource Manager

<http://www.clusterresources.com/torquedocs21/>

Maui Cluster Scheduler

<http://www.clusterresources.com/products/maui-cluster-scheduler.php>

IBM XL C/C++ and XL Fortran Compilers

<http://www.ibm.com/software/awdtools/xlcpp/linux/>

<http://www.ibm.com/software/awdtools/fortran/xlfortran/linux/>

libhugetlbfs
<http://sourceforge.net/projects/libhugetlbfs/>

ATLAS math libraries
<http://sourceforge.net/projects/math-atlas/>

Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Appendix. Related information

You can find additional information about the processes and tools described in these procedures.

IBM Installation

Toolkit for PowerLinux

<http://www14.software.ibm.com/webapp/set2/sas/f/lopdiags/installtools>

Open MPI

<http://www.open-mpi.org/>

IBM Advance

Toolchain for PowerLinux

<ftp://linuxpatch.ncsa.uiuc.edu/toolchain>

TORQUE Resource Manager

<http://www.clusterresources.com/torquedocs21/>

Maui Cluster Scheduler

<http://www.clusterresources.com/products/maui-cluster-scheduler.php>

IBM XL C/C++ and XL Fortran Compilers

<http://www.ibm.com/software/awdtools/xlcpp/linux/>

<http://www.ibm.com/software/awdtools/fortran/xlfortran/linux/>

libhugetlbfs

<http://sourceforge.net/projects/libhugetlbfs/>

ATLAS math libraries

<http://sourceforge.net/projects/math-atlas/>

Related concepts:

Chapter 1, “Scope, requirements, and support,” on page 1

This blueprint applies to PowerLinux (POWER6 and later). You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Dept. LRAS/Bldg. 903
11501 Burnet Road
Austin, TX 78758-3400
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Trademarks

IBM, the IBM logo, and ibm.com[®] are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol ([®] and [™]), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Java[™] and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.



Printed in USA