Chameleon Cloud Tutorial

National Science Foundation

Program Solicitation # NSF 13-602

CISE Research Infrastructure: Mid-Scale Infrastructure - NSFCloud (CRI: NSFCloud)

MPI and HPCC

Objectives

In this tutorial, you will learn about Open MPI and HPCC on Chameleon machines.

#	Action	Detail	Time (min)
1	Introduction of OPEN MPI	You will learn what is OPEN MPI and how it works with Open Stack.	5
2	Password less authentication		5
3	Installation of Open MPI	To begin, we will learn the process of installing Open MPI	10
4	HPCC Introduction and installation	We will discuss about HPCC introduction and installation.	15
5	MPI and HPCC on Chameloen machines	Lastly, we will implement the HPC and MPI on Chameleon machines.	10

Prerequisites

- Knowledge of MPI and HPCC
- A SSH Client(how to use SSH)
- Basic Knowledge of LINUX
- Basic knowledge on Open Stack applications.

What is MPI?

Message Passing Interface
MPI is a library specification for message-passing, proposed as a standard by a broadly based committee of vendors, implementors, and users wh ich is designed for high performance on both massively parallel machines and on workstation clusters.

What is Open MPI?

Open MPI is able to combine the expertise, technologies, and resources from all across the High Performance Computing community in order to build the best MPI library available. Open MPI offers advantages for system and software vendors, application developers and computer science researchers.

Password less authentication.**

Log in to the node and generate SSH keys:

```
ssh-keygen
```

The output should be:

```
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
c6:66:93:16:73:0b:bf:46:46:28:7d:a5:38:a3:4d:6d root@node01
The key's randomart image is:
+--[ RSA 2048]----+
|              .   |
|         . + o    |
|        . @ E     |
|         * & .    |
|        . S =     |
|         = + .    |
|           o      |
|          .       |
|                  |
+-----------------+
```

The above command will generate the following files.

```
cd ~/.ssh
ls -la
```

Change the permissions accordingly to the client machines.

```
cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
```

OPEN MPI installation

```
yum install openmpi-devel.x86_64
```

```
sudo yum provides */mpicc
```

Setting the environment variables.

```
yum install environment-modules
module load mpi/openmpi-x86_64
module add mpi/openmpi-x86_64
cd ~/
echo <server IP> > mpi_hosts
```

Verification

Displays the server id:

```
cat mpi_hosts
mkdir /root/samples
cd /root/samples
yum install wget
wget http://svn.open-mpi.org/svn/ompi/tags/v1.6-series/v1.6.4/examples/hello_c.c
mpicc hello_c.c -o hello
mpirun ./hello
```

The output will be:

> Hello, world, I am 0 of 1

To run on multiple machines (considering 2 machines):

```
mpirun -v -np 2 --hostfile ~/mpi_hosts /root/samples/hello
```

The output will be:

> Hello, world, I am 0 of 1
> Hello, world, I am 1 of 1

HPCC - High Performance Computing Cluster:

HPC Challenge is a benchmark suite that measures a range memory access patterns. Development of the benchmark takes place on the HPCC SourceForge page (http://sourceforge.net/projects/hpcc/#_blank). The HPC Challenge benchmark consists of basically 7 tests and the commonly used are Ptrans and HPL.

Installation

This will install all the BLAS and CBLAS which are prerequisites for HPCC and it also downloads HPCC files.

```
sudo yum update
echo "--- Installing gfortran Compiler ---"
sudo yum: install gfortran
echo "--- Downloading BLAS ---"
wget http://www.netlib.org/blas/blas.tgz
echo "--- Extracting BLAS ---"
tar -xvzf blas.tgz
cd BLAS
echo "--- BLAS Compilation Started ---"
echo " "
make all
echo " "
echo "--- BLAS Compilation Finished ---"
cd
echo "--- Downloading CBLAS ---"
wget http://www.netlib.org/blas/blast-forum/cblas.tgz
echo "--- Extracting CBLAS ---"
tar -xvzf cblas.tgz
cd CBLAS
echo "#Updating Makefile.in with the appropriate values"
sed -i "/BLLIB =/ s:/.*:$HOME/BLAS/blas_LINUX.a:" Makefile.in
echo "--- Bulding CBLAS ---"
make all
cd
echo "--- Downloading HPCC --- "
wget http://icl.cs.utk.edu/projectsfiles/hpcc/download/hpcc-1.4.3.tar.gz
echo "--- Extracting HPCC ---"
tar -zxvf hpcc-1.4.3.tar.gz
```

HPCC

This will install all the benchmarks of LINPACK. LINPACK includes HPL, Ptrans, DEGMM, FFT, and stream. Running this file will create an HPCC out file and input text file showing the results.

```
make arch = xxxxxx(file name of make file)
```

Makefile:

```
#
# -- High Performance Computing Linpack Benchmark (HPL)
# HPL - 2.0 - September 10, 2008
# Antoine P. Petitet
# University of Tennessee, Knoxville
# Innovative Computing Laboratory
# (C) Copyright 2000-2008 All Rights Reserved
#
# -- Copyright notice and Licensing terms:
#
# Redistribution and use in source and binary forms, with or without
# modification, are permitted provided that the following conditions
# are met:
```

```
ARCH = $(arch)
#
# ----------------------------------------------------------------------
# - HPL Directory Structure / HPL library ------------------------------
# ----------------------------------------------------------------------
#
TOPdir = ../../..
INCdir = $(TOPdir)/include
BINdir = $(TOPdir)/bin/$(ARCH)
LIBdir = $(TOPdir)/lib/$(ARCH)
#
HPLlib = $(LIBdir)/libhpl.a
#
# ----------------------------------------------------------------------
# - Message Passing library (MPI) --------------------------------------
# ----------------------------------------------------------------------
# MPinc tells the C compiler where to find the Message Passing library
# header files, MPlib is defined to be the name of the library to be
# used. The variable MPdir is only used for defining MPinc and MPlib.
#
MPdir = /usr/lib64/openmpi
MPinc = -I$(MPdir)/include
MPlib = $(MPdir)/lib/libmpi.so
#
# ----------------------------------------------------------------------
# - Linear Algebra library (BLAS or VSIPL) -----------------------------
# ----------------------------------------------------------------------
# LAinc tells the C compiler where to find the Linear Algebra library
# header files, LAlib is defined to be the name of the library to be
# used. The variable LAdir is only used for defining LAinc and LAlib.
#
LAdir = $(HOME)
LAinc =
LAlib = $(LAdir)/CBLAS/lib/cblas\_LINUX.a $(LAdir)/BLAS/blas\_LINUX.a
#
# ----------------------------------------------------------------------
# - F77 / C interface --------------------------------------------------
# ----------------------------------------------------------------------
# You can skip this section if and only if you are not planning to use
# a BLAS library featuring a Fortran 77 interface. Otherwise, it is
# necessary to fill out the F2CDEFS variable with the appropriate
# options. \*\*One and only one\*\* option should be chosen in \*\*each\*\* of
# the 3 following categories:
#
# 1) name space (How C calls a Fortran 77 routine)
#
# -DAdd\_ : all lower case and a suffixed underscore (Suns,
# Intel, ...), \[default\]
# -DNoChange : all lower case (IBM RS6000),
# -DUpCase : all upper case (Cray),
```

```
# -DAdd\_\_ : the FORTRAN compiler in use is f2c.
#
# 2) C and Fortran 77 integer mapping
#
# -DF77\_INTEGER=int : Fortran 77 INTEGER is a C int, \[default\]
# -DF77\_INTEGER=long : Fortran 77 INTEGER is a C long,
# -DF77\_INTEGER=short : Fortran 77 INTEGER is a C short.
#
# 3) Fortran 77 string handling
#
# -DStringSunStyle : The string address is passed at the string loca-
# tion on the stack, and the string length is then
# passed as an F77\_INTEGER after all explicit
# stack arguments, \[default\]
# -DStringStructPtr : The address of a structure is passed by a
# Fortran 77 string, and the structure is of the
# form: struct {char \*cp; F77\_INTEGER len;},
# -DStringStructVal : A structure is passed by value for each Fortran
# 77 string, and the structure is of the form:
# struct {char \*cp; F77\_INTEGER len;},
# -DStringCrayStyle : Special option for Cray machines, which uses
# Cray fcd (fortran character descriptor) for
# interoperation.
#
F2CDEFS =
#
# ----------------------------------------------------------------------
# - HPL includes / libraries / specifics ------------------------------
# ----------------------------------------------------------------------
#
HPL\_INCLUDES = -I$(INCdir) -I$(INCdir)/$(ARCH) $(LAinc) $(MPinc)
HPL\_LIBS = $(HPLlib) $(LAlib) $(MPlib) -lm
#
# - Compile time options ----------------------------------------------
#
# -DHPL\_COPY\_L force the copy of the panel L before bcast;
# -DHPL\_CALL\_CBLAS call the cblas interface;
# -DHPL\_CALL\_VSIPL call the vsip library;
# -DHPL\_DETAILED\_TIMING enable detailed timers;
#
# By default HPL will:
# \*) not copy L before broadcast,
# \*) call the BLAS Fortran 77 interface,
# \*) not display detailed timing information.
#
HPL\_OPTS = -DHPL\_CALL\_CBLAS
#
# ----------------------------------------------------------------------
#
HPL\_DEFS = $(F2CDEFS) $(HPL\_OPTS) $(HPL\_INCLUDES)
```

#

References

http://docs.openstack.org/user-guide/cli_cheat_sheet.html#block-storage-cinder
(http://docs.openstack.org/user-guide/cli_cheat_sheet.html#block-storage-cinder)
http://docs.openstack.org/cli-reference/content/install_clients.html (http://docs.openstack.org/cli-reference/content/install_clients.html)