GOVERNMENT OF INDIA

INDIRA GANDHI CENTRE FOR ATOMIC RESEARCH
KALPAKKAM – 603102

SYSTEM	**HIGH PERFORMACE COMPUTING FACILITIES**
TITLE	**Using Different MPI Implementations in IGCAR NEHA Cluster**

No.	REVISIONS	DATE	APPROVED
A	ORIGINAL ISSUE	✓	

COMPUTER DIVISION	High Performance Computing
COMPUTING SYSTEMS SECTION	IGCAR/CD/CSS/HPC/04

	NAMES	SIGN	DATE
PREPARED	**SUJA RAMACHANDRAN**	Suja	7.5.13
CHECKED	**M.L. JAYALAL**	[illegible]	7.5.13
	R. JEHADEESAN	R. Jehadeesan	7/5/13
REVIEWED	**S. RAJESWARI**	[illegible]	7/5/2013
	K.K. KURIAKOSE	[illegible]	
APPROVED	**S.A.V. SATYA MURTY**	[illegible]	10/5/13

REV	A				

Distribution:

Head CSS,
Head CD,
Director EIG.

SUMMARY

Message Passing Interface (MPI) is a language-independent communications protocol used by parallel programs. In IGCAR HPC clusters, a variety of MPI implementations are available, namely, Intel MPI, Open MPI, MPICH, MVAPICH and LAM/MPI. This document intends to provide a guide on using different MPI implementations in the 134 node NEHA cluster.

Using Different MPI Implementations in IGCAR NEHA Cluster

Message Passing Interface (MPI) is a language-independent communications protocol used by parallel programs. It is a message-passing application programmer interface, together with protocol and semantic specifications for how its features must behave in any implementation. MPI's goals are high performance, scalability and portability. MPI remains the dominant model used in high-performance computing today. MPI implementations support programs written in C, C++ and Fortran.

In IGCAR HPC clusters, a variety of MPI implementations are available, namely, Intel MPI, Open MPI, MPICH, MVAPICH and LAM/MPI. This document intends to provide a guide on using different MPI implementations in the 134 node NEHA cluster.

Compiling and Running MPI Programs

A Sample Program

```
/* C Example */
#include <stdio.h>
#include <mpi.h>

int main ( )
{
  int rank, size;
  char hostname[1024];
  gethostname(hostname,1023);
  MPI_Init (&argc, &argv);   /* starts MPI */
  MPI_Comm_rank (MPI_COMM_WORLD, &rank);          /* get current process id */
  MPI_Comm_size (MPI_COMM_WORLD, &size); /* get number of processes */
  printf( "Hello world from process %d of %d host %s\n", rank, size,hostname );
  MPI_Finalize();
  return 0;
}
```

1) Open MPI

MPI_PATH=/opt/mpi/openmpi-1.4.3-gcc/bin or
MPI_PATH=/opt/mpi/openmpi-1.4.3-intel/bin

The Open MPI Project is an open source MPI-2 implementation that is developed and maintained by a consortium of academic, research, and industry partners. Open MPI provides wrapper compilers for C (mpicc), C++ (mpiCC/mpic++) and Fortran (mpif77, mpif90).

In order to run an MPI program, a 'hostfile' has to be created with the name of hosts in which program is to be run. This filename can be given with the '-machinefile' option while running the program.

hostfile
icn1.hpc.igcar.in
icn2.hpc.igcar.in

Set the environment variables by running
source $MPI_PATH/mpivars.sh

For compiling the program, the command is
$MPI_PATH/mpicc -o test test.c

Then the program can be run using
$MPI_PATH/mpirun -n 4 -machinefile ./hostfile ./test
where '4' indicates number of copies of the program that has to be run.

2) MVAPICH2

MPI_PATH=/opt/mpi/mvapich2-1.6-gcc/bin or
MPI_PATH=/opt/mpi/mvapich2-1.6-intel/bin

MVAPICH2 is an MPI-2 implementation based on MPICH2 and MVICH. MVAPICH2 provides many features including RDMA_CM support, iWARP support, optimized collectives, on-demand connection management, multi-core optimized and scalable shared memory support. MVAPICH2 has compilers for C (mpicc), C++ (mpiCC/mpic++) and Fortran (mpif77, mpif90).

Set the environment variables by running
source $MPI_PATH/mpivars.sh

A 'hostfile' has to be prepared with the name of hosts as in the case of Open MPI. This filename can be given with the '-hostfile' option while running the program.

For compiling the program, the command is
$MPI_PATH/mpicc -o test test.c

The program can be run as

$MPI_PATH/mpirun_rsh -ssh -np 16 -hostfile ./hostfile ./test
$MPI_PATH/mpirun_rsh -np 4 n1 n2 n1 n2 ./test
where the argument 'np' is the number of copies of the program to be run and n1, n2 are nodes in which the program has to run.

3) LAM/ MPI

MPI_PATH=/opt/mpi/lam-7.1.4-gcc/bin/ or
MPI_PATH=/opt/mpi/lam-7.1.4-intel/bin

LAM/MPI (Local Area Multicomputer) is a high-quality open-source implementation of the MPI specification, including all of MPI-1.2 and much of MPI-2. LAM/ MPI has compilers for C (mpicc), C++ (mpiCC) and Fortran (mpif77).

Set the environment variables by running
source $MPI_PATH/mpivars.sh

The LAM program can be compiled using
$MPI_PATH/mpicc -o test test.c

The user can create a hostfile called 'lamhosts' with the names of hosts in which the program has to be run.
lamhosts
icn1.hpc.igcar.in

icn2.hpc.igcar.in

Each machine will be given a node identifier (nodeid) starting with 0 for the first listed machine, 1 for the second, etc. The lamboot tool can be started on the hosts using
$MPI_PATH/lamboot -v lamhosts

The name of all lam hosts can be viewed using the command
$MPI_PATH/lamnodes

Finally, to run 4 number of processes,
$MPI_PATH/mpirun −np 4 ./test

For removing all user processes and messages,
$MPI_PATH/lamclean -v

For terminating LAM,
$MPI_PATH/lamhalt

4) MPICH

MPI_PATH=/opt/mpi/mpich-1.2.7p1-gcc/bin/ or
MPI_PATH=/opt/mpi/mpich-1.2.7p1-intel/bin/

MPICH is a high performance and widely portable implementation of the Message Passing Interface (MPI) standard. The goal of MPICH is to provide an MPI implementation that efficiently supports different computation and communication platforms, high-speed networks and proprietary high-end computing systems. MPICH provides compilers for C (mpicc), C++ (mpicxx) and Fortran (mpif77/mpif90).

Set the environment variables by running
source $MPI_PATH/mpivars.sh

The program can be compiled using
$MPI_PATH/mpicc -o test test.c

A hostfile has to be prepared with the name of hosts as in the case of Open MPI. This filename can be given with the '-machinefile' option while running the program. Finally to run 4 copies of the process,
$MPI_PATH/mpirun -np 4 -machinefile ./hostfile ./test

5) MPICH2

MPI_PATH=/opt/mpi/mpich2-1.3.2p1-gcc/bin/ or
MPI_PATH=/opt/mpi/mpich2-1.3.2p1-intel/bin/

MPICH2 is a high-performance and widely portable implementation of the MPI Standard, designed to implement all of MPI-1 and MPI-2 (including dynamic process management, one-sided operations, parallel I/O and other extensions). MPICH2 provides compilers for C (mpicc), C++ (mpicxx) and Fortran (mpif77/mpif90). The default runtime environment in MPICH2 is called Hydra.

Set the environment variables by running

source $MPI_PATH/mpivars.sh

Create a file called mpd.hosts with node names and number of processes to be run on each node.
icn1.hpc.igcar.in:2
icn2.hpc.igcar.in:2

Start mpdboot on the nodes with the number of nodes as the '-n' argument.
$MPI_PATH/mpdboot -n 3 -f mpd.hosts

The name of all nodes running mpd can be viewed using the command
$MPI_PATH/mpdtrace

The program can be compiled using
$MPI_PATH/mpicc -o test test.c

To run 4 copies of the process,
$MPI_PATH/mpiexec -n 4 ./test

Finally to exit mpd,
$MPI_PATH/mpdallexit

6) Intel MPI

MPI_PATH=/opt/intel/impi/4.0.2.003/intel64/bin/

Intel MPI Library 4.1 focuses on making applications perform better on Intel architecture-based clusters, implementing the high performance MPI - 2.2 specifications on multiple fabrics.

First, set the environment variables by running
source $MPI_PATH/mpivars.sh

The program can be compiled using
$MPI_PATH/mpicc -o test test.c

A hostfile can be created with node names and number of processes to be run on each node. This filename can be given with the '-f' option while running the program.
icn1.hpc.igcar.in:2
icn2.hpc.igcar.in:2

or *mpd.hosts file as*
icn1.hpc.igcar.in:2
icn2.hpc.igcar.in:2

To run 4 copies of the program,
$MPI_PATH/mpirun -np 4 -f ./hostfile ./test or
$MPI_PATH/mpirun -np 4 ./test (with mpd.hosts file)

Selecting an MPI Implementation

The '*mpi-selector*' is a simplistic tool to select one of the multiple MPI implementations. It allows system administrators to set a site-wide default MPI implementation while also allowing users to set their own default MPI implementation. The system is having a system-wide shell startup file that looks first at the user's MPI preferences. If found, the MPI implementation

indicated by the user's preferences is setup in the current environment. If not found, system looks for a site-wide default.

The *mpi-selector* command provides four main actions: 1) List which MPI implementations are available 2) Set a default (either on a per-user or site-wide basis) 3) Unset a default (either on a per-user or site-wide basis) and 4) Query what the current default is

'mpi-selector-menu' is a menu-based wrapper around the mpi-selector command. The user can issue the *'mpi-selector-menu'* command and select a MPI implementation as shown in the screenshot.

```
File  Edit  View  Search  Terminal  Help
[suja@hn1 ~]$ mpi-selector-menu
Current system default: mpich2-1.3.2p1-icc
Current user default:   openmpi-1.4.3-gcc

    "u" and "s" modifiers can be added to numeric and "U"
    commands to specify "user" or "system-wide".

1. impi-4.0.2.003
2. lam-7.1.4-gcc
3. lam-7.1.4-intel
4. mpich-1.2.7p1-gcc
5. mpich-1.2.7p1-intel
6. mpich2-1.3.2p1-gcc
7. mpich2-1.3.2p1-intel
8. mvapich-1.2rc1-gcc
9. mvapich-1.2rc1-intel
10. mvapich2-1.6-gcc
11. mvapich2-1.6-intel
12. openmpi-1.4.3-gcc
13. openmpi-1.4.3-intel
U. Unset default
Q. Quit

Selection (1-13[us], U[us], Q): 12
Operator on the per-user or system-wide default (u/s)? u
Defaults already exist; overwrite them? (y/N) y
```

Once an MPI implementation is selected and a new terminal is opened, the selected implementation will become the user's default. This can be ensured by issuing the "*which mpirun*" and "*which mpicc*" commands.