HP-SEE

MPI libraries on PARADOX

www.hp-see.eu

Dusan Stankovic
Scinetific Computer Laboratory
Institute of Physics Belgrade
sdule@ipb.ac.rs

HP-SEE
High-Performance Computing Infrastructure
for South East Europe's Research Communities

The HP-SEE initiative is co-funded by the European Commission under the FP7 Research Infrastructures contract 261499

MPI introduction

- MPI is a standard, not an implementation. It defines what an MPI library should be.
- First version completed in early 1990s.
- Reasons to use MPI:
  - Standardization – supported on all HPC platforms
  - Portability – library takes care of serialization
  - Functionality – more than a 115 routines in MPI-1 alone
  - Availability – lots of implementations available, both vendor and open source
- MPI-1 and MPI-2 versions available, work on MPI-3 draft is in progress

MPI programming model [1/2]

- Originally targeted for distributed memory systems
- No shared variables, all parallelism explicit, data allocation and movement responsibility of the programmer
- Somewhat harder to learn (compared to OpenMP for example), allows for great customization
- Implementations for shared memory and hybrid architectures exist, don't degrade performance. Aware of the hardware beneath – e.g. will use shared memory for messages if CPUs physically share memory
- Library accompanied by middleware, it maps logical organization to physical

MPI programming model [2/2]

- Groups of processes and communicators allow message passing between processes.
- By default, all processes included in MPI_COMM_WORLD
- Allows creation of new communicators or even virtual topologies
- Message can be point-to-point or collective, with or without data, blocking or non-blocking etc.
- Bindings exist for C/C++ and Fortran
- Also there are custom bindings that better suit OOP model in C++ (like Boost.MPI)
- Note – in latest versions of the MPI-2 standard, C++ bindings are considered deprecated

An MPI use case

- Typical MPI use case scenario:
  - Initially divide data between processes
  - Each process does some computation on its local data
  - Neighbors exchange data for bordering regions
  - Processes update their data using exchanged information
- Important to reduce the amount of data for communication as much as possible and to avoid unnecessary waiting to obtain the best performance.

Available implementations on PARADOX

- On PARADOX cluster, there are multiple MPI implementations available:
  - Mpich (MPI-1)
  - Mpich2 (MPI-2)
  - OpenMPI (MPI-2)
- MPI-2 specification included:
  - One-sided communication
  - Dynamic process management
  - I/O

MPICH [1/2]

- Located in **`/opt/mpich-1.2.7p1`**
- Or you can use environment variable **`MPI_MPICH_PATH`**
- Last updated in 1995
- Use `mpicc` to compile and link programs
- `mpicc -compile-info` shows how exactly the underlying compiler is invoked
- `mpicc -cc=gcc` (or `-cc=icc`) overrides the default compiler setting

MPICH [2/2]

- Use **`mpirun`** to execute programs
- **`mpirun -np 2 -machinefile my_machine_file test`**
- Or use existing PBS batch system to schedule and start jobs
- Predefined environment variable **`MPI_MPICH_MPIEXEC`**
- An example:

```
${MPI_MPICH_PATH}/bin/mpicc -o test test.c
${MPI_MPICH_MPIEXEC} test
```

MPICH2 [1/2]

- Located in **`/opt/mpich2-1.1.1p1`**
- Or you can use environment variable **`MPI_MPICH2_PATH`**
- An up-to-date implementation
- Use `mpicc` to compile and link programs
- `mpicc -compile-info` shows how exactly the underlying compiler is invoked
- `mpicc -cc=gcc` (or `-cc=icc`) overrides the default compiler setting
- The same can be done with `export MPICH_CC=gcc`

MPICH2 [2/2]

- Use **`mpirun`** to execute programs
- **`mpirun -np 2 -machinefile my_machine_file test`**
- Or use existing PBS batch system to schedule and start jobs
- Predefined environment variable **`MPI_MPICH2_MPIEXEC`**
- An example:

```
${MPI_MPICH2_PATH}/bin/mpicc -o test test.c
${MPI_MPICH2_MPIEXEC} test
```

OpenMPI [1/2]

- Located in **`/opt/openmpi-1.2.5`**
- Or you can use environment variable **`MPI_OPENMPI_PATH`**
- Widespread, used by top supercomputers in the world
- Use `mpicc` to compile and link programs
- `mpicc -show` shows how exactly the underlying compiler is invoked
- Change default compilers by setting environment variables
  - **`OMPI_CC`** for C compiler
  - **`OMPI_CXX`** for C++ compiler
  - **`OMPI_FC`** for Fortran 90 compiler

OpenMPI [2/2]

- On PARADOX, applications can be ran by using `mpirun` script or PBS scheduling system
- When using `mpirun`, some useful arguments are:
  - **`-np X`**, to run X MPI processes
  - **`-hostfile my_hostfile`**, to specify on which hosts to run
  - **`-npernode X`**, to run X processes on each specified node
  - **`-display-map`**, to display mapping of processes to hosts
- Hostfile should contain addresses of hosts, an example:

```
int1.ipb.ac.rs int2.ipb.ac.rs
```

PBS complete example

- Download the archive `Openmpi.tgz` from: **http://wiki.ipb.ac.rs/index.php/Openmpi**
- Unzip with **`tar xf Openmpi.tgz ; cd openmpi`**
- Compile with **`$MPI_OPENMPI_PATH/bin/mpicc -o job job.c`** (or use provided Makefile)
- Submit a .pbs job script
- For MPICH and MPICH2 just modify correspoding environment variables
- MPICH2 is backwards compatible with MPICH, but the other way around doesn't work
- Example: compile with MPICH, run with MPICH2

Conclusion

- There are different libraries available on PARADOX, if not sure, use OpenMPI
- There are different ways to compile/run jobs, user interface has the same architecture as nodes on the cluster
- For more details about using batch system on PARADOX, please consult **`http://wiki.ipb.ac.rs/index.php/PBS_examples`**
- If you run into problems, there are tools to help you debug or profile an application:
  - gdb (not so straightforward to setup)
  - TotalView (native MPI/OpenMP support)
  - Scalasca, TAU

References

- https://computing.llnl.gov/tutorials/mpi/
- http://www.mcs.anl.gov/research/projects/mpich2/
- http://www.open-mpi.org/
- http://wiki.ipb.ac.rs/index.php/PBS_examples