UBDA Platform

Example for C++ with OpenMPI

User Guide

Version 1.0

16 July 2018

Revision History

Version	Date	Prepared By	Summary of Changes
1.0	Jul 16, 2018		Initial release

Table of Contents

1. Introduction

This document is shown a C++ example by using OpenMPI method running on the UBDA platform.

Note: User should first register a user account through UBDA website at: https://www.polyu.edu.hk/pfs/index.php/177729 to access the UBDA Platform

2. Perform the test

2.1 Login to ubdaplatform.polyu.edu.hk via SSH

2.2 Create the testing directory

```
$ mkdir -p $HOME/CPP_test
$ cd $HOME/CPP_test
```

2.3 Prepare the C++ program (Filename: hello_mpi.cpp)

```
# include <cstdlib>
# include <ctime>
# include <iomanip>
# include <iostream>
# include <mpi.h>

using namespace std;

int main ( int argc, char *argv[] );
void timestamp ( );

//****************************************************************
************80

int main ( int argc, char *argv[] )

//****************************************************************
************80
//
//  Purpose:
//
//    MAIN is the main program for HELLO_MPI.
//
//  Discussion:
//
//    This is a simple MPI test program.
//    Each process prints out a "Hello, world!" message.
//    The master process also prints out a short message.
//
//    Modified to use the C MPI bindings, 14 June 2016.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    14 June 2016
//
//  Author:
```

```
//
//    John Burkardt
//
//  Reference:
//
//    William Gropp, Ewing Lusk, Anthony Skjellum,
//    Using MPI: Portable Parallel Programming with the
//    Message-Passing Interface,
//    Second Edition,
//    MIT Press, 1999,
//    ISBN: 0262571323,
//    LC: QA76.642.G76.
//
{
  int id;
  int ierr;
  int p;
  double wtime;
//
//  Initialize MPI.
//
  ierr = MPI_Init ( &argc, &argv );
//
//  Get the number of processes.
//
  ierr = MPI_Comm_size ( MPI_COMM_WORLD, &p );
//
//  Get the individual process ID.
//
  ierr = MPI_Comm_rank ( MPI_COMM_WORLD, &id );
//
//  Process 0 prints an introductory message.
//
  if ( id == 0 )
  {
    timestamp ( );
    cout << "\n";
    cout << "HELLO_MPI - Master process:\n";
    cout << "  C++/MPI version\n";
    cout << "  An MPI example program.\n";
    cout << "\n";
    cout << "  The number of processes is " << p << "\n";
    cout << "\n";
  }
//
//  Every process prints a hello
//
  if ( id == 0 )
  {
    wtime = MPI_Wtime ( );
  }
  cout << "  Process " << id << " says 'Hello, world!'\n";
//
//  Process 0 says goodbye.
//
```

```
  if ( id == 0 )
  {
    wtime = MPI_Wtime ( ) - wtime;
    cout << "  Elapsed wall clock time = " << wtime << "
seconds.\n";
  }
//
//  Terminate MPI.
//
  MPI_Finalize ( );
//
//  Terminate.
//
  if ( id == 0 )
  {
    cout << "\n";
    cout << "HELLO_MPI:\n";
    cout << "  Normal end of execution.\n";
    cout << "\n";
    timestamp ( );
  }
  return 0;
}
//****************************************************************
************80

void timestamp ( )

# define TIME_SIZE 40

  static char time_buffer[TIME_SIZE];
  const struct std::tm *tm_ptr;
  size_t len;
  std::time_t now;

  now = std::time ( NULL );
  tm_ptr = std::localtime ( &now );

  len = std::strftime ( time_buffer, TIME_SIZE,
"%d %B %Y %I:%M:%S %p", tm_ptr );

  std::cout << time_buffer << "\n";

  return;
# undef TIME_SIZE
}
```

2.4 Load the MPI user environment. You can select OpenMPI with gcc.

For select OpenMPI, use the following:

```
$ module available
----------------- /ubda/apps/modules/modules-
4.1.2/modulefiles -----------------
anaconda3-5.2.0    cuda-9.1                  perl-5.26.1
ansys-17.2-fluent  fsl-5.0.11                python-2.7.15
atlas-3.10.3       gcc-5.5.0                 python-3.6.6
autodock-4.2.6     ips_xe_2018_u3            quantum-
espresso-6.2.1
cmake-3.12.1       mpich-3.2.1-gcc-5.5.0     R-3.5.0
cuda-9.0           openmpi-3.0.1-gcc-5.5.0   singularity-
2.6.0

$ module load openmpi-3.0.1-gcc-5.5.0
$ module list
Currently Loaded Modulefiles:
     1) openmpi-3.0.1-gcc-5.5.0
$ which mpicc
/ubda/apps/openmpi/openmpi-3.0.1-gcc-5.5.0/bin/mpicc
```

2.5 Build the executable binary by using the selected MPI

```
$ mpicc -o helloworldopenmpi.out hello_mpi.cpp
$ file helloworldopenmpi.out
helloworldopenmpi.out: ELF 64-bit LSB executable, x86-64, version
1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux
2.6.32, not stripped
```

2.6 Prepare the job script file.

For select OpenMPI, use the following (Filename: helloworld.pbs)

```
#!/bin/sh
#PBS -N cpptest
#PBS -l nodes=1:ppn=8
#PBS -l walltime=12:00:00
#PBS -q q2s01
#PBS -V
#PBS -S /bin/bash

module load openmpi-3.0.1-gcc-5.5.0

EXEC=helloworldopenmpi.out
############################################################
NP=`cat $PBS_NODEFILE | wc -l`
NN=`cat $PBS_NODEFILE | sort | uniq | tee /tmp/nodes.$$ | wc -l`
cat $PBS_NODEFILE > /tmp/nodefile.$$
echo "process will start at : "
date
echo "++++++++++++++++++++++++++++++++++++++++"
cd $PBS_O_WORKDIR

mpirun -n $NP --mca blt self,openib $EXEC > result.out


echo "++++++++++++++++++++++++++++++++++++++++"
echo "processs will sleep 5 minutes"
#sleep 300
echo "process end at : "
date
rm -f /tmp/nodefile.$$
rm -f /tmp/nodes.$$
module unload openmpi-3.0.1-gcc-5.5.0
```

Please remind to change the values for your application.

```
#PBS -N cpptest {your job name}
#PBS -l nodes=1:ppn=8 {your requested resource;nodes and
processors per node}
#PBS -q q2s01 {the job queue}
EXEC=helloworld.out {your execute file name}
```

Example files can be found at:

/ubda/apps/examples/CPP/helloworld/

hello_mpi.cpp
helloworld.pbs

3 Job submission

3.1 Submit the script (***helloworld.pbs***) to job queue.
A job ID number will be returned.

```
$ qsub helloworld.pbs
1896.ubda-mgt01
```

3.2 Enquiry the submitted job status.

```
$ qstat -na

ubda-mgt01:
                                                                  Req'd       Req'd       Elap
Job ID                  Username    Queue    Jobname          SessID  NDS   TSK   Memory      Time    S   Time
----------------------- ----------- -------- ---------------- ------ ----- ------ --------- --------- - ---------
1896.ubda-mgt01         ubdademo9   q2s01    cpptest          219882     1      8       --   12:00:00 R  00:00:01
   ubda-d050/0-7
```

Job status field name	Explanation	Example
JOB ID	Unique Job id.	1896.ubda-mgt01
Job name	Name for the job allocation	cpptest
Queue name	Name of the job queue that the job has assigned.	q2s01
Username	Your NetID	ubdademo9
S	Job current status. Q = queued R = Job is executing C = Job was completed	Running
ELAP TIME	Time for the job executed	00:00:01 (1 second)
Req'd Time	The maximum execution time	12:00:00 (1 day)
NDS	Total number of nodes assigned	1
TSK	Total number of cores assigned	8

4 How to check the result

4.1 Check for any error messages for JobID

```
$ more cpptest.e1896
Host key verification failed.(know error message can be ignored)

$ more cpptest.o1896
process will start at :
Wed Jul 11 15:52:36 HKT 2018
++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++
process end at :
Wed Jul 11 15:52:37 HKT 2018
```

4.2 Check for result

```
$ cat result.out
  Process 1 says 'Hello, world!'
  Process 2 says 'Hello, world!'
  Process 3 says 'Hello, world!'
  Process 4 says 'Hello, world!'
  Process 5 says 'Hello, world!'
  Process 6 says 'Hello, world!'
  Process 7 says 'Hello, world!'
11 July 2018 03:52:37 PM

HELLO_MPI - Master process:
  C++/MPI version
  An MPI example program.

  The number of processes is 8

  Process 0 says 'Hello, world!'
  Elapsed wall clock time = 4.46801e-06 seconds.

HELLO_MPI:
  Normal end of execution.

11 July 2018 03:52:37 PM
```

5 Useful Reference

- Tutorial of C++
  URL: http://www.cplusplus.com/doc/tutorial/

- Command reference for qstat
  URL: http://docs.adaptivecomputing.com/torque/6-0-0/help.htm#topics/torque/commands/qstat.htm?Highlight=qstat