Hybrid (MPI+OpenMP) Codes on the FASRC cluster

Contents [hide]

Introduction

This page will help you compile and run hybrid (MPI+OpenMP) applications on the cluster. Currently we have both OpenMPI and Mvapich2 MPI libraries available, compiled with both Intel and GNU compiler suits.

Example Code

Below are simple hybrid example codes in Fortran 90 and C++.
Fortran 90:

```
!=========================================================
! Program: hybrid_test.f90 (MPI + OpenMP)
!          FORTRAN 90 example - program prints out
!          rank of each MPI process and OMP thread ID
!=========================================================
program hybrid_test
```

```
  implicit none
  include "mpif.h"
  integer(4) :: ierr
  integer(4) :: iproc
  integer(4) :: nproc
  integer(4) :: icomm
  integer(4) :: i
  integer(4) :: j
  integer(4) :: nthreads
  integer(4) :: tid
  integer(4) :: omp_get_num_threads
  integer(4) :: omp_get_thread_num
  call MPI_INIT(ierr)
  icomm = MPI_COMM_WORLD
  call MPI_COMM_SIZE(icomm,nproc,ierr)
  call MPI_COMM_RANK(icomm,iproc,ierr)
!$omp parallel private( tid )
  tid = omp_get_thread_num()
  nthreads = omp_get_num_threads()
  do i = 0, nproc-1
     call MPI_BARRIER(icomm,ierr)
     do j = 0, nthreads-1
        !$omp barrier
        if ( iproc == i .and. tid == j ) then
           write (6,*) "MPI rank:", iproc, " with thread ID:", tid
        end if
     end do
  end do
!$omp end parallel
  call MPI_FINALIZE(ierr)
  stop
end program hybrid_test
```

C++:

```
//============================================================
//  Program: hybrid_test.cpp (MPI + OpenMP)
//           C++ example - program prints out
//           rank of each MPI process and OMP thread ID
//============================================================
#include <iostream>
#include <mpi.h>
#include <omp.h>
using namespace std;
```

```
int main(int argc, char** argv){
  int iproc;
  int nproc;
  int i;
  int j;
  int nthreads;
  int tid;
  int provided;
  MPI_Init_thread(&argc,&argv, MPI_THREAD_MULTIPLE, &provided);
  MPI_Comm_rank(MPI_COMM_WORLD,&iproc);
  MPI_Comm_size(MPI_COMM_WORLD,&nproc);
#pragma omp parallel private( tid )
  {
    tid = omp_get_thread_num();
    nthreads = omp_get_num_threads();
    for ( i = 0; i <= nproc - 1; i++ ){
      MPI_Barrier(MPI_COMM_WORLD);
      for ( j = 0; j <= nthreads - 1; j++ ){
        if ( (i == iproc) && (j == tid) ){
          cout << "MPI rank: " << iproc << " with thread ID: " << tid << endl;
        }
      }
    }
  }
  MPI_Finalize();
  return 0;
}
```

Compiling the program

```
MPI Intel, Fortran 90: [username@rclogin02 ~]$ mpif90 -o hybrid_test.x hybrid_test.f90 -o
MPI Intel, C++:        [username@rclogin02 ~]$ mpicxx -o hybrid_test.x hybrid_test.cpp -o
MPI GNU, Fortran 90:   [username@rclogin02 ~]$ mpif90 -o hybrid_test.x hybrid_test.f90 -f
MPI GNU, C++:          [username@rclogin02 ~]$ mpicxx -o hybrid_test.x hybrid_test.cpp -f
```

Running the program

You could use the following SLURM batch-job submission script to submit the job to the queue:

```
#!/bin/bash
#SBATCH -J hybrid_test
#SBATCH -o hybrid_test.out
```

```
#SBATCH -e hybrid_test.err
#SBATCH -p shared
#SBATCH -n 2
#SBATCH -c 4
#SBATCH -t 180
#SBATCH --mem-per-cpu=4000
export OMP_NUM_THREADS=4
srun -n 2 --cpus-per-task=4 --mpi=pmix ./hybrid_test.x
```

The OMP_NUM_THREADS environmental variable is used to set the number of threads to the desired number. **Please notice that this job will use 2 MPI processes (set with the `-n` option) and 4 OpenMP threads per MPI process (set with the `-c` option), so overall the job will reserve and use 8 compute cores.** If you name the above script `omp_test.batch`, for instance, the job is submitted to the queue with

```
sbatch omp_test.batch
```

Upon job completion, job output will be located in the file `hybrid_test.out` with the contents:

```
 MPI rank:           0  with thread ID:           0
 MPI rank:           0  with thread ID:           1
 MPI rank:           0  with thread ID:           2
 MPI rank:           0  with thread ID:           3
 MPI rank:           1  with thread ID:           0
 MPI rank:           1  with thread ID:           1
 MPI rank:           1  with thread ID:           2
 MPI rank:           1  with thread ID:           3
```

Last Updated November 15, 2019

Was this article helpful? ☆ ☆ ☆ ☆ ☆