

Hands-on: MPI call in thread parallel region

- Choose either C/C++ (`c`) or Fortran (`fortran`) samples. Both of them are fine, as well.

C/C++

How to compile and how to execute

1. Compile program

- We have five kinds of kernels. In hands-on, we mainly focus on `sendrecv` and `thread_nt2`.
 - The original serial code is put in `serial`. The others are the MPI-parallel version with different communication patterns.
- The executable file (`run.x`) is generated in each directory.

```
$ cd c/sendrecv
$ make # make -f Makefile.own # if using own compiler
$ ls
run.x
$ cd c/thread_nt2
$ make # make -f Makefile.own # if using own compiler
$ ls
run.x
```

2. Run program

- A job script (`task.sh`) to execute the program is located in `results/` of each kernel directory.
- You can run the program either:

```
## To run as a batch job
$ cd c/sendrecv/results
$ pjsub task.sh
$ cd c/thread_nt2/results
$ pjsub task.sh
## Or, to run in an interactive job
$ cd c/sendrecv/results
$ bash task.sh
$ cd c/thread_nt2/results
$ bash task.sh
```

- The jobs in the Exercises will be completed within 1-2 minutes in 12 nodes with process-per-node=4.
 - For safety, we set the elapsed time of the job scripts as 4 minutes.
 - One exceptional case is `serial`. It will takes about 2 hours.

Exercises A

- E1: Check whether the thread level is acceptable in `thread_nt2/main.c` from the viewpoints of Fujitsu MPI.
- E2: Compare the MPI communication part (e.g., part indicated by `/* p2p communication along x-direction */`) of `sendrecv/main.c` to that of `thread_nt2/main.c`.

Exercises B (advanced)

- E3: Try to do performance analyses in `thread_nt2/results.fipp` (sampling with `fipp`) and in `sendrecv.fapp` (simple profiling with `fapp`). After the measurement, you simply type `bash perf.sh`. Then, you can obtain the resultant text data of the profiler.

Fortran

How to compile and how to execute

1. Compile program

- We have five kinds of kernels. In hands-on, we mainly focus on `sendrecv` and `thread_nt2`.
 - The original serial code is put in `serial`. The others are the MPI-parallel version with different communication patterns.
- The executable file (`run.x`) is generated in each directory.

```
$ cd fortran/sendrecv
$ make # make -f Makefile.own # if using own compiler
$ ls
run.x
$ cd fortran/thread_nt2
$ make # make -f Makefile.own # if using own compiler
$ ls
run.x
```

2. Run program

- A job script (`task.sh`) to execute the program is located in `results/` of each kernel directory.
- You can run the program either:

```
## To run as a batch job
$ cd fortran/sendrecv/results
$ pjsub task.sh
$ cd fortran/thread_nt2/results
$ pjsub task.sh
## Or, to run in an interactive job
$ cd fortran/sendrecv/results
$ bash task.sh
$ cd fortran/thread_nt2/results
$ bash task.sh
```

- The hobs in the Exercises will be completed within 1-2 minutes in 12 nodes with process-per-node=4. For safety, we set the elapsed time of the job scripts as 4 minutes.
 - One exceptional case is `serial`. It will takes about 2 hours.

Exercises A

- E1: Check whether the thread level is acceptable in `thread_nt2/main.F90` from the viewpoints of Fujitsu MPI.
- E2: Compare the MPI communication part (e.g., part indicated by `! p2p communication along x-direction`) of `sendrecv/main.F90` to that of `thread_nt2/main.F90`.

Exercises B (advanced)

- E3: Try to do performance analyses in `thread_nt2/results.fipp` (sampling with `fipp`) and in `sendrecv.fapp` (simple profiling with `fapp`). After the measurement, you simply type `bash perf.sh`. Then, you can obtain the resultant text data of the profiler.