Build and Run of Parallel Applications

(Last update: May 9, 2024: add notes about --with-pbs configure option of mvapich, --disable-cuda is removed)

OpenMP parallel

You just need to prepare OpenMP-enabled binary and specify number of OpenMP threads with ompthreads= in the jobscript. (You can use OMP_NUM_THREADS environment variable instead. However, in the MPI-OpenMP hybrid programs, something unexpected can happen depending on the MPI library.)

In case of OpenMP enabled libraries such as MKL or OpenBLAS, you just need to set ompthreads= in the header part of the jobscript. If you can change number of threads for MKL or OpenBLAS part only, you can use environment variable MKL_NUM_THREADS or OPENBLAS_NUM_THREADS, respectively.

If you met error like "libiomp5 not found", you need to load Intel oneAPI runtime environment before running program. (You should also check output of "ldd (binary file path)" command.) In case of unexpected error only when OpenMP threads are enabled, you might be loading a wrong OpenMP library. Try unloading (or purge) modules you are not using.

Non-OpenMP parallel

Programs using non-OpenMP parallelism does not need any special setting in the jobscript. "ompthreads=" specificiation do nothing for non-OpenMP threads. Either of "ompthreads=1" or "omptreads=(number of threads)" is OK. This specification does not change anything as long as OpenMP parallelism is not involved. (You still need to specify ncpus= value correctly.) If you need to specify the number of (non-OpenMP) threads before running program, please add those lines in the jobscript. Note that the value of "ompthreads=" is not concerned with CPU points of jobs, since the CPU points are determined from number of CPU cores, number of GPUs, and elapsed time.

If you are using OpenMP-enabled libraries (such as OpenBLAS and MKL) in some part of your job, you may need to choose the number of threads carefully.

MPI parallel

Use library provided by RCCS

Modules for Intel MPI, Open MPI, and MVAPICH2 are available. If you want to use Intel Compilers in combination with MPI library, please install them in your home directory first. Some useful information about Intel Compilers might be available in this page. If you are trying to use GCC and Intel MPI, you don't need to install oneAPI toolkits.

example: load MPI module (Open MPI, MVAPICH, and Intel MPI) with gcc11 (don't type leading $.)

```
$ module load openmpi/4.1.6/gcc11
```

```
$ module load mvapich/2.3.7/gcc11
```

```
$ module load intelmpi/2021.11
```

NOTE: for Intel MPI case, you don't need to specify the version of GCC. However, in case of fortran, MPI module (mpi.mod) for the specified version of gfortran might not exist. Please use newer version of MPI or change the GCC version in this case.

Once you successfully loaded the module, commands like mpicc, mpicxx, and mpif90 become available. If you want to use AOCC, please load the corresponding build of MPI library.

```
$ module load aocc/4.2.0
$ module load openmpi/4.1.6/aocc4.1
```

Build your own library

- Open MPI => see below
- Intel MPI (without Intel Compiler) => please install oneAPI HPC Toolkit
    - You can use Intel MPI installed by RCCS as described above.
- Intel Compiler + Intel MPI => you need to install oneAPI Base and HPC Toolkits in your directory.
    - Please also check this page for installation of Intel compilers.
- MVAPICH => you don't need special setting. Specify compilers with CC, CXX, FC, F77 variables and run "configure && make && make install". Normally it works. You can ensure the validity and safety of the build by running "make check" after the build.
    - Tighter integration with PBS can be achieved by adding --with-pbs=/apl/pbs/22.05.11 to "configure".
        - ($PBS_NODEFILE is taken into account even when --with-pbs option is not specified.)
    - You can add --enable-fortran --enable-cxx upon configure.

After the successful installation (and path settings in your configuration file such as .bash_profile), you will try to build the application using mpicc, mpicxx, mpif90 etc. MPI libraries/environments other than Intel MPI, Open MPI, MVAPICH are not tested.

Build of Open MPI library

You may need to specify compilers via CC, CXX, FC variables. This is the same as the case of other libraries. We recommend you to add the following option at the configure step.

```
--with-tm=/apl/pbs/22.05.11
```

This option is necessary to collaborate with the queuing system. You can omit machine file and other environment variable settings upon running application by specifying this option. This option is always enabled when we build Open MPI. (e.g. Open MPI 4.1.6) You can also add --enable-mpi-cxx and --enable-mpi1-compatibility in configure option.

If you need CUDA-aware Open MPI, load the cuda module (e.g. cuda 12.2 update 2) before running configure and add the following option. The other options should be the same as the normal CPU version.

```
--with-cuda=/apl/cuda/12.2u2
```

Please use ccgpu for the test of MPI-parallel GPU applications. Two GPU cards are available in ccgpu. You can login to ccgpu from ccfep with "ssh ccgpu" command.

If you build Open MPI with compilers of NVIDIA HPC SDK, you may need to add -fPIC option to CFLAGS (--with-tm=/apl/pbs/22.05.11 is also recommended on configure). CUDA in the SDK can be specified like --with-cuda=/apl/nvhpc/23.9/Linux_x86_64/23.9/cuda.

Notices upon running job

You need to load MPI settings (PATH, LD_LIBRARY_PATH etc.) in your jobscript unless those settings are defined in .bash_profile etc. List of node names is provided by the queuing system via the file specified with PBS_NODEFILE environment variable. If error message like "mpirun: command not found" is shown, MPI environment may not be correctly loaded. Please check what is happening. Removing -s option from module command may be helpful, since the error message from module command is discarded if -s exists.

If MPI environment has the support for the queuing system, just loading MPI library and running program via mpirun command are enough. The MPI environment automatically finds the hostfile and uses it.

```
#!/bin/sh
#PBS -l select=1:ncpus=32:mpiprocs=16:ompthreads=2
#PBS -l walltime=24:00:00

cd ${PBS_O_WORKDIR}
# list of nodes given by queuing system
cat ${PBS_NODEFILE}

module -s purge
module -s load openmpi/4.1.6/gcc11

MYPROG=/home/users/${USER}/bin/myprog-mpi

# mpirun command from openmpi/4.1.6/gcc11 package
mpirun -np 16 ${MYPROG} input.inp > output.out
```

If your MPI environment does not support the queuing system, you need to provide machinefile (or hostfile) manually

(please use the file specified in PBS_NODEFILE environment variable). In case of Open MPI without queuing system support, you need to add --hostfile ${PBS_NODEFILE} to options of mpirun.