Compilation of a Sequential Program and its Execution with OpenMP and MPI

MOOC : Dopez vos calculs

1. Compilation of a sequential C++ program
2. Its parallel behavior with OpenMP
3. Its parallel behavior with MPI
4. Analysis of the differences

```
$ vim helloworld.cpp
```

```
#include <iostream>
using namespace std ;

int main()
{
  cout << "hello world" << endl;
  return 0 ;
}
```

Compilation of the helloworld program

Localy

```
g++ -O3 -o helloworld.out helloworld.cpp
```

On Nautilus

```
module load intel/compiler

icpx -O3 -o helloworld.out helloworld.cpp
```

Compilation of the helloworld program

```
$ ls -lh
total 32K
-rw-rw-r-- 1 user user 123 Sep 16 08:50 helloworld.cpp
-rwxrwxr-x 1 user user 26K Sep 16 08:50 helloworld.out

$ file helloworld.cpp
helloworld.cpp: C++ source, ASCII text

$ file helloworld.out
helloworld.out: ELF 64-bit LSB executable, x86-64, [...]
```

Execution of the program

Localy

```
$ ./helloworld.out
hello world
```

Execution with OpenMP

Localy

```
$ export OMP_NUM_THREADS=4; ./helloworld.out
hello world
```

Execution with MPI

Localy

```
$ mpirun -np 4 ./helloworld.out
hello world
hello world
hello world
hello world
```

Execution of the program with SLURM

Sequential job

```
$ sbatch -M nautilus --partition=standard --qos=short run_seq.slurm
```

OpenMP job

```
$ sbatch -M nautilus --partition=standard --qos=short run_omp.slurm
```

MPI job

```
$ sbatch -M nautilus --partition=standard --qos=short run_mpi.slurm
```

run_seq.slurm file

```
#!/bin/bash
#SBATCH -J Job_SEQ
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=1
#SBATCH --time=00:05:00
#SBATCH --exclusive

module purge
module load intel/compiler

./helloworld.out  > out_seq_${SLURM_JOB_ID}.txt
```

run_omp.slurm file

```
#!/bin/bash
#SBATCH -J Job_OMP
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=4
#SBATCH --time=00:05:00
#SBATCH --exclusive

module purge
module load intel/compiler

if [ -n "$SLURM_CPUS_PER_TASK" ]; then
  omp_threads=$SLURM_CPUS_PER_TASK
else
  omp_threads=1
fi
export OMP_NUM_THREADS=$omp_threads
export OMP_PROC_BIND="spread,close"
export OMP_PLACES=core

ni=`printf "%07d\n" ${omp_threads}`
./helloworld.out > out_${ni}t_${SLURM_JOB_ID}.txt
```

run_mpi.slurm file

```
#!/bin/bash
#SBATCH -J Job_MPI
#SBATCH --nodes=1
#SBATCH --ntasks=4
#SBATCH --time=00:05:00
#SBATCH --exclusive

module purge
module load intel/compiler intel/mpi

export I_MPI_COLL_EXTERNAL=0
export I_MPI_ADJUST_BCAST=0
export I_MPI_FABRICS=shm:ofi
export FI_PROVIDER=psm3

ni=`printf "%07d\n" ${SLURM_NPROCS}`
srun --mpi=pmi2 ./helloworld.out  > out_${ni}c_${SLURM_JOB_ID}.txt
```

run_mpi.slurm file (bis)

```
#!/bin/bash
#SBATCH -J Job_MPI
#SBATCH --nodes=1
#SBATCH --ntasks=4
#SBATCH --time=00:05:00
#SBATCH --exclusive

module purge
module load intel/compiler intel/mpi

export I_MPI_COLL_EXTERNAL=0
export I_MPI_FABRICS=shm:ofi
export FI_PROVIDER=mlx
export I_MPI_ADJUST_ALLGATHER=0
export I_MPI_PMI_LIBRARY=/lib64/libpmi2.so
export UCX_NET_DEVICES=mlx5_2:1
export UCX_TLS=dc_mlx5

ni=`printf "%07d\n" ${SLURM_NPROCS}`
srun --mpi=pmi2 ./helloworld.out > out_${ni}c_${SLURM_JOB_ID}.txt
```