HelloWorld:
OpenMP and MPI Versions

1. Create HelloWorld with OpenMP & MPI
2. Compile the code with OpenMP & MPI
3. Execute the code with OpenMP & MPI

Compilation of helloworld with OpenMP

Localy

```
g++ -O3 -fopenmp -o helloworld_omp.out helloworld_omp.cpp
```

On Nautilus

```
module load intel/compiler

icpx -O3 -fopenmp -o helloworld_mpi.out helloworld_mpi.cpp
```

Compilation of helloworld with MPI

Localy

```
mpicxx -O3 -o helloworld_mpi.out helloworld_mpi.cpp
```

On Nautilus

```
module load intel/compiler intelmpi

mpicxx -cxx=icpx -O3 -o helloworld_mpi.out helloworld_mpi.cpp
```

Compilation of a hybrid helloworld (MPI/OpenMP)

Localy

```
mpicxx -O3 -fopenmp -o helloworld_mpi.out helloworld_mpi.cpp
```

On Nautilus

```
module load intel/compiler intelmpi

mpicxx -cxx=icpx -fopenmp -O3 -o helloworld_mpi.out helloworld_mpi.cpp
```

Headers to include

```
#include <omp.h>  // For OpenMP

#include <mpi.h>  //  For MPI
```

Built-in Fonctions

OpenMP	MPI
int omp_get_num_threads();	int MPI_Comm_size(MPI_Comm comm, int *size);
int omp_get_thread_num();	int MPI_Comm_rank(MPI_Comm comm, int *rank);

MPI Communicator

```
[...]
int worldsize ;
MPI_Com_size(MPI_COMM_WORLD,&worldsize)
cout << "Calculation involves " << wordsize << " processes" <<  endl;
[...]
```

OpenMP "Communicator"

```
[...]
cout << "Calculation involves "<< omp_get_num_threads() <<" threads"<< endl;
[...]
```

OpenMP Section with Pragma

```
// sequential section
[...]

#pragma omp parallel
{  // <- opening parallel section

  [...]  // parallel section of code

}  // <- closing parallel section

// sequential section
[...]
```

MPI Initialization and Finalization

```
int MPI_Init( int *argc, char ***argv );
int MPI_Finalize( );
```

Essential Code for MPI

```
int main(int argc, char** argv)
{
  MPI_Init(&argc,&argv);

  [...]

  MPI_Finalize();
  return 0;
}
```

Measuring time with OpenMP and MPI

OpenMP	MPI
`double omp_get_wtime();`	`double MPI_Wtime();`

Example of timer use

```
double t0, t1;
t0 = MPI_Wtime();

// portion of code to be evaluated

t1 = MPI_Wtime();
cout << "time : " << t1-t0 << " seconds" << endl;
```