1. DISTRIBUTED OPEN MPI

Open MPI programs can run in a distributed manner. To deploy it on multi-node cluster, installation must be done on all participating nodes to ensure a consistent environment. First, install OpenMPI on every machine, then configure password less SSH between nodes. In each node, generate an SSH key and distribute it to all other nodes.

```
# Generate SSH key
ssh-keygen

# Distribute to all other nodes
ssh-copy-id <USERNAME_1>@<IP_1>
ssh-copy-id <USERNAME_2>@<IP_2>
```

Once nodes can reach each other via SSH without password, in each node, prepare a hostfile (e.g., hosts.txt) listing the IP addresses (or hostname) of the machines and the number of slots (CPUs) each provides. For example:

```
# in hosts.txt
10.1.11.1   slots=4
10.1.11.2   slots=4
10.1.11.3   slots=8
```

To run an MPI program across multiple nodes, you must explicitly specify the –hostfile option. By default, Open MPI attempts a fair distribution of processes across nodes, but it does not guarantee exactly one process per node. To enforce this behavior, use the –map-by option with the value ppr:1:node, which stands for “Processes Per Resource: 1 per node.” Other parts stay the same as local MPI environment.

```
# Execute Open MPI program on a distributed environment
mpirun -np <NUM_PROCESS> --hostfile hosts.txt ./<PROGRAM_NAME>

# Force MPI to distribute exactly 1 process per node
# It returns error if <NUM_PROCESS> is greater than the number of nodes
mpirun -np <NUM_PROCESS> --hostfile hosts.txt --map-by ppr:1:node ./<PROGRAM_NAME>
```

In a typical large-scale MPI program, the computation is distributed across multiple processes running on multiple nodes. Each MPI process is often responsible for a portion of the overall workload and may internally create multiple threads (using shared-memory parallelism, such as OpenMP) to further parallelize the computation on its local node. This hybrid parallelism model — combining MPI for inter-node communication and multithreading for intra-node computation — is commonly used to fully exploit the capabilities of modern High Performance Computing (HPC) systems.

2. EXERCISES

In this exercise, students are required to simulate the heat diffusion, radioactive contamination concentration, and shock wave blast resulting from the detonation of a DF-61 ICBM. The simulation runs for 100 iterations, with each time step representing 1 second (or 1 convolution), as in the previous exercises. This assignment focuses on distributed parallel computing, requiring students to perform computations using multiple processes and multi-threading **across multiple machines** (distributed MPI). In particular, students are expected to:

- Implement the simulation using a synchronous execution method.
- Implement the simulation using an asynchronous execution method.
- Perform a runtime performance comparison between the synchronous and asynchronous implementations.
- Provide a detailed report on the asynchronous implementation, including an explanation and analysis of the results.

**** Submission:** Works can be done in groups.

- Compress all necessary files as **"Lab_4_<Student ID>.zip"**.
- Students must submit and demonstrate their results before the end of the class.