ETH zürich

High Performance Computing for Science and Engineering I

P. Koumoutsakos, W. Petersen, P. Hadjidoukas
ETH Zentrum, CLT E 13
CH-8092 Zürich

Fall semester 2017

Set 9 - Non-blocking communication and collectives

Issued: November 24, 2017
Hand in (optional): December 01, 2017 8:00am

Question 1: MPI bug hunting and asynchronous communication

In the following MPI code, rank 0 distributes a number ($= M * size$) of input values to all ranks of the MPI application. Each input value is processed by the `do_work()` function, which has uniform but significant execution time.

1. Identify and explain possible issues in the MPI code.
2. Explain how you can address the above issues.
3. Optional: Provide a solution code .

```
// void do_work(int);
// rank: MPI process id
// size: number of MPI processes

int M = 2;  //  any value > 1
int input;

if (rank == 0) {
    srand48(time(0)); // initialize the random seed

    int N = M*size;
    for (int i = 0; i < N; i++) {
        input = lrand48() % 1000;   // some random value
        MPI_Send(&input, 1, MPI_INT, i%size, 100, MPI_COMM_WORLD);
    }
}

for (int i = 0; i < M; i++) {
    MPI_Recv(&input, 1, MPI_INT, 0, 100, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
    do_work(input);
}
```

Hints:

- Make sure your solution does not introduce or imply correctness issues (e.g. race conditions).
- Avoid any assumptions on the communication protocol and the number of MPI processes.
- Try to overlap communication with computation.

Question 2: Diffusion: non-blocking communication and statistics

In the provided skeleton code `diffusion2d_mpi.cpp`, you find a simplified version, with slightly modified initial conditions, of the MPI code for the 2D diffusion problem.

a) Implement the non-blocking MPI communication by filling in all parts of the code marked by TODO:MPI.

b) The provided code includes a sequential diagnostics function `compute_max_density()` which computes and prints the maximum density value and its location of the local subdomain. Provide an MPI parallel implementation of the above diagnostics function in `compute_max_density_mpi()`. Process with rank 0 must print the overall maximum density value and its location.

Hints:

- To enable correctness check, the `compute_diagnostics()` routine of exercise 8 has been added to the skeleton code.
- Use of the `MPI_REQUEST_NULL` (and `MPI_STATUSES_IGNORE`) can simplify the implementation. An example of how they can be used in `MPI_Waitall` can be found here: `https://gitlab.ethz.ch/hpcse17/hs2017/blob/master/examples/mpi2/irecv_waitall.c`.
- You can call the sequential `compute_max_density()` function on all MPI processes and see the partial result for each local subdomain.
- Avoid any assumptions on the number of MPI processes and prefer scalable solutions.