ETH zürich

High Performance Computing for Science and Engineering I

P. Koumoutsakos
S. Martin
ETH Zentrum, CLT E 13
CH-8092 Zürich

Fall semester 2020

Set 5 - MPI Part II

Issued: November 27, 2020
Hand in (optional): December 11, 2020 08:00am

Grading: To get full credits solve any two of the questions.

Question 1: Parallel Scaling (30 points)

A program simulates N particles, all particles interact with each other and, thus, the number of interactions is proportional to N^2. The runtimes of the program in seconds on P processor cores are in the table:

P\N	500	1000	1500	2000
1	6.00	30.00	72.00	120.00
4	1.50	7.50	18.00	30.00
9	0.75	3.50	9.00	20.00
16	0.50	2.15	6.00	12.00
24	0.40	1.50	4.50	10.00

Plot four points of the strong scaling. Show all steps of the calculations. Label the axes.

Plot four points of the weak scaling using $N = 500$ and $P = 1$ as a reference. Show all steps of the calculations. Label the axes.

Question 2: Diffusion (30 points)

The diffusion of a substance can be described by the equation

$$\frac{\partial c(x,y,t)}{\partial t} = D\left(\frac{\partial^2 c(x,y,t)}{\partial x^2} + \frac{\partial^2 c(x,y,t)}{\partial y^2}\right),$$

where c is the concentration of the substance at position $(x,\ y)$ and at time t, and D is the diffusion constant. The diffusion process happens in the domain $|x| < L/2$ and $|y| < L/2$. The concentration is zero on the boundaries of the domain. The initial concentration is

$$c(x,y,0) = \begin{cases} 1, & \text{if } |x| < L/4 \text{ and } |y| < L/4, \\ 0, & \text{otherwise.} \end{cases}$$

a) The skeleton code solves the equation on a uniform grid using a central finite difference scheme in space and forward Euler time integration. Parallellize the code by filling parts marked by `TODO` in the functions `advance` and `main`. Use a tiling decomposition scheme (i.e., distribute the rows evenly to the MPI processes).

b) For a given time compute the integral of $c(x,y,t)$ over the domain (total ammount of the substance). Fill the missing MPI parts in `compute_diagnostics`, and plot the result as a function of time using $D = 1$, $L = 2$ and $N = 100$.

c) For a given time compute the histogram of $c(x,y,t)$ in the function `compute_histogram` by implementing the missing MPI parts marked by `TODO`, and plot the resulting histogram for $t = 0.5$ using $D = 1$, $L = 2$ and $N = 100$.

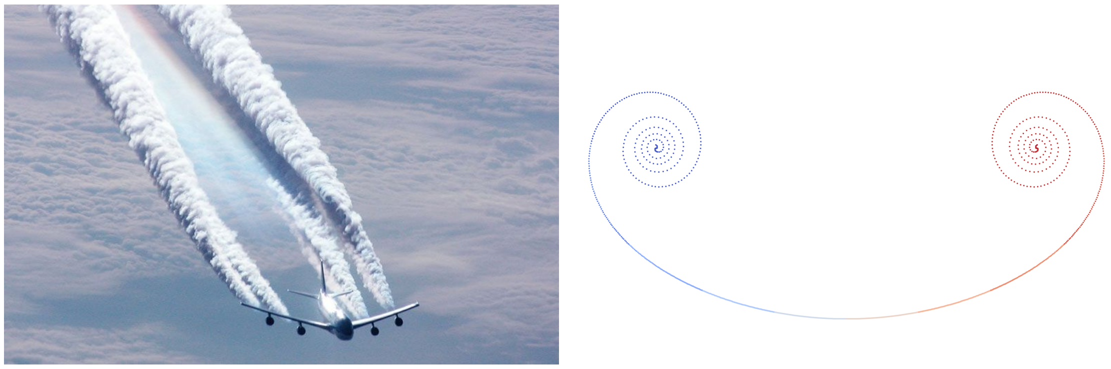


Figure 1: Left: Wake of an airplane visualized by condensation behind the engines. The vorticity sheet is generated at the trailing edge of the wings. Right: Vortex sheet at $t = 1$ from the simulation.

Question 3: Roll-up of a vortex line (30 points)

We want to simulate the evolution of a two-dimensional vorticity sheet similar to the wake of an airplane (Figure 1). We use N particles moving with time t, each particle i is located at $(x_i(t), y_i(t))$ and carries a constant value Γ_i (circulation). The velocity field defined as

$$u(x,y,t) = \sum_{i=0}^{N-1} \frac{\Gamma_i}{2\pi} \frac{-\left[y - y_i(t)\right]}{\varepsilon^2 + \left[x - x_i(t)\right]^2 + \left[y - y_i(t)\right]^2},$$

$$v(x,y,t) = \sum_{i=0}^{N-1} \frac{\Gamma_i}{2\pi} \frac{\left[x - x_i(t)\right]}{\varepsilon^2 + \left[x - x_i(t)\right]^2 + \left[y - y_i(t)\right]^2}$$

is an approximate solution of Euler equations and corresponds to the vorticity field

$$\omega = \frac{\partial v(x,y,t)}{\partial x} - \frac{\partial u(x,y,t)}{\partial y} = \sum_{i=0}^{N-1} \Gamma_i \delta_\epsilon \left(x - x_i(t), y - y_i(t)\right)$$

where $\delta_\epsilon(x,y) = \frac{1}{\pi}\frac{\epsilon^2}{(\epsilon^2+x^2+y^2)^2}$ approximate a Dirac-delta function for a small ϵ. A particle moves with the velocity at its location

$$\frac{dx_i(t)}{dt} = u(x_i(t), y_i(t), t), \qquad \frac{dy_i(t)}{dt} = v(x_i(t), y_i(t), t),$$

Initially, particles are placed at $y_i = 0$ and

$$x_i = -\frac{1}{2} + \frac{i + 1/2}{N}, \qquad i = 0, ..., N-1$$

and have circulation $\Gamma_i = \frac{1}{N}\frac{4x_i}{\sqrt{1-4x_i^2}}$

a) Implement the interaction function `computeVelocities` in `q3/serial.cpp`. Check your results by visualizing the csv files with paraview.

b) Parallelize your code using MPI by filling in the TODOs in `q3/mpi.cpp`. Each MPI rank must contain an equal number of particles. The parallelization of the `computeVelocities` can be done using a multi-pass communication, as described in the table:

		process p				
		0	1	$\cdots$	P-2	P-1
	0	D_0	D_1	$\cdots$	D_{P-2}	D_{P-1}
	1	D_{P-1}	D_0	$\cdots$	D_{P-3}	D_{P-2}
pass q	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	P-1	D_1	D_2	$\cdots$	D_{P-1}	D_0

the data necessary to compute the velocities (x_i, y_i and Γ_i) must be communicated to every rank in a cyclic manner until every rank has computed the interactions between its own particles with every particles in the simulation. Parallelize the function `dumpToCsv` by gathering the data on the root.

c) Use non blocking MPI routines to overlap the communication with computation. Write your solution in `q3/mpi_non_blocking.cpp`.

Question 4: Roofline Model (30 points)

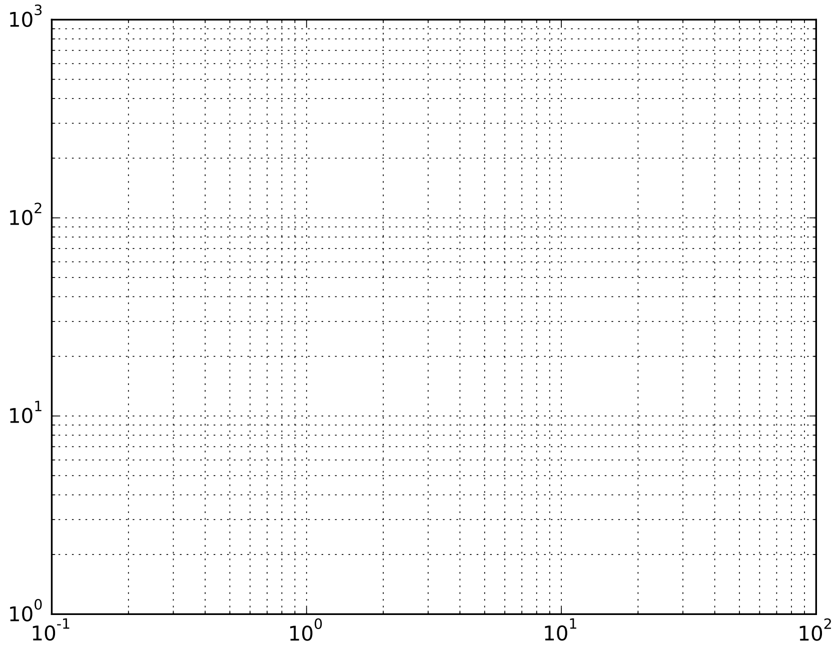
Given the following code:

```
float A[N], B[N], C[N];
...
const int P = 2;
for (int i = 0; i < N; ++i) {
    int j = 0;
    while (j < P) {
        A[i] = B[i] * A[i] + 0.5;
        ++j;
    }
    C[i] = 0.9 * A[i] + C[i];
}
```

a) What is the floating point operational intensity of the code? State all the assumptions you made and show your calculations.

b) For a peak performance of 409.7 GFLOP/s (single precision) and a memory bandwidth of 34 GB/s, find all positive P for which the code is memory bound. Assume an infinite cache and state further assumptions you made. Show your calculations.

c) Draw below the roofline corresponding to (b) and label the axes.



10^3
10^2
10^1
10^0
10^{-1}
10^0
10^1
10^2