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

Set 3 - Hybrid MPI & OPENMP and Communication Tolerant Programming

Issued: March 23, 2020
Due Date: April 6, 2020 08:00am

The skeleton codes for this homework are located in the gitlab repository: https://gitlab.ethz.ch/hpcse20/exercise.

In this exercise, you will continue working on the wave equation

$$\frac{\partial^2 u}{\partial t^2} - c^2 \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right) = 0 \tag{1}$$

with periodic boundary conditions

$$u(t,0,y,z) = u(t,1,y,z) \ , \ u(t,x,0,z) = u(t,x,1,z) \ , \ u(t,x,y,0) = u(t,x,y,1) \tag{2}$$

and initial conditions

$$u(0,x,y,z) = f(x,y,z) = e^{-10r^2} \ , \ \frac{\partial u}{\partial t}(0,x,y,z) = 0 \tag{3}$$

with $r = \sqrt{(x-0.5)^2 + (y-0.5)^2 + (z-0.5)^2}$. It is assumed that $c = 1$, the problem domain is the unit cube and a uniform mesh of $N_{tot} \times N_{tot} \times N_{tot}$ grid points is used.

You will start with a skeleton code that solves eq.1 using MPI. You will be asked to gradually modify the code so that thread-level parallelism (OPENMP) is also exploited. Similarly to the previous exercise, p^3 MPI processes are assumed, as well as τ^3 OPENMP threads per MPI process. Each process will be working on a problem subdomain of $N \times N \times N$ grid points, where $N = N_{tot}/p$; additionally, this subdomain will need to be further divided among threads. Each thread will be working on a $n \times n \times n$ cube, where $n = N/\tau$. See also fig.1.

Question 1: Grid point update (20 points)

Use OPENMP threads to parallelize the loop that updates grid points according to

$$u^{n+1}_{i,j,k} = 2u^n_{i,j,k} - u^{n-1}_{i,j,k} + c^2 \frac{\Delta t^2}{h^2} (u^n_{i+1,j,k} + u^n_{i-1,j,k} + u^n_{i,j+1,k} + u^n_{i,j-1,k} + u^n_{i,j,k+1} + u^n_{i,j,k-1} - 6u^n_{i,j}) \tag{4}$$

Each thread needs to work on a different but fixed part of its process' subdomain; thus you should not simply use '#pragma omp parallel for'. In the auxiliary.cpp file you will see an example of how you should implement this. Please refer to the skeleton code for further clarification.

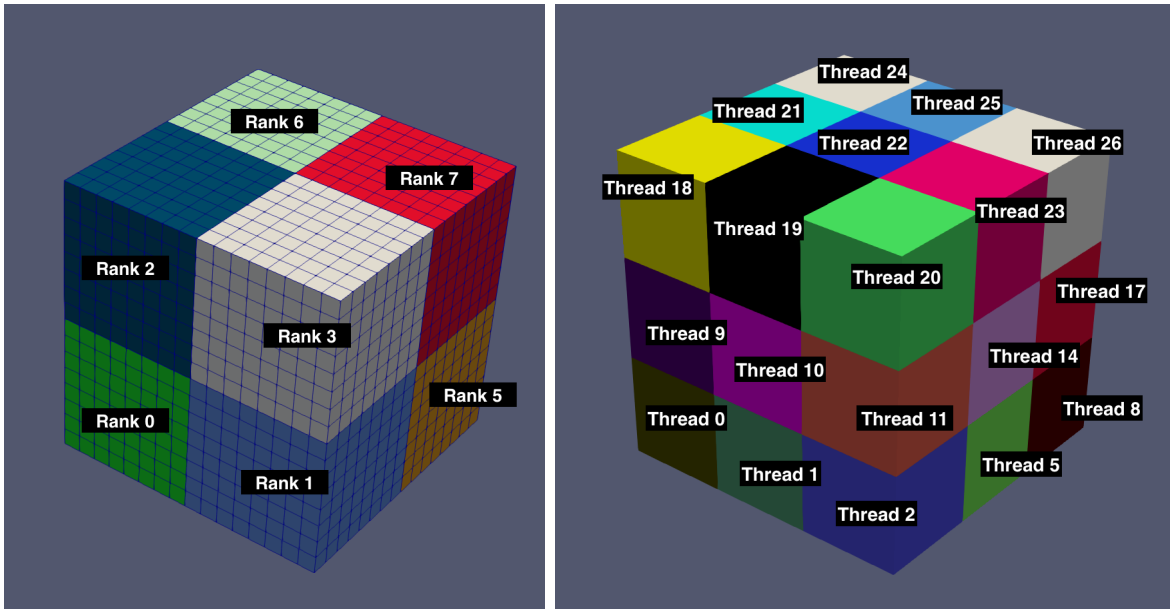

Figure 1: Domain decomposition of unit cube for 8 MPI processes, $p = 2$ (left) and subdomain decomposition per MPI process, $\tau = 3$ (right).

Question 2: Data packing and message exchange (40 points)

Use OPENMP threads to speed up the packing/unpacking of data exchanged among processes. Instead of having a single thread packing/unpacking all data, each thread should pack/unpack whatever data lies inside its $n \times n \times n$ subdomain. Then, use OPENMP threads to communicate between processes. Now that each thread has packed its own data, you are asked to send/receive multiple messages using multiple threads. Each thread should send/receive data that lies inside its $n \times n \times n$ subdomain. Please refer to the skeleton code for further clarification.

Question 3: Wrapping it all up (20 points)

You should now have a code with four different parallel regions: two for packing/unpacking data, one for updating each grid point and one for exchanging messages between processes. Each parallel region is associated with some overhead, when it is created. To avoid this, combine all four regions to a single one; that is, you should solve the same problem by creating a single parallel regions throughout the whole execution of your program. Why is an *omp barrier* required at the end of each timestep? Please refer to the skeleton code for further clarification.

Question 4: Communication and computation overlap (20 points)

In a last step, you are asked to overlap communication and computation. First, send the data required to update the points on the boundary, and then update the inner points of each thread's subdomain. Finally, wait for communication to finish and update the points on the boundary. Please refer to the skeleton code for further clarification.

Please consult the README file that is provided with the code for compilation instructions and further information.

Guidelines for reports submissions:

- Archive your source code (e.g.: .tar, .rar, .zip) and submit it via Moodle until April 6, 2020, 08:00am.