Innovative Computing Laboratory University of Tennessee, Knoxville

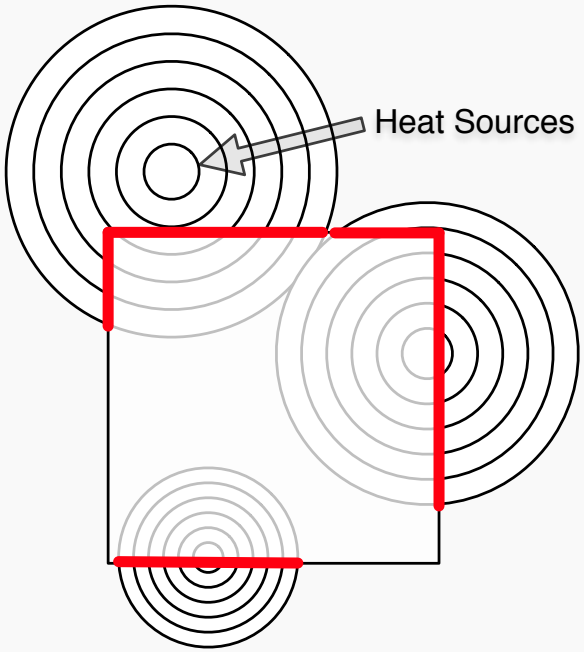
HW4 : Heat propagation using OpenSHMEM

George Bosilca

October 16, 2017

Stencil 2D
Heat transfer

- Heat transfer is a discipline of thermal engineering that concerns the generation, use, conversion, and exchange of thermal energy (heat) between physical systems.
- Heat convection occurs when bulk flow of a fluid (gas or liquid) carries heat along with the flow of matter in the fluid.



The square is the 2D surface where the heat propagation is to be observed. The circles are the heat sources, and the heat waves they generate. The red lines are the impact of these heat sources on the boundaries of the 2D surface, and represent the stable boundary condition of the problem (they are stored in an extra column/row and are not supposed to be altered during the execution).

Stencil 2D
Jacobi on distributed environments

You should start with the provided template and transform it into a fully distributed application using OpenSHMEM. We will work under the same assumptions as for the MPI homework, the data is row-major (contiguous in memory per row). Assume the application will use a cartesian grid of *PxQ* processors, and the data will be distributed by 2d blocks. We will use the data distribution described in the lecture (with the ghost region on each node) as indicated in the Figure 1. This is similar to the slicing used in the MPI version of the code.

To facilitate the development, here are the steps you should follow:

- Design the data exchange with the north and south neighbors.
- Design the data exchange with the east and west neighbors.
- Build the communication pattern where each process (with the exception of those on the boundary) exchange data with its 4 neighbors: east, west, north and south.
- Plug the node level Jacobi into the code
- Make sure you reuse as much code as possible.

Stencil 2D
Communication pattern

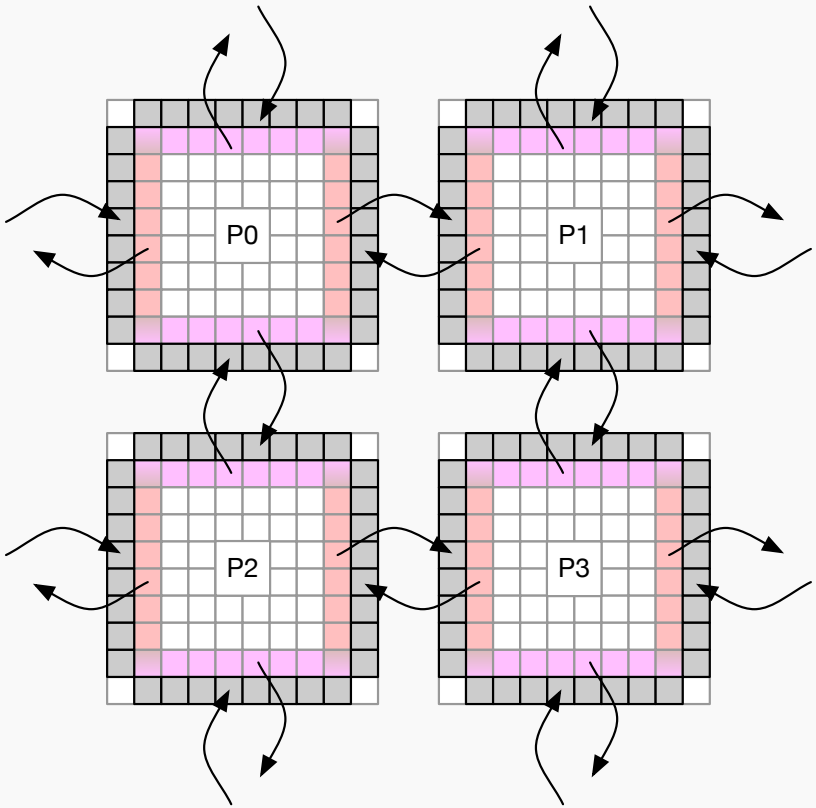


Figure: Communication patterns between neighbors

- Design the communication pattern. Pay attention to ensure correct synchronizations between processes.
- The synchronization can be made global (where all processes participate on the same synchronization), or local where the synchronization is only between peers exchanging data. Minimize the synchronization burden by implementing a fine grain synchronization scheme.
- Do you have any overlap between communications and computations ? If the answer is no, then you are not yet done with the homework.

What to check

- The code provided delivers the correct answer. A correct implementation of the distributed version of the Jacobi should not only remain deterministic, but provide bit-wise reproducibility of the result. Check your result against the provided version (sequential Jacobi).
- Validate the performance you obtain. The computational part is almost embarrassingly parallel, but its performance is impacted by the cost of the data exchange)the ghost regions at each iteration. Make sure that, compared with the provided code, your code delivers the expected performance boost, depending on the number of processes.
- Compare the performance of the obtained code against the MPI version from the HW2. How does the performance compare ?

The deadline for HW4 is 11/06/2017!