Innovative Computing Laboratory University of Tennessee, Knoxville

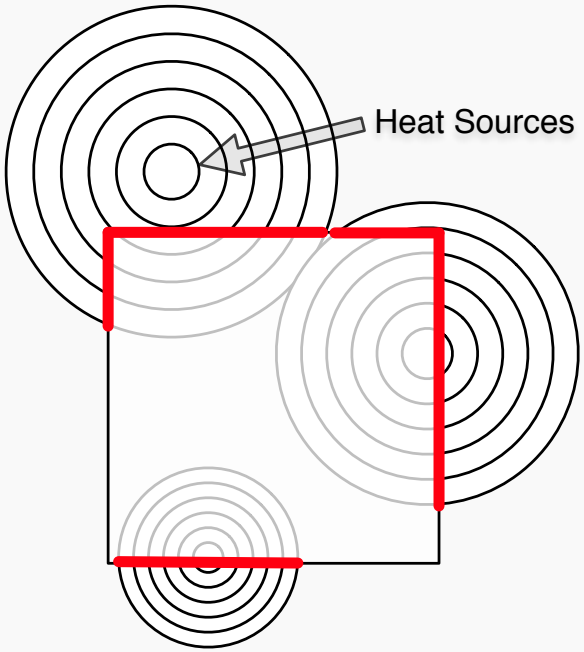
HW5 : Heat propagation using OpenSHMEM and OpenMP

George Bosilca

November 1, 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).

You should start with the provided template and transform it into a fully distributed application using a mixed programming approach where you combine OpenSHMEM and OpenMP. Unlike the previous approaches where we assumed a process has a hardware core to execute on, this time we will work under the assumption that there is a single process per node, and this process will use the specified number of cores (using OpenMP). We will work under the same assumptions as for most of the previous 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.

To facilitate the development, you should follow the same steps as in the OpenSHMEM homework, and then improve upon by using OpenMP to take over the computation local on each node (in this case there is a single process per node).

Stencil 2D
Communication pattern

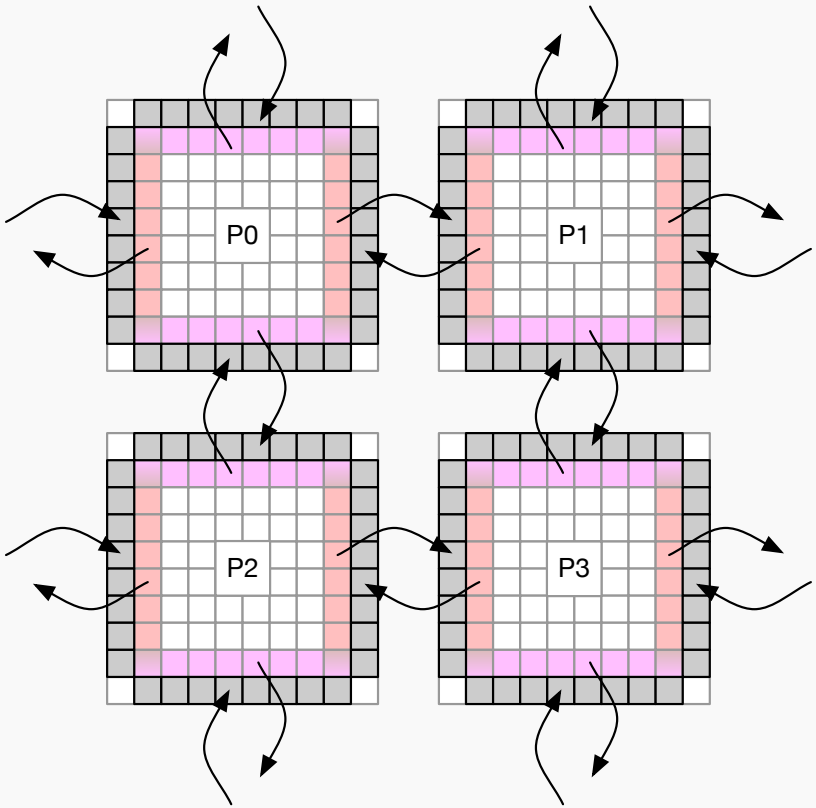


Figure: Communication patterns between neighbors

- Remember the discussion from the class on the interaction between the data distribution (across cores) and the communication costs. Make sure the data is distributed among processes and then among threads in the most efficient way.
- Do you have overlap between communications and computations ?
- In the context of each process distribute the data between threads while trying to minimize the restrictions imposed on the communication/computation overlap.

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 total number of cores (larger sizes of the problem might be needed to achieve such performance scalability).
- Compare the performance of the obtained code against the pure OpenSHMEM version from the HW4. How does the performance compare ? And in comparison with the outcome of HW3 ?

The deadline for HW5 is 11/22/2017!