**Blue Waters Petascale Semester Curriculum v1.0**

**Unit 4: OpenMP**

**Lesson 5: Convolution in OpenMP (Heat Transfer example)**

**Exercise Instructions for Students**

*Developed by Maria Pantoja for the Shodor Education Foundation, Inc.*



*Except where otherwise noted, this work by The Shodor Education Foundation, Inc. is licensed under CC BY-SA 4.0. To view a copy of this license, visit*[*https://creativecommons.org/licenses/by-sa/4.0*](https://creativecommons.org/licenses/by-sa/4.0)

*Browse and search the full curriculum at*[*http://shodor.org/petascale/materials/semester-curriculum*](http://shodor.org/petascale/materials/semester-curriculum)

*We welcome your improvements! You can submit your proposed changes to this material and the rest of the curriculum in our GitHub repository at*[*https://github.com/shodor-education/petascale-semester-curriculum*](https://github.com/shodor-education/petascale-semester-curriculum)

*We want to hear from you! Please let us know your experiences using this material by sending email to* [*petascale@shodor.org*](mailto:petascale@shodor.org)

1. Follow the instructor directions
   1. Download: Slides, Video, and code
   2. Before continuing on the next exercises make sure you understand the provided code and you are able to compile and run it
2. Do answer Sample Assessment question to evaluate your basic understanding of learning objectives

Sample Exercise:

1. Rewrite provided code provided so instead of using only the neighbor to the North, South, East, and West uses also the neighbor

|  |  |  |
| --- | --- | --- |
| NW | North | NE |
| West | Center | East |
| SW | South | SE |

1. Use the function MPI\_Wtime() to measure execution times of different loops and/or functions on the code. Example

**double start;**

**double end;**

**start = omp\_get\_wtime();**

... work to be timed ...

**end = omp\_get\_wtime();**

**printf(~Work took %f seconds\n~, end - start);**

1. Change the number of OpenMP threads used and obtain runtimes of sequential code (also provided) vs. distributed code using those different process

Create a table and graphs to study execution.

|  |  |
| --- | --- |
| Sequential | Execution time |
| OpenMP 1 Thread |  |
| OpenMP 2 Threads |  |
| …. |  |

Do the difference execution times for different threads match your expectation? What happens if you use too many threads (“too many threads” can be 32, 64, …)?