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

**Unit 6: Hybrid MPI + OpenMP**

**Lesson 3: Pebble in Pond Wave Equation**

**Instructor Guide**

*Developed by R. Phillip Bording 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)

* Since this is only a 25 minutes lesson, the CPU multicore/multithread and GPU hardware architectures should already be covered in previous lessons. The focus of this lesson should only be on adding OpenMP to existing MPI code implementations and its use cases in different scientific applications.
* Instructors should review the materials covered in the presentation slide set and do further readings of the concepts being presented.
* This lesson will start with presentation slides that covers multi-core and multi-threaded CPU’s, concepts, and data array movement to/from memory cache.
* Instructors should use simple random array addition examples to demonstrate primary MPI subroutines for copying the data from node to node, and vice versa. Then, this program can be expanded to use OpenMP for testing compiler directives.
* The acoustic wave equation example is used as a more scientific application use case. This example will also demonstrate good and bad practices in MPI/OpenMP programming that students should be aware of.
* Understand and present the idea of a post processing step to develop visual graphics animations and know how to use ImageMagik.

**Common Pitfalls for Students and Instructors**

* Depending on when this lesson is used for teaching or learning, MPI/OpenMP implementation might have changed. Therefore, both instructors and students are encouraged to check the HPC community for updates.
* Watch out for IO, as MPI outputs data writes in random order. Moving data between nodes could potentially reduce application performance tremendously.
* Beware of making many changes and then timing results. Use an incremental approach – make one change – then time entire program. See what runtime improvements happened. Rerun several times to determine if actual times are consistent.
* In the beginning, do a paper design of memory layout and decide either a one dimension or two-dimensional compute node structure is best.