The wave equation as a motivating example for high performance computing

David A. Joiner
New Jersey Center for Science, Technology, and Mathematics
Kean University
Union, New Jersey, USA
djoiner@kean.edu

***Abstract*—A series of activities based on the solution of the wave equation in 1 and 3 dimensions spanning multiple concepts in a traditional high performance computing class are presented.**

Keywords—parallel computing education, high performance computing education

I. INTRODUCTION

Topics for use in a classroom setting for high performance computing suffer from competing challenges of (1) showing reproducible speedup and scalability, (2) having authenticity, and (3) being able to present and complete in a standard classroom period. This paper will describe the numerical solution of the wave equation as a motivating example used in CPS 5965 High Performance Computing at Kean University. The problem is used in multiple contexts across different course meetings, and provides an example giving continuity across 4 key topics in my course (performance programming, threaded parallelism, accelerated parallelism, and distributed parallelism).

II. INITIAL USE, PERFORMANCE COMPUTING

The problem posed to students is to solve for a solution of the equation

$$(d^2 \psi)/(dt^2)=\nabla^2 \psi$$

Given fixed boundary conditions at a set extreme and an initial perturbed (typically Gaussian) configuration between the extremes. Students are provided with an algorithm for solving the second derivative of a function with regards to a spatial variable, as well as an overview of Euler's method. They are asked to implement the pseudocode, which includes initial conditions and sample output, in their language of choice. This allows for us to begin one of the first key discussions in the course, choice of language. Most students entering in the course have backgrounds in languages other than C, C++, or Fortran, and will typically write their code in Java, Python, or MATLAB. The student implementations will be compared to each other, as well as solved examples in those languages as well as C, C++, Fortran, JavaScript, C#, and PERL. For languages with the availability of an optimizing compiler, a range of optimization options are used, as well as commercial compilers if available. The wall time of the solution is sorted by language given a simple student created benchmark of available languages for HPC use. Results can vary by platform and installation, but generally students will see significant performance differences between interpreted languages and compiled languages, as well as a significant performance difference between architecture specific optimizing compilers and platform independent non-optimizing compilers (e.g. PERL, Python, MATLAB typically performs slower than C#, Java, JavaScript, which in turn typically performs slower than C, C++, and Fortran). This helps to motivate the use of C/C++ as the language in which my course is taught, and leads into subsequent discussions of performance tuning, particularly the importance of optimizing for sequential access through contiguous memory.

III. REPRISE OF ACTIVITY, PARALLELIZATION

We return to the wave equation when discussing threaded parallelism in OpenMP, where students implement OpenMP in a C version of the wave equation code (written by the students as a homework after the previously described activity). Students discuss in class different options for choices of loops to parallelize, and scheduling options to be used. Students are asked to compare speedups for increasingly higher resolutions of their wave, and to determine how large their wave array needs to be before any performance improvement is seen. Students typically will quickly see that the size of the array required to show speedup for the 1D solution is both substantially greater than needed for an accurate solution to the problem given, but also that in making the problem higher resolution than needed, the wall time for even the parallel implementation is greater than the simpler, serial approach that they started from. The follow up activity involved the students being given starter code for a 3D version of the same problem, which they are require to parallelize and profile. They typically will determine that in the 3D version of the problem, the greater amount of work required to solve the problem regardless of parallelization is a more natural fit for a parallel solution. The 3D wave equation code is then also used as a first example in units on CUDA and MPI.

A handout on the mathematics of the wave equation, pseudocode given to the student, solved examples in multiple languages for the language comparison, starter and solved code for OpenMP, CUDA, and MPI for 1 and 3D will be provided to participants.