Accelerating PDE Approximations Through Parallel Architectures

URL: ParallelPDE

Team member: Hao Wu

Summary

I will compare and combine SIMD, CUDA, OpenMP and Open MPI libraries to efficiently approximate solutions to basic partial differential equations.

Background

The specific types of PDEs I'm targeting are: 1D wave equations, and 2D Laplace and Poisson equations. The first 2 types are basic grid solvers: approximating 1D wave equations only involve updating a 1D array, whereas 2D Laplace equations need updating a 2D grid. Basic blocking and SIMD instructions can be used to parallelize updates. I'll compare memory and time costs of using CUDA threads, OpenMP tasks, and Open MPI tasks for allocating work. If I have extra time, I'll incorporate a dynamic scheduler to improve work balance. 2D Poisson equations are much more complicated. They involve solving a large sparse linear system, which I intend to approximate by Gauss-Seidel, Jacobi, and SOR methods. Sequential solvers are already implemented, and I need to find ways to parallelize most parts of these methods. In particular, I will research ways on efficiently storing and multiplying sparse matrices, probably by blocking. In this process, I'll construct a small, specialized, and efficient linear algebra library based on the optimal combination of different parallel architectures.

Challenge

For the first 2 types of PDEs, it's not hard to parallelize: the difficulty lies in determining which scheme is best for which kind of updates. Comparing the performance on specific applications of these schemes provide me a deeper understanding of their differences. For the last type of PDEs, solving large sparse linear systems typically cause work imbalance if work is not assigned carefully. So (probably dynamic) work assignment is the hardest part. Optimizing matrix operations for sparse matrices is also difficult.

Resources

- Machines: I'll first run on my personal computer with Intel 6-core i7 CPU and Nvidia Geforce GTX 1050 Ti Max-Q GPU. Then I'm going to optimize for Latedays Cluster machines with 2 6-core Xeon e5-2620 V3 CPUs and a NVIDIA K40 GPU + 60-core Xeon Phi 5110P.
- Starter code: I implemented sequential version of G-S, Jacobi, SOR, as well as the whole 2D Poisson equations solver for a fixed simple set of boundary conditions.
- Formulae: For 1D wave equations and 2D Laplace equations, I can refer to Parallel Algorithms for Solving Partial Differential Equations. For 2D Poisson equations, I can refer to PARALLEL METHODS FOR SOLVING PARTIAL DIFFERENTIAL EQUATIONS.

Goals and Deliverables

Plan to achieve:

First 2 equations:

- Comparison graphs of CUDA, OpenMP, and Open MPI performance in terms of runtime and memory cost, as well as arithmetic intensities for different problem sizes and machines.

2D Poisson equations:

- A C++ library of matrix operations involved in solving sparse linear systems, implemented with SIMD instructions, and best-performing architecture possibly combining CUDA, OpenMP and Open MPI.
- Comparison graphs of my utilization code versus the famous Eigen library in terms of runtime for dense and sparse matrices.
- A 3D model based on the approximate solution of a 2D Poisson equation.

Hope to achieve

- Optimize matrix operations specifically for sparse matrices.
- Dynamic work assignment.

Platform Choice

I'm using C++ because I'm familiar with its SIMD, OpenMP and Open MPI instructions, and also because I wrote sequential version of 2D Poisson equation solver in C++. I'm targeting 2 types of machines in order to compare how different instruction sets behave in different runtime environments.

Schedule (subject to frequent changes)

Oct 31 - Nov 3

Research: SIMD, CUDA vs Open MPI

Nov 4 - 6

Research: Solving 1D wave + 2D Laplace equations by finite element method

Nov 7 - 10

Implement: Sequential + CUDA versions of 1D wave equation

Nov 11 - 13

Implement: Open MPI version of 1D wave equation

Implement: CUDA version of 2D Laplace equation

Nov 14 - 17

Implement: Open MPI version of 2D Laplace equation

Research: OpenMP

Wrap-up: Evaluate performance of CUDA vs Open MPI versions

Nov 18 - 19

Reflection + Checkpoint

Nov 20

Midterm 2

Nov 21 - 24

Implement: OpenMP version of 1D wave equation

Implement: OpenMP version of 2D Laplace equation

Wrap-up: Evaluate performance of OpenMP

Nov 25 - Dec 1

Research: Sparse linear system optimizations

Implement: Most SLA functions using SIMD with CUDA, OpenMP and Open MPI

Dec 2 - 4

Implement: All remaining SLA functions

Wrap-up: Evaluate performance of optimized SLA functions

Dec 5 - 8

Implement: Main 2D Poisson equations solver

Wrap-up: Create all relevant deliverables

Dec 9

Final Report