Accelerating PDE Approximations Through Parallel Architectures

URL: ParallelPDE

Team member: Hao Wu

Summary

I will compare and combine SIMD, CUDA, OpenMP and 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 Poisson equations. The first type is a basic grid solver: approximating 1D wave equations only involve updating a 1D array. 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 MPI tasks for allocating work. 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 of 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 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 second 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. I'll use 256-bit SIMD instructions, and MS-MPI.
- Starter code: I implemented a 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, 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:

1D Wave equations:

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

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 MS-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 MPI instructions, and also because I wrote a sequential version of 2D Poisson equation solver in C++.

Schedule (subject to frequent changes)

Oct 31 - Nov 3

Research: SIMD, CUDA vs MS-MPI

Nov 4 - 6

Research: Solving 1D wave by finite element method

Nov 7 - 13

Implement: Sequential + SIMD versions of 1D wave equation

Nov 14 - 17

Implement: CUDA + MS-MPI version of the 1D wave equation

Nov 18 - 19

Research: OpenMP

Reflection + Checkpoint

Nov 20

Midterm 2

Nov 21 - 24

Implement: OpenMP version of the 1D wave equation

Nov 25 - Dec 1

Research: Sparse linear system optimizations

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

Dec 2 - 4

Implement: All remaining SLA functions

Wrap-up: Evaluate the performance of optimized SLA functions

Dec 5 - 8

Implement: Main 2D Poisson equations solver

Wrap-up: Create all relevant deliverables

Dec 9

Final Report