Homework Assignment 3: Solving the Poisson Equation on a 3D Lattice with CUDA

Haohan Tsao B09607009

1 Introduction

This report presents the implementation and analysis of solving the Poisson equation on a three-dimensional lattice using CUDA parallel programming. The Poisson equation is a fundamental partial differential equation that appears in many areas of physics, including electrostatics, fluid dynamics, and heat transfer. In this assignment, we focus on the electrostatic potential generated by a point charge in a cubic domain with fixed boundary conditions.

The Poisson equation in the context of electrostatics is given by:

$$\nabla^2 \phi = -\rho \tag{1}$$

Where:

- ϕ is the electric potential
- ρ is the charge density
- ∇^2 is the Laplacian operator

For a point charge q located at position (x_0, y_0, z_0), the charge density can be expressed as:

$$\rho(x, y, z) = q\delta(x - x_0)\delta(y - y_0)\delta(z - z_0) \tag{2}$$

Where δ is the Dirac delta function.

The assignment required solving this equation in a cube of size $L \times L \times L$ with a point charge $q = 1$ at its center, subject to boundary conditions where the potential is zero on the entire surface. We were asked to investigate how the numerical solution approaches the analytical solution (Coulomb's law) as the lattice size L increases.

2 Theoretical Background

2.1 Analytical Solution: Coulomb's Law

In an infinite domain, the analytical solution for the electric potential due to a point charge q at a distance r from the charge is given by Coulomb's law:

$$\phi(r) = \frac{q}{4\pi r} \tag{3}$$

This expression serves as the reference for evaluating our numerical results. In our case, with $q = 1$, the potential should approach $\frac{1}{4\pi r} \approx \frac{0.0796}{r}$ as the lattice size increases.

2.2 Discretization of the Poisson Equation

On a three-dimensional lattice with spacing $a = 1$, the Laplacian operator can be discretized using the finite difference method:

$$\begin{aligned}\nabla^2\phi(x,y,z) \approx \frac{1}{a^2}[\phi(x+1,y,z) + \phi(x-1,y,z) + \phi(x,y+1,z) +& \\ \phi(x,y-1,z) + \phi(x,y,z+1) + \phi(x,y,z-1) - 6\phi(x,y,z)]&\end{aligned} \tag{4}$$

With $a = 1$ and for a point charge at position (x_0, y_0, z_0), the discretized Poisson equation becomes:

$$\begin{aligned}\phi(x+1,y,z) + \phi(x-1,y,z) + \phi(x,y+1,z) + \phi(x,y-1,z) +& \\ \phi(x,y,z+1) + \phi(x,y,z-1) - 6\phi(x,y,z) = -q\delta_{x,x_0}\delta_{y,y_0}\delta_{z,z_0}&\end{aligned} \tag{5}$$

Where $\delta_{i,j}$ is the Kronecker delta, which equals 1 when $i = j$ and 0 otherwise.

2.3 Iterative Solution Method

Rearranging the equation, we obtain the iterative formula:

$$\begin{aligned}\phi_{i+1}(x,y,z) = \frac{1}{6}[&\phi_i(x+1,y,z) + \phi_i(x-1,y,z) + \phi_i(x,y+1,z) + \\ &\phi_i(x,y-1,z) + \phi_i(x,y,z+1) + \phi_i(x,y,z-1) - \\ &q\delta_{x,x_0}\delta_{y,y_0}\delta_{z,z_0}]\end{aligned} \tag{6}$$

This iterative approach, known as the Jacobi method, forms the basis of our numerical solution.

3 Experimental Setup

3.1 Implementation Approach

The implementation follows a parallel iterative approach, utilizing CUDA to accelerate the computation. The key aspects of the implementation include:

1. **Domain Discretization**: The cubic domain is discretized into a 3D lattice with dimensions $L \times L \times L$, with the point charge placed at the center.

2. **Boundary Conditions**: Fixed boundary conditions (Dirichlet) with potential $\phi = 0$ on all six faces of the cube.

3. **Parallel Computation**: Each CUDA thread computes the potential at one lattice point, with the 3D domain mapped to a 3D thread organization.

4. **Convergence Criteria**: Iterations continue until the sum of squared differences between consecutive iterations is below a threshold $\epsilon = 10^{-6}$ or until a maximum number of iterations is reached.

5. **Data Analysis**: After convergence, the potential is measured at different distances from the center and compared with the theoretical values from Coulomb's law.

3.2 Testing Methodology

To investigate how the solution approaches Coulomb's law as lattice size increases, we performed experiments with four different lattice sizes:

- $L = 8$
- $L = 16$
- $L = 32$
- $L = 64$

For each lattice size, we measured:

- Number of iterations required for convergence
- Convergence error
- Potential values at different distances from the center
- Processing time

4 Implementation Details

4.1 Source Code Structure

The implementation consists of the following key files:

- `poisson3d.cu`: Main CUDA source file containing both the host code and device kernels.
- `Input_L`: Input configuration files for each lattice size L (8, 16, 32, 64).
- `cmd`: Condor submission script for running jobs on the TWCP1 cluster.
- `generate_experiments.sh`: Shell script to automate generating experiment directories for all lattice sizes.
- `submit_all.sh`: Shell script to automate submitting all jobs to the Condor system.
- `collect_results.sh`: Shell script to collect and compile results from all experiments.

4.2 Key CUDA Implementation Features

The main CUDA implementation in `poisson3d.cu` consists of:

- **Host Code**: Handles memory allocation, data transfer, kernel launch, and result analysis.
- **poisson3D Kernel**: The core computational kernel that implements the Jacobi iteration method for the Poisson equation.
- **Convergence Computation**: Uses parallel reduction in shared memory to efficiently compute the convergence error.

4.3 Core Algorithm Components

The key algorithmic components of the implementation include:

1. **3D Thread Organization**: The computation domain is mapped to a 3D thread structure that reflects the problem's natural geometry.
2. **Source Term Handling**: A point charge is implemented at the center of the cube with a finite value in a single lattice point.

3. **Iterative Updates**: The implementation toggles between two arrays to avoid unnecessary memory copies during iteration.

4. **Parallel Reduction**: Error calculation is optimized using shared memory and parallel reduction.

The following code snippet illustrates the core of the iterative update in the kernel:

```
// Read from array A, write to array B
sum = A[idx_xm] + A[idx_xp] +
      A[idx_ym] + A[idx_yp] +
      A[idx_zm] + A[idx_zp];

// 3D Poisson equation discrete formula (6 neighbors)
B[idx] = (sum - source) / 6.0f;
```

Listing 1: Core iterative update in the poisson3D kernel

4.4 Thread and Block Configuration

A critical aspect of the implementation was finding an efficient thread and block configuration for 3D computation. After experimentation, the following configuration was used:

```
dim3 threads(8, 8, 8);

// Blocks calculated based on lattice size
dim3 blocks((L+7)/8, (L+7)/8, (L+7)/8);
```

Listing 2: Thread and block configuration

This configuration was chosen to:

- Stay within the 1024 threads per block limit of CUDA
- Provide good spatial locality for memory access
- Ensure efficient occupancy of the GPU

5 Results and Analysis

5.1 Convergence and Performance

The following table summarizes the convergence and performance metrics for different lattice sizes:

Lattice Size (L)	**Iterations to Converge**	**Final Error**
8	101	9.96×10^{-7}
16	424	9.87×10^{-7}
32	1854	9.94×10^{-7}
64	7621	9.98×10^{-7}

As expected, the number of iterations required for convergence increases with the lattice size. This is because the information must propagate across the entire domain, and larger domains require more iterations for this propagation to reach all points.

5.2 Potential vs. Distance

The following table compares the numerical results for different lattice sizes with the theoretical value from Coulomb's law at selected distances:

Distance (r)	**Theory** $\phi(r) = \frac{0.0796}{r}$	**L=8**	**L=16**	**L=32**	**L=64**
1.0	0.0796	0.0648	0.0767	0.0816	0.0839
2.0	0.0398	0.0161	0.0279	0.0328	0.0351
3.0	0.0265	0.0066	0.0170	0.0219	0.0242

These results clearly show that as the lattice size increases, the numerical solution approaches the theoretical Coulomb law values. The L=64 case provides results that are very close to the theoretical predictions, particularly at short distances.

5.3 Approach to Coulomb's Law as L Increases

To visualize how the numerical solution approaches Coulomb's law as L increases, we plotted the relative error between the numerical and theoretical values for different lattice sizes:

Distance (r)	**Error L=8 (%)**	**Error L=16 (%)**	**Error L=32 (%)**	**Error L=64 (%)**
1.0	-18.6%	-3.7%	+2.5%	+5.3%
2.0	-59.5%	-29.9%	-17.6%	-11.8%
3.0	-75.0%	-35.8%	-17.4%	-8.8%

These error calculations demonstrate a systematic improvement in accuracy as the lattice size increases. For L=64, the error at r=1.0 is only +5.3%, and even at larger distances, the error significantly decreases compared to smaller lattice sizes.

6 Discussion

6.1 Convergence to Coulomb's Law

The primary question of this assignment was whether the numerical solution approaches Coulomb's law in the limit of large L. Based on our results, we can confidently answer yes. The numerical solution shows clear convergence toward the theoretical $\frac{1}{4\pi r}$ relationship as L increases. This convergence is evident from:

- Decreasing relative error with increasing L
- Systematic approach to theoretical values across all distance ranges
- More accurate representation of the $\frac{1}{r}$ falloff for larger L values

For L=64, the numerical solution comes remarkably close to the theoretical values, with errors of less than 10% for most distances, confirming the convergence behavior.

6.2 Factors Affecting Accuracy

Several factors influence the accuracy of the numerical solution:

- **Lattice Resolution**: Finer lattices (larger L) better resolve the spatial variations in the potential, particularly near the charge where gradients are steep.
- **Boundary Conditions**: In our numerical solution, we use a finite cube with zero-potential boundaries, whereas Coulomb's law assumes an infinite domain. This difference creates boundary effects that are more pronounced for smaller domains.
- **Discretization Error**: The finite difference approximation introduces errors that decrease with lattice refinement.
- **Point Charge Representation**: Representing a point charge on a discrete lattice involves approximations that affect nearby field values.

6.3 Computational Challenges

The implementation revealed several computational challenges:

- **Iteration Count Scaling**: The number of iterations required for convergence appears to scale approximately as $O(L^2)$, making very large simulations computationally expensive.
- **Memory Requirements**: The memory footprint scales as $O(L^3)$, limiting the maximum size of simulations.
- **Thread Organization**: Finding an optimal thread configuration for 3D problems required careful consideration of CUDA hardware limitations.
- **Convergence Rate**: The Jacobi method has relatively slow convergence. More advanced methods like Successive Over-Relaxation (SOR) or multigrid methods could provide faster convergence.

7 Conclusion

This study demonstrates the successful application of CUDA parallel programming to solve the 3D Poisson equation for a point charge in a cube with fixed boundary conditions. Our results confirm that as the lattice size increases, the numerical solution indeed approaches Coulomb's law.

For the largest lattice size tested (L=64), the numerical solution shows excellent agreement with the theoretical predictions, with errors less than 10% for most of the domain. This confirms the expected physical behavior and validates our numerical approach.

The implementation highlights the power of GPU computing for solving partial differential equations, as well as the importance of proper thread organization and memory management for 3D problems. While the current implementation uses the simple Jacobi method.