Introduction to CUDA Parallel Programming Homework Assignment 3

- March, 2025
- NTNU
- 41173058H

Problem Statement

Solve the Poisson equation on a 3D lattice with boundary conditions. Consider a cube of size $L \times L \times L$ with a point charge $q = 1$ at its center $(L/2, L/2, L/2)$, with lattice sites $(0, 1, 2, ..., L)$ in each direction, subject to boundary conditions with potential equal to zero on its entire surface. Find the potential versus the distance r from the point charge, for $L = 8, 16, 32, 64, 128, 256$ respectively. Determine if the potential approaches Coulomb's law in the limit $L \gg 1$.

Source Code Analysis

Numerical Method Implementation

The implementation uses the finite difference method to solve the 3D Poisson equation:

$\nabla^2 \phi = -\rho/\epsilon_0$

Core Algorithm: 6-Point Stencil

```
__global__ void poissonKernel(float *potential, float *new_potential, bool *convergence,
                             int L, float tolerance) {
    int idx = blockIdx.x * blockDim.x + threadIdx.x;
    int idy = blockIdx.y * blockDim.y + threadIdx.y;
    int idz = blockIdx.z * blockDim.z + threadIdx.z;

    if (idx > 0 && idx < L-1 && idy > 0 && idy < L-1 && idz > 0 && idz < L-1) {
        int index = idx + L * idy + L * L * idz;

        // 6-point stencil for Laplacian operator
        float new_val = (potential[index + 1] + potential[index - 1] +
                        potential[index + L] + potential[index - L] +
                        potential[index + L*L] + potential[index - L*L]) / 6.0f;

        new_potential[index] = new_val;

        // Convergence check with tolerance
        if (fabsf(new_val - potential[index]) > tolerance) {
            *convergence = false;
        }
    }
}
```

Key Implementation Features:

1. **3D Indexing**: Uses `idx + L*idy + L*L*idz` for linearized 3D array access
2. **Boundary Handling**: Excludes boundary points (0 and L-1) from computation
3. **Jacobi Iteration**: Updates all points simultaneously using previous iteration values
4. **Convergence Detection**: Global convergence flag updated atomically

Memory Management Strategy

```
// 3D thread block configuration
dim3 block(threadsPerBlock, threadsPerBlock, threadsPerBlock);
int numBlocks = (L + threadsPerBlock - 1) / threadsPerBlock;
dim3 grid(numBlocks, numBlocks, numBlocks);
```

The implementation uses 3D CUDA blocks and grids to naturally map to the 3D problem domain, ensuring optimal memory access patterns and thread utilization.

Initialization and Boundary Conditions

```
void initializePotential(float *potential, int L) {
    int size = L * L * L;
    for (int i = 0; i < size; i++) {
        potential[i] = 0.0f;  // Zero boundary conditions
    }

    // Point charge at center
    int center = L/2;
    potential[center + L * center + L * L * center] = 1.0f;
}
```

Results and Analysis

Performance Analysis by Grid Size

Convergence Behavior Analysis The results reveal a critical issue with the convergence detection algorithm. Most configurations show **premature convergence** (1 iteration) with identical error values across different L sizes, indicating a bug in the convergence checking mechanism.

L Size	Optimal Block	Grid Size	Kernel Time (ms)	Iterations	Max Error
8	16^3	1^3	0.012	1	7.96×10^{-2}

L Size	Optimal Block	Grid Size	Kernel Time (ms)	Iterations	Max Error
16	12^3	2^3	0.015	1	7.96×10^{-2}
32	16^3	2^3	0.023	1	7.96×10^{-2}
64	12^3	6^3	0.098	1	7.96×10^{-2}
128	16^3	8^3	0.099	1	7.96×10^{-2}
256	12^3	22^3	0.101	1	7.96×10^{-2}

Block Size Performance Characteristics **Small Block Sizes** ($2 \times 2 \times 2$ **to** $4 \times 4 \times 4$):

- **Poor Performance**: Extremely long execution times (up to 7500ms for L=256)
- **High Iteration Count**: Millions of iterations before convergence
- **Low GPU Utilization**: Insufficient parallelism to saturate GPU resources

Medium Block Sizes ($6 \times 6 \times 6$ **to** $10 \times 10 \times 10$):

- **Transition Zone**: Some configurations show proper convergence behavior
- **Variable Performance**: Inconsistent results across different L values

Large Block Sizes ($12 \times 12 \times 12$ **to** $16 \times 16 \times 16$):

- **Optimal Performance**: Sub-millisecond execution times
- **Suspicious Convergence**: All show 1 iteration, indicating algorithmic issues
- **Resource Efficiency**: Good GPU utilization but questionable numerical accuracy

Critical Algorithm Issues

Convergence Detection Problem The identical error values (7.96×10^{-2}) across all L sizes and the premature convergence suggest:

1. **Global Convergence Flag Issue**: The `*convergence = false` operation may not be working correctly across all threads
2. **Race Condition**: Multiple threads writing to the same convergence flag without proper synchronization
3. **Tolerance Setting**: The tolerance (1×10^{-6}) may be inappropriate for the problem scale

Recommended Fixes

```
// Use atomic operations for convergence checking
__global__ void poissonKernel(float *potential, float *new_potential,
                              float *max_diff, int L) {
    // ... existing code ...
    // Use shared memory for block-level reduction
    extern __shared__ float sdata[];
    float local_diff = fabsf(new_val - potential[index]);

    // Block-level reduction to find maximum difference
    // Then atomic max operation to global memory
    atomicMax(max_diff, local_diff);
}
```

Physical Analysis: Approach to Coulomb's Law

Theoretical Background For a point charge $q = 1$ in free space, Coulomb's law gives:

$V(r) = \frac{q}{4\pi\varepsilon_0 r} = \frac{1}{4\pi r}$ (in Gaussian units)

Finite Size Effects **Boundary Influence**:

- **Near Field ($r \ll L$)**: Minimal boundary effects, should approach Coulomb's law
- **Far Field ($r \approx L/2$)**: Strong boundary effects due to zero potential constraint
- **Convergence Rate**: Expected $O(1/L^2)$ convergence to analytical solution

Grid Resolution Effects:

- **Discretization Error**: $O(h^2)$ where $h = 1$ is the grid spacing
- **Numerical Dispersion**: High-frequency modes poorly resolved on coarse grids

Expected Results vs. Observed The identical error values across all L sizes indicate the algorithm is not properly converging to the correct solution. For proper implementation, we should observe:

1. **L=8**: Significant deviation from Coulomb's law due to boundary proximity
2. **L=16,32**: Improved accuracy in central region
3. **L=64,128,256**: Excellent agreement with $1/(4\pi r)$ for $r < L/4$

Data Output Analysis

The code generates `potential_L*.dat` files containing:

```
fprintf(fp, "%f %f %f\n", r, numerical, analytical);
```

This provides distance r, numerical solution, and analytical Coulomb potential for comparison. However, given the convergence issues, these files likely contain incorrect numerical results.

Discussion

Algorithm Correctness Issues

The current implementation has several critical problems:

1. **Convergence Detection**: The global convergence flag mechanism is flawed
2. **Numerical Stability**: Single precision may be insufficient for tight tolerance
3. **Boundary Conditions**: Point charge initialization may create numerical instabilities

Performance Optimization Opportunities

1. **Shared Memory Utilization**: Current implementation doesn't use shared memory for stencil operations
2. **Memory Coalescing**: 3D access patterns may not be fully coalesced
3. **Occupancy Optimization**: Block sizes should be tuned for specific GPU architecture

Recommended Improvements

Algorithmic Enhancements

```
// Use reduction-based convergence checking
__global__ void computeResidual(float *potential, float *new_potential,
                                float *residual, int L) {
    // Compute local residual
    // Use block-level reduction
    // Store per-block results for CPU reduction
}
```

Numerical Stability

- Use double precision for accumulation
- Implement successive over-relaxation (SOR) for faster convergence
- Add proper error handling and validation

Physical Interpretation

Despite the implementation issues, the theoretical expectation is clear: as L increases, the numerical solution should converge to Coulomb's law $V = 1/(4\pi r)$ in regions where $r \ll L$. The boundary effects become negligible for large L, and the discrete Laplacian operator approaches the continuous operator.

Conclusion

While the CUDA implementation demonstrates proper 3D parallelization techniques and efficient memory management, critical algorithmic issues prevent meaningful physical analysis. The convergence detection mechanism requires fundamental revision to produce reliable results. Once corrected, the implementation should successfully demonstrate that the finite difference solution approaches Coulomb's law in the limit $L \gg 1$, with convergence rate proportional to $1/L^2$.

The performance analysis reveals that medium-to-large block sizes (12^3 to 16^3) provide optimal GPU utilization, but the numerical accuracy must be validated through proper convergence testing before drawing physical conclusions.

Submission Guidelines

Submit your homework report including source codes, results, and discussions. Prepare the discussion file using a typesetting system (LaTeX, Word, etc.) and convert to PDF. Compress all files into a gzipped tar file named with your student number and problem set number (e.g., `r05202043_HW3.tar.gz`). Send your homework with the title "your_student_number_HW3" to twchiu@phys.ntu.edu.tw before 17:00, June 11, 2025.