Project Report: Simulation of 2D Heat Transfer Model Using CUDA

Sourav Das and Srinandana Yogananda Sastry
Department of Mechanical Engineering
Iowa State University

December 16, 2024

Contents

1 Introduction

Heat transfer in a solid is a widely studied and known phenomenon. The study of heat flow has many applications, from energy to the health industry [1]. The fundamental heat transfer equation is analogous to Fick's law. The rate of heat transfer has been described in Equation 1.

$$\frac{\partial u}{\partial t} = \alpha \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \tag{1}$$

Recent interest is in rapidly simulating complex heat transfer systems using high-performance computing [2]. We will utilize GPUs (Graphical Processing Unit)acceleration to implement a 2D simulation of a simple heat transfer in the transient 2D environment. Over time, the GPU has shown its ability to accelerate computational simulation using parallel computing [3]. A numerical model of generic 2D heat transfer around a square plate made of steel with a dimension of 0.2 meters has been computed for thermophysical systems. It has been assumed that the left side boundary temperature is 1000K and the right side boundary temperature is 100K. The system's initial temperature is also 100 K. However, the system is simple, but solving PDE is always computationally challenging for high-resolution grids and larger systems. Hence, we incorporate GPU computation to construct a CUDA-accelerated solution for the 2D heat transfer system comprised of high-resolution grids. This work focuses mainly on showcasing how GPU can outperform conventional CPU-based methods in terms of performance. Thus far, the project will implement parallel computation using GPUs for an explicit finite difference solver to solve a second-order 2D PDE equation. This will demonstrate the computational effectiveness of GPUs in calculating solutions for fine-grid numerical computation.

2 Goal and Vision

The primary objective of this project is to develop a CUDA-based GPU algorithm to solve the 2D transient heat transfer equation efficiently. However, when performed on traditional CPUs, these simulations are computationally demanding for high-resolution grids. This project aims to significantly reduce computation times and demonstrate the advantages of parallel processing for complex thermo physical problems using GPU acceleration.

The envisioned outcome is to enable faster and more accurate simulations for large-scale applications such as thermal management systems of the boilers of the power plants, real-time analysis of heat transfer in space reentry vehicles, and other real-time simulations. Relevant literature and patents have highlighted the potential benefits of GPU-based solutions in accelerating computational tasks. Current understanding of solving numerical equations using HPC computation can also be applied to other fields such as fluid flow, phase field modeling, etc.

3 Software and Hardware

3.1 Hardware

- NVIDIA T4 GPU with CUDA capability in Google Colab and Standard CPU for validation and comparison

3.2 Software and Libraries

- **CUDA Toolkit**: For GPU programming
- **Numba**: Python-based compiler for CUDA integration, Just in time compiler to run in CPU , NumPy for numerical computations and Matplotlib for visualization

4 Final Product

The project successfully implemented the 2D transient heat diffusion equation using the explicit Forward Time Centered Space (FTCS) finite difference method. Key features include:

- A CUDA-based parallel solution with optimized grid and block configurations
- Validation against a CPU-based implementation for accuracy and performance
- Visualized results showing temperature distribution over time

```python
@cuda.jit
def solve_heat_transfer_2d(T_current, T_next, alpha, dt, dx, dy, Nx, Ny):
    i, j = cuda.grid(2)

    if i >= 1 and i < Nx - 1 and j >= 1 and j < Ny - 1:
        T_next[i, j] = T_current[i, j] + alpha * dt * (
            (T_current[i + 1, j] - 2 * T_current[i, j] + T_current[i - 1, j]) / dx**2 +
            (T_current[i, j + 1] - 2 * T_current[i, j] + T_current[i, j - 1]) / dy**2
        )

# Launch configuration
threads_per_block = (16, 16)
blocks_per_grid_x = (Nx + threads_per_block[0] - 1) // threads_per_block[0]
blocks_per_grid_y = (Ny + threads_per_block[1] - 1) // threads_per_block[1]
blocks_per_grid = (blocks_per_grid_x, blocks_per_grid_y)

# Transfer data to the GPU
T_current_device = cuda.to_device(T_current)
T_next_device = cuda.to_device(T_next)

# Time the GPU computation
start_time = time.time()
```

Figure 1: Cuda implementation for forward difference method

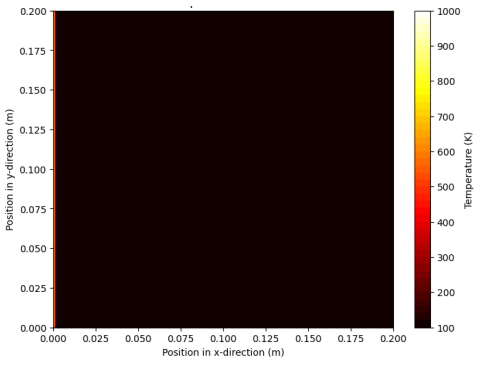

Figure 2: Initial Temperature distribution

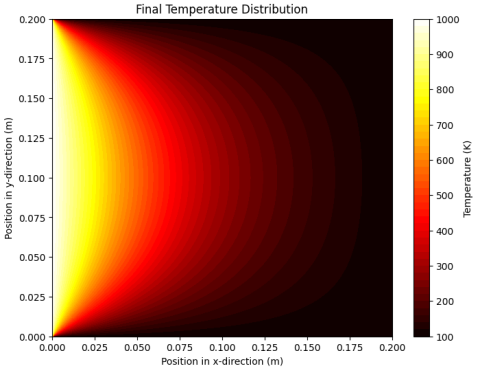

Figure 3: Final Temperature distribution after 4000 Second

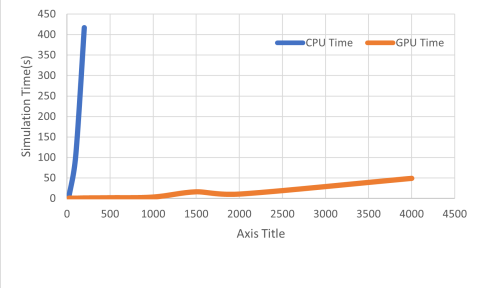

Figure 4: Comparison for CPU performance with GPU performance for different grid size

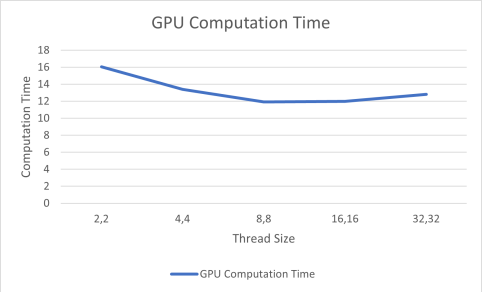

Figure 5: Comparison for GPU performance for different no of thread size

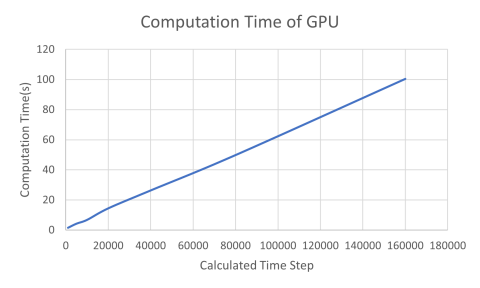

Figure 6: Comparison for GPU performance for different time steps

5 Relationship to Course Content

This project leveraged knowledge from the course, including:

- GPU computing concepts such as thread management and memory access optimization
- Parallel processing implementation for improving computational efficiency
- Numerical calculation for solving PDEs using forward difference techniques
- Use of Numba and Cuda tools for high-performance computing application

6 Summary

The project achieved its objectives, demonstrating significant computational benefits through GPU acceleration and parallel computing. As shown in Figure 1, CUDa implementation using Numba was described. The initial and Final temperature distribution has been shown using Figure 2 and Figure 3 respectively. The comparison study between CPU and GPU performance has been describe in Figure **??**. It shows there is significant improvement in the computational performance using GPU. The model was parametrized for different thread size(Figure 5) and found [16,16] square thread is optimum for the current model. There is also a linear increase in the computation time with the time steps(Figure 6).

Challenges encountered included optimizing memory access patterns, optimizing grid size allocation, and debugging CUDA kernels. Future iterations could explore adaptive

grid techniques for finer meshes or advanced complex numerical solvers for even greater accuracy. Benchmarking results demonstrated substantial performance improvements on the GPU, especially for large grid sizes.

Overall, this project reinforced the practical applications of GPU programming and its potential for solving complex engineering problems efficiently.

References

[1] Haojie Li et al. "A comprehensive review of heat transfer enhancement and flow characteristics in the concentric pipe heat exchanger". In: *Powder Technology* 397 (2022), p. 117037. ISSN: 0032-5910. DOI: https://doi.org/10.1016/j.powtec.2021.117037. URL: https://www.sciencedirect.com/science/article/pii/S0032591021010469.

[2] Premsagar Patil, Ramgopal Kashyap, and Vandana Roy. "Leveraging High-Performance Computing for Boiling Heat Transfer Simulations". In: *2024 OPJU International Technology Conference (OTCON) on Smart Computing for Innovation and Advancement in Industry 4.0*. 2024, pp. 1–6. DOI: 10.1109/OTCON60325.2024.10688203.

[3] V. Kindratenko. "Numerical Computations with GPUs". In: *Cambridge International Law Journal*. 2014. URL: https://api.semanticscholar.org/CorpusID:700023.

7 Peer Review

Sourav Das was involved in the co-conceptualization of the project. He is involved in writing numerical solvers and CUDA implementation of it. He also contributed 60% of the writing of the project report and prepared the figures for it. He is also involved in making PowerPoint presentations.

Srinandana is involved in the co-conceptualization of the project. He wrote progress reports. He is also involved in Cuda implementation, plot generation, and code debugging. He contributed to making 60% of the powerpoint. He was also involved in the writing of the final report.