SCHOOL OF COMPUTER SCIENCE
HIGH PERFORMANCE ARCHITECTURE (COMPSCI 243)

GPU Benchmarking Report

Zahra Montazeri (84168407)
Nitin Agarwal (84246130)
May 31, 2016

CONTENTS

1 VISION AND SCOPE

In this project, a set of well-defined test cases for GPU have been used to compare the performances of different graphics cards across differnt platforms. The main focus of the project has been to use the various test cases with similar settings on multiple graphics cards, keeping everything else the same. We perform benchmarking on three different NVIDIA Graphic Cards and profile the test cases using an NVIDIA profiler **nvprof**. We show the results of each of the test cases and further perform analysis.

The project involved searching of well-known, published test cases, which could be tested on Linux platforms. All benchmarks have been performed using the release 3.1 version of the Rodinia **??**. Several test cases from Rodinia were profiled using nvprof (NVIDIA profiling tool), which is a command-line profiler available for Linux, Windows, and OS X. It helps users to understand and optimize the performance of cuda applications and easily compare and analyze the result of benchmarks on different GPUs. A cuda version of 7.5 was used to benchmark all three graphic cards in this project. The three NVIDIA graphics cards benchmarked are GTX 630, GTX 650 & Quadro M6000.

We chose seven different test cases and profiled several parameters for each test case. We compare and analyze these values across all three different GPU cards. We profiled the following parameters:

- *Total time for all API calls:* explanation??
- *Total time for all Kernel calls:* explanation??
- *Number of registers per thread for each Kernel call:* explanation?
- *Temperature:* We understand that temperature and fan-speed would be more relevant in for benchmarking mobile GPU's, but for completeness we profiled them both.
- *Fan Speed: All the graphics cards requires a robust fan to keep it from overheating under a load. This parameter is the percentage of the total amount of fan power depending on how much you tax the graphics card in the system. You may want to adjust its fan speed to provide better cooling or produce less noise when not pushing it to its limits.*

2 Description of Benchmarks

This section provides a brief description of all the seven test cases used to benchmark the three NVIDIA GPU cards.

2.1 Breadth First Search

Graph algorithms are fundamental and widely used in many disciplines and application areas. Large graphs involving millions of vertices are common in scientific and engineering applications. This benchmark suite provides the GPU implementations of breadth-first search (BFS) algorithm which traverses all the connected components in a graph.

2.2 K-Nearest Neighbor

NN (Nearest Neighbor) finds the k-nearest neighbors from an unstructured data set. The sequential NN algorithm reads in one record at a time, calculates the Euclidean distance from the target latitude and longitude, and evaluates the k nearest neighbors. The parallel versions read in many records at a time, execute the distance calculation on multiple threads, and the master thread updates the list of nearest neighbors.

2.3 B+Tree

B+ Tree application has many internal commands that maintain database and process querries. Only J and K commands had enough parallelism to be ported to parallel languages (OpenMP, CUDA, OpenCL). In these implementations, in case of both J and K, the same algorithms (optimized for exposing fine-grained parallelism) were used for fair comparison purposes. For C/OpenMP execution, it is possible to use the original algorithm.

2.4 PathFinder

PathFinder uses dynamic programming to find a path on a 2-D grid from the bottom row to the top row with the smallest accumulated weights, where each step of the path moves straight ahead or diagonally ahead. It iterates row by row, each node picks a neighboring node in the previous row that has the smallest accu- mulated weight, and adds its own weight to the sum. This kernel uses the technique of ghost zone optimization.

2.5 LU Decomposition

LU Decomposition is an algorithm to calculate the solutions of a set of linear equations. The LUD kernel decomposes a matrix as the product of a lower triangular matrix and an upper triangular matrix.

2.6 GAUSSIAN ELIMINATION

Gaussian Elimination computes result row by row, solving for all of the variables in a linear system. The algorithm must synchronize between iterations, but the values calculated in each iteration can be computed in parallel.

2.7 HOTSPOT

HotSpot is a widely used tool to estimate processor temperature based on an architectural floorplan and simulated power measurements. The thermal simulation iteratively solves a series of differential equations for block. Each output cell in the computational grid represents the average temperature value of the corresponding area of the chip. Our CUDA implementation re-implements the transient thermal differential equation solver from HotSpot.

3 METHODOLOGY

The main objective of this project is to use same test cases, with similar settings on three different NVIDIA GPU graphics cards. During benchmarking of the seven test cases, we made sure that no other programs, code etc were running on these three machines. We further ensured that all the settings were as similar as possible e.g for GPU architecture, cuda settings etc.

Instead of profiling the total execution time, which collectively sums up the CPU & GPU running time, we only profiled the time for all the API calls and the Kernel calls. Since we are only concerned with GPU benchmarking, this seems reasonable as all three machines have different processors (Table 3.1), which would lead to different CPU running times. We also profiled the number of registers used per thread for each kernel call in each program. And lastly we profiled the GPU temperature and fan speed. Though we understand that for desktop machine people are not that concerned about them, however when compiling different servers temperature and fan speed play an important role.

3.1 GPU PARALLELISATION

The GPU computing approach uses graphic card to perform the calculation in parallel. This approach is based on CUDA which is made by NVIDIA and works on NVIDIA graphic cards that have Compute Capability 2.0 or higher. The point is worth mentioning that only hydrodynamic calculations are performed on the GPU and the additional calculations are run on the CPU. Then these calculations are parallelised using shared memory approach, OpenMP.

3.2 HARDWARE

The following are the specifications of the three NVIDIA graphic cards which were used to perform benchmark:

Table 3.1: GPU specifications

GPU	Compute Capability	Number of CUDA cores	Memory (GB)
GeForce GT 630	2.1	96	1.8
Geforce GTX 650	3.0	384	5
Quadro 6000	2.0	448	6

The specifications of the machines in which the graphic cards resided are shown in the below table:

3.3 SOFTWARE

All benchmarks have been performed using the release 3.1 version of the Rodinia: accelerating Compute-Intensive Applications with Accelerators. It released to address some concerns in

Table 3.2: Hardware Platforms

	Processor	Memory (GB)	Operating system
1	Intel Xeon(R) E5405 (4 cores, 2.00GHz)	2	ubuntu 14.04
2	??	16	ubuntu 14.04
3	Intel Core 2 Quad Q9400 (4 cores, 2.66GHz)	7.8	ubuntu 14.04

which platforms face. For example, there are many suites for parallel computing on general-purpose CPU architectures, but accelerators fall into a gap that is not covered by previous benchmark development.

4 Benchmarking using a GeForce GT 630

These tests have been performed using a GeForce GT 630 and hardware platform 1 specified in Table 3.2. The below figures depict the time take for each individual API call for all seven test cases.

Figure 4.1: Time take for each API for seven test cases

5 Benchmarking using a GeForce GTX 650

These tests have been performed using a GeForce GTX 650 and hardware platform 2 specified in Table 3.2. The below figures depict the time take for each individual API call for all seven test cases.

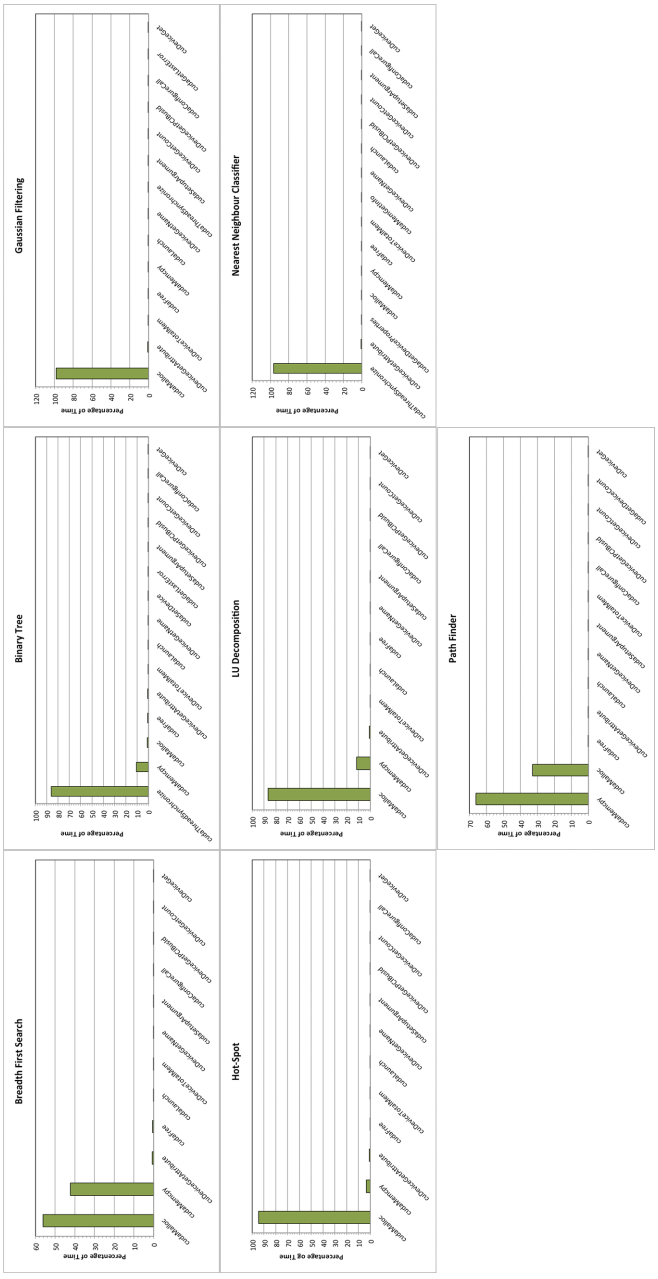

Figure 5.1: Time take for each API for seven test cases

6 Benchmarking using a Quadro 6000

These tests have been performed using a Quadro 6000 and hardware platform 3 specified in the table in Table 3.2. The below figures depict the time take for each individual API call for all seven test cases.

Figure 6.1: Time take for each API for seven test cases

7 Results

The below figures summarize all the results from all three different NVIDIA GPUs.

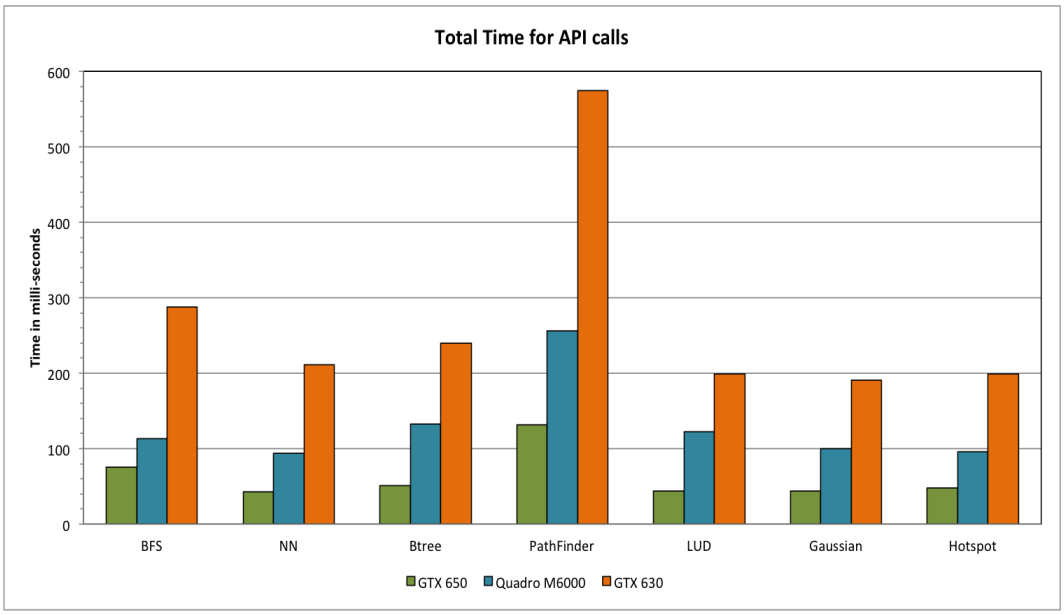

Figure 7.1: Total time for API calls for seven test cases

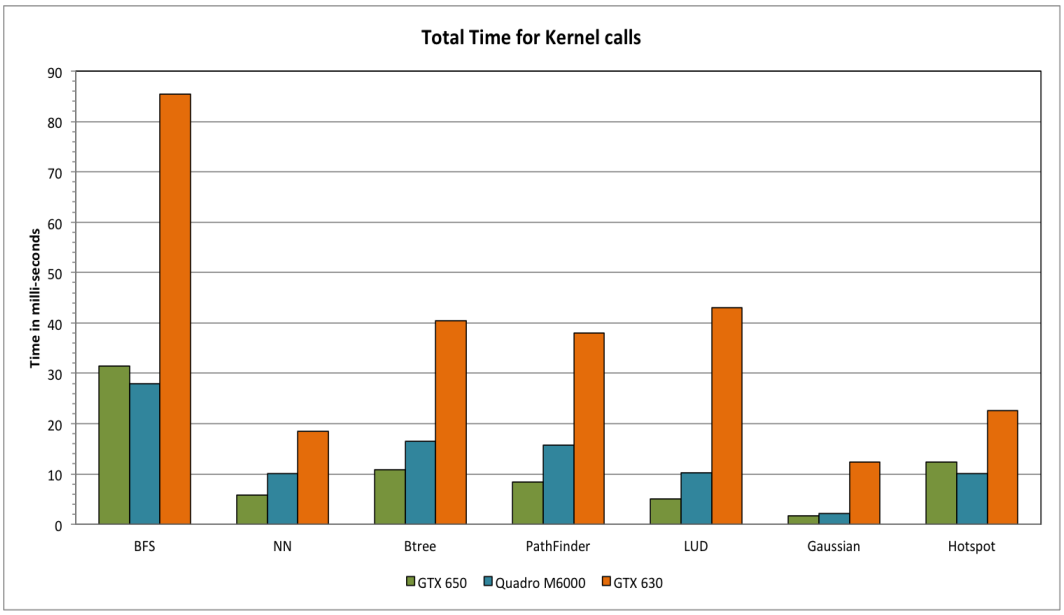

Figure 7.2: Total time for kernel calls for seven test cases

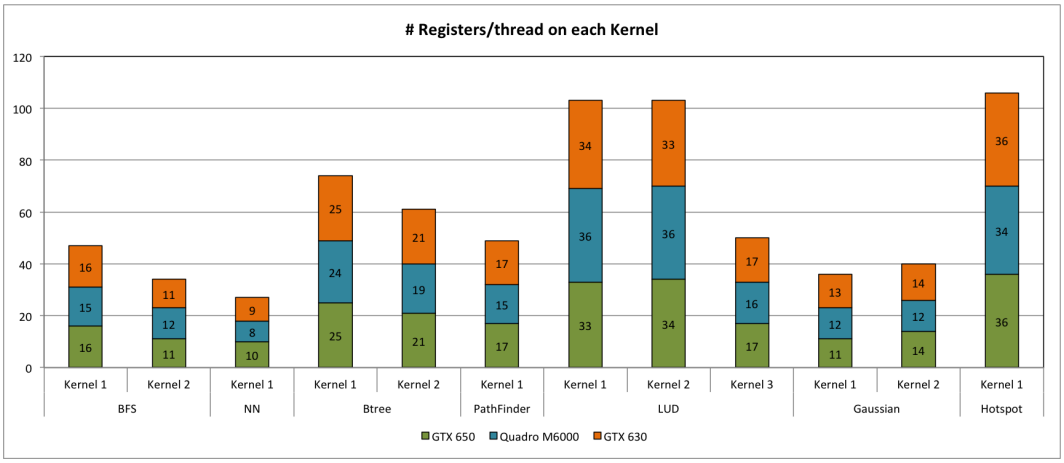

Figure 7.3: Number of registers per thread on each kernel for seven test cases

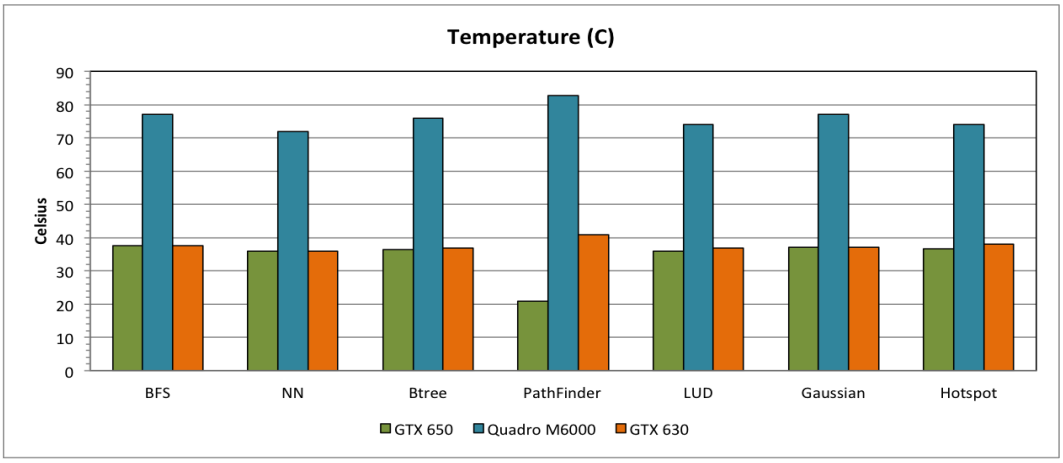

Figure 7.4: Temperature (C) for seven test cases

8 Analysis

.....

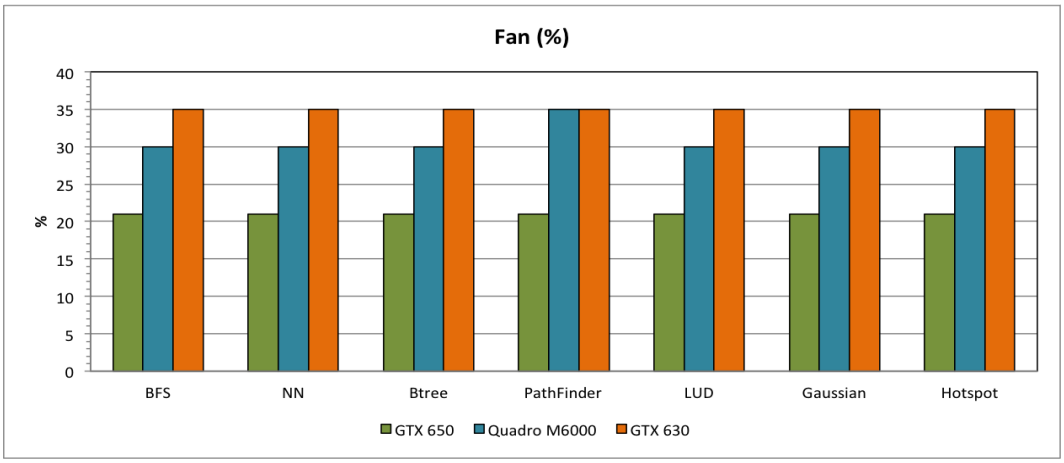

Figure 7.5: Fan spean for seven test cases

9 Conclusion

Though the test cases ran for several seconds ..if they ran for minutes the differences will be even more profound