Journal of Electrical Engineering and Computer Science
Issn: 2287-106x
Volume: 2, issue: 1, march,2020

Equidistant execution of resemblance computation on GPU construction using CUDA

Harshat yadav[1], Manoj singh[2], Minakshi kumar[3], Rakesh sharma[4]

Department of Computer Science and Engineering College of Engineering Roorkee, Roorkee-247667, INDIA[1,2,4]

Department of Electrical and Engineering Gautam Budha University, Greater Noida, INDIA[3]

Abstract- Picture handling and example acknowledgment calculations set aside more effort for execution on a solitary center processor. Illustrations Processing Unit (GPU) is progressively mainstream now-a-days because of their speed, programmability, minimal effort and progressively inbuilt execution centers in it. A large portion of the analysts began work to utilize GPUs as a handling unit with a solitary center PC framework to speedup execution of calculations and in the field of Content based clinical picture recovery (CBMIR), Euclidean separation and Mahalanobis assumes a significant job in recovery of pictures. Separation recipe is significant on the grounds that it assumes a significant job in coordinating the pictures. Right now, we parallelized Euclidean separation calculation on CUDA. CPU with Intel® Dual-Core E5500 @ 2.80GHz and 2.0 GB of principle memory which run on Windows XP (SP2). The following stage was to change over this code in GPU position for example to run this program on GPU NVIDIA GeForce arrangement 9500GT model having 1023 MB of video memory of DDR2 type and transport width of 64bit. The realistic driver we utilized is of 270.81 arrangement of NVIDIA. Right now, the CPU and GPU form of calculation is being actualized on the MATLAB R2010. The CPU adaptation of the calculation is being dissected in straightforward MATLAB however the GPU variant is being executed with the assistance of moderate programming Jacket-win-1.3.0. For utilizing Jacket, we need to make a few changes in our source code so to make the CPU and GPU to work at the same time and accordingly lessening the by and large computational speeding up. Our work utilizes broad use of exceptionally multithreaded design of multicored GPU. An effective utilization of shared memory is required to enhance equal decrease in Compute UnifiedGadget Architecture (CUDA), Graphic Processing Units (GPUs) are developing as amazing equal frameworks at a modest expense of two or three thousand rupees.

Keywords: Euclidean distance, Mahalanobis Distance, Content Based Medical Image Retrieval (CBMIR), CUDA, GPU, Parallelization

1. Introduction

Content-based image retrieval (CBIR) has gained considerable attention especially in the last decade. Image retrieval based on content is extremely useful in several applications such as medicine, publishing and Kuldeep Yadav et al / Indian Journal of Computer Science and Engineering (IJCSE) ISSN : 0976-5166 Vol. 3 No. 1 Feb -Mar 2012 1 advertising, historical research, fashion and graphic design, architectural and engineering design, crime prevention etc [1-4]. In this paper we are focusing on medical Image retrieval and similarity measures. Numerous commercial and experimental CBIR systems are now available e.g. IBM's QBIC (Query by Image Content), Virage's VIR Image Engine, Excalibur's Image Retrieval Ware, or Columbia University's Web SEEK. Also, many web search engines are now equipped with CBIR facilities, for example Alta Vista.

2. BACKGROUND

In this section, we have discussed about the Jacket v 1.3.0 which is a Graphics Processors for general purpose computing. Over the past few years, specialized coprocessors from floating point hardware to field. programmable gate arrays have enjoyed a widening performance gap with traditional x86 based processors. Of these, graphics processing units (GPUs) have advanced at an astonishing rate, currently capable of delivering over 1 TFOPS of single precision performance and over 300 GFLOPS of double precision while executing up to 240 simultaneous threads in one low cost package. As such, GPUs have gained significant popularity as powerful tools for high performance computing (HPC) achieving 20100 times the speed of their x86 counterparts in applications such as physics simulation, computer vision, options pricing, sorting, and search. As with previous research compressed sensing studies based on Graphics Processing Units (GPUs) provide fast implementations. However, only a small number of these GPU-based studies concentrate on compressed sensing Since the GPU which we have taken (NVIDIA 9500 GT) is the most basis model has high portability and is easily available in present day laptop and desktops so can be implemented directly. However, synchronizing of host and device with suitable parallel implementation is the most challenging part. Which has been parallelized by us? We have broken the process in threads of blocks and managed those threads inside a special thread managing hardware called as GPU with the help of the environment and set of libraries provided by CUDA.

2.1 JACKET OVERVIEW

Jacket connects Matlab to the GPU. Matlab is a technical computing language that integrates computation, visualization and programming in an easy to use environment that has found wide popularity both in the industry and academia. It is used across the breath of technical computing applications including mathematical computations, algorithm development, data analysis, data visualization and application development. With the GPU as a backend computation engine, Jacket brings together the best of three important computational worlds: computational speed, visualization, and the user friendliness of M programming. Jacket enables developers to write and run code on the GPU in the native M language used in Matlab. Jacket accomplishes this by automatically wrapping the M language into a GPU compatible form. By simply casting input data to Jacket's GPU data structure, Matlab functions are transformed into GPU functions. Jacket also preserves the interpretive nature of the M language by providing real time, transparent access to the GPU compiler [5]

2.2 INTEGRATION WITH MATLAB

Once Jacket is installed, it is transparently integrated with the Matlab's user interface and the user can start working interactively through the Matlab desktop and command window as well as write M-functions using the Matlab editor and debugger. All Jacket data is visible in the Matlab workspace, along with any other Matlab matrices.

2.3 INTRODCTION TO NVIDIA CUDA ARCHITECTURE

CUDA™ is a general-purpose parallel computing architecture introduced by NVIDIA. It contains the CUDA Instruction Set Architecture (ISA) and parallel compute engine in the GPU. The CUDA architecture is programmed using C language, which can then be run with great performance on a CUDA enabled processor. CUDA-enabled GPUs have hundreds of cores that can collectively run thousands of computing threads. Each core has shared resources, including registers and memory. The on-chip shared memory

allows parallel tasks running on these cores to share data without sending it over the system memory bus [6]. Thread hierarchy, shared memories and barrier synchronization are the three key abstractions of CUDA. A kernel can be executed by a one dimensional or two-dimensional grids of multiple equally-shaped thread blocks. A thread block is a 3, 2 or 1-dimensional group of threads. Threads within a block can cooperate among themselves by sharing data through some shared memory and synchronizing their execution to coordinate memory accesses. Threads in different blocks cannot cooperate and each block can execute in any order relative to other blocks. The number of threads per block is therefore restricted by the limited memory resources of a processor core. CUDA kernel function is a fundamental building block of CUDA programs. When launching a CUDA kernel function, a developer specifies how many copies of it to run. We call each of these copies a task. Because of the hardware support of the GPU, each of these tasks can be small, and the developer can queue hundreds of thousands of them for execution at once. These tasks are organized in a two-level hierarchy, block and grid. Small sets of tightly coupled tasks are grouped into blocks. In a given execution of a CUDA kernel function, all blocks contain the same number of tasks. The tasks in a block run concurrently and can easily communicate with each other, which enables useful optimizations such as those of the section “Shared Memory”. GPU’s hardware keeps multiple blocks in flight at once, with no guarantees about their relative execution order. As a result, synchronization between blocks is difficult. The set of all blocks run during the execution of a CUDA kernel function is called a grid [7].

2.4 GPU DATA TYPES

Jacket provides GPU counterparts to MATLAB’s CPU data types, such as real and complex double, single, uint32, int32, logical, etc. Any variable residing in the host (CPU) memory can be cast to Jacket’s GPU data types. Jacket’s memory management system allocates and manages memory for these variables on the GPU automatically, behind the scenes. Any functions called on GPU data will execute on the GPU automatically without any extra programming, GPU function Jacket provides the largest available set of GPU functions in the world, ranging from functions like sum, sine, cosine, and complex arithmetic to more sophisticated functions like matrix inverse, singular value decomposition, Bessel functions, and Fast Fourier Transforms. The supported set of functions continues to grow with every release of Jacket (see the Function Reference Guide), runtime of jacket is the most advanced GPU runtime in the world, providing automated memory management, compile on the fly, and execution optimizations for Jacket enable code, Jacket’s Graphics Toolbox is the only tool in the world that enables a merger of GPU visualizations with computation. With Jacket a simple graphics command can be added at the end of a simulation loop to visualize data as it is being computed while maintaining performance, The Developer SDK makes integration of custom CUDA code into Jacket’s runtime very easy. With a few simple SDK functions, your CUDA code can benefit from the optimized Jacket platform. When Jacket applications have completed the development, test, and optimization stages and are ready for deployment, the Jacket MATLAB Compiler allows users to generate license free executables for distribution to larger user bases. (See the SDK and JMC Wiki pages) and Interactive help for any Jacket function is available using Jacket’s ghelp function [8].

3. IMPLEMENTATION

In this section, first, we introduce the general scheme for Euclidean Distance and Mahalanobis Distance. Then, we introduce our GPU implementation environment by first discussing why GPUs are a good fit for medical imaging applications and then presenting NVIDIA’s CUDA platform and GeForce 9500 GT architecture. Next, we talk about the CPU implementation environment. This is followed by description of

the test data used in the experiments. Finally, we provide the list of CUDA kernels used in our GPU implementation.

3.1 FLOWCHART

This Flowchart helps to explain the process diagrammatically. First, take input as an image, read it as a matrix. Then decomposes the image into blocks, then from blocks to many columns. This process is repeated for all the rows. Using CUDA we decompose it in threads. Then the basic operation is performed [11]

3.2 CPU IMPLEMENTATION ENVIRONMENT

The CPU version of Distance formula is implemented in Matlab with integration of jacket v1.3.0. The computer used for the sequential implementation is an Intel® Dual-Core E5500 @ 2.80GHz and 2.0 GB of main memory which run on Windows XP. The algorithm was implemented in both single threading mode and multi-threading mode. Open MP is used to implement the multi-threading part.

4. CONCLUSION

The work presented in this paper shows the difference in the processing time of CPU and GPU when implemented on images in compressed domain. It shows the difference in time for each image at different sizes by implementing them in the Euclidean Distance and Mahalanobis Distance formula. We used a NVIDIA GeForce series 9500GT model having 1023 MB of video memory which reduced the process time from 150ms to 15ms. We obtained the speedup of the system 10 x times.

5. REFERENCES

[1] W. M. Smeulders, M. Worring, S. Santini, R. Jain and A. Gupta, “Content-Based Image Retrieval at the End of Early Years,” IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol.22, No.12, 2000.

[2] J. P. Eakins and M. E. Graham, “A Report to the JISC Technology Applications Programme,”Institute for Image data Research, University of Northumbria at Newcastle, 1999.

[3] V. Castelli and L.D. Bergman (Editors), Image Databases: Search and Retrieval of Digital Imagery, J. Wiley &Sons, NY, 2002.

[4] Visual Communications and Image Processing, Touradj Ebrahimi, Thomas Sikora, Editors, Proceedings of SPIE Vol. 5150, 2003.

[5] Huiyu Zhou*, Abdul H. Sadka, Mohammad R. Swash, Jawid Azizi and Abubakar S. Umar: Content Based Image Retrieval and Clustering:A Brief Survey 2009 , Vol.1.

[6] Liwei Wang, Yan Zhang, Jufu Feng: On the Euclidean Distance of Images, 2005, vol.27, pg.-1334-1339.

[7] Sharadh Ramaswamy and Kenneth Rose: fast adaptive mahalanobis distance-based search and retrieval in image databases, Dec.2009, vol 18, pp. 1057-7149.

[8] T. Huang and X. S. Zhou, "Image retrieval with relevance feedback: From heuristic weight adjustment to optimal learning methods," in ICIP, 2001, vol. 3, pp. 2–5.

[9] Kuldeep Yadav, Ankush Mittal M. A. Ansari, Avi Srivastava: Parallel Implementation of Compressed Sensing Algorithm on CUDAGPU, IJCSIS-2011, vol.9, No. 3, pp. 112-119.

[10] NVIDIA CUDA Programming Guide,Version 2.2, page10,27-35,75-97,2009.

[11] Shih, S.-T., Chao, C.-Y., Hsu, C.-M. (2018) "A smart laboratory management supply chain model based on fuzzy technology", Technology Reports of Kansai University, vol.60, pp. 67-77.