CUDA for GPGPU Applications – A Survey

Sarabjeet Singh
Department of CSE
SBS State Technical Campus,
Ferozepur-152004
sarabjeet_singh13@yahoo.com

Satvir Singh
Department of ECE
SBS State Technical Campus,
Ferozepur-152004
satvir15@gmail.com

Vijay Banga
Department of ECE
Amritsar College of Engg.&
Tech.,Amritsar-143001
v_banga@rediffmail.com

Durlabh Chauhan
Department of ECE
SBS State Technical Campus,
Ferozepur-152004
abhishekthakur421@yahoo.com

***Abstract*–Since inception man is trying to speed up things that too in parallel. From bullock carts to supersonic, the journey is still going on. GPGPU (General Purpose Computing with Graphics Processing Unit) is the concept combined with CUDA (Compute Unified Device Architecture) for parallel processing. Numerous applications exist where CUDA has been used by researchers for parallel implementations. This paper surveys different applications where CUDA C has been used for parallel processing. In every sphere, parallel programming has its own advantages. It can be implemented in any sphere with restriction that parallel operations exist in the phenomenon. This paper discusses Fluid Simulation, Remote Sensing, Weather Forecasting, K-Means, Bayesian Computation, Molecular Dynamics, Artificial Intelligence Algorithms and Vehicle Routing where CUDA can be used for GPGPU. CUDA exploits the parallel processing power of GPU, that has number of cores to speed up calculations.**

***Index Terms*—Graphics Processing Unit, CUDA, Artificial Intelligence, Survey, GPGPU**

I. Introduction

Latest advancements in parallel computing exploit GPGPUs that have multi-core architecture which supports parallel computations especially required for graphical processing. They devote more transistors for arithmetic and logical operations as compared to data caching and flow control compared to a CPU. Due to processing demand GPUs have advanced rapidly and beating the CPUs in terms of number of cores and their computational power. As nVidia has launched CUDA software development kit in 2007, the use of GPU's computational powers for general purpose computing has become easy. It gives an API built upon the C language that can be used to write parallel computer programs. The GPU device operates as a coprocessor to the host i.e., CPU, running C program. This paper presents a survey on different problems where CUDA and GPU can be used for parallel processing.

II. FLUID Animation

Smoothed Particle Hydrodynamics (SPH) is a mesh free lagrangian particle method used for modeling flow of fluid. It was introduced by Lucy and Monaghan in 1977[1]. Due to certain disadvantages of existing methods SPH was investigated for scientific analysis of fluid flow, carried out with few particles. The first implementation of SPH on GPU was realized by T. Harada [1].

SPH is a relatively new Fluid Dynamics technique to simulate motion of fluid particles. SPH is a particle based parallelizable technique, hence, more suitable for GPU based architecture [2]. Equations of SPH need to be more intensive in computation; hence the animation's computational part may be executed in real time. However, the major hurdle in SPH based animation is the tuning of SPH parameters like smoothing radius and rest density. A small change in any of these parameter results into almost explosion of fluid. As no proper guidelines are available for tuning SPH, considerable amount of efforts are required to find these parameters using brute force approach.

Parallel Algorithm Design is the most important issue in any application development for CUDA. Even though most of the algorithms introduced till today are already designed for some of earlier parallel architectures, it is not possible to adopt them without modification for CUDA. Besides complexity of algorithm, many additional factors also contribute significantly in deciding execution performance of any parallel algorithm are Memory access pattern, inter-processor communication overheads, synchronization overheads, and the degree of parallelism.

CUDA can be used for implementation of stable fluids that deals with internal and moving boundaries. The parallel implementation of CUDA is much faster as compared to sequential implementation [3].

III. Leukocyte Tracking

This application demonstrates an urgent need for dramatic speedups. It is a useful case study as it illustrates many of the issues that other applications will face in trying to use many core systems. It is an application presenting nontrivial software engineering challenges as well as presenting a workload that is representative of a much broader class of applications. The detection and tracking algorithm originally implemented in MATLAB, and re-implemented in C resulted in a significant performance improvement. Researchers, further, improved the performance by accelerating the most computationally demanding stages using CUDA and, for comparison, OpenMP [4]. The speedups in performance clearly demonstrate the advantages of the throughput-oriented nature of GPUs. There are number of bottlenecks, in both hardware and software, whose elimination will enable significant speedups and simplify the development of efficient CUDA applications.

IV. Remote Sensing

New Earth Observation instruments that are expected to substantially increase their spatial and spectral resolutions, thus producing a nearly continual stream of computationally intense image processing data sets. There have been tremendous

advances not only expected in optical instruments, but also in remote sensing systems. Image processing in remote sensing is particularly time consuming, and can greatly benefit from high performance computing techniques and practices to speed up processing of this type of data. If multiple types of remote sensing data is available, computing resources have to be used to address problems for efficient data sharing and distribution, compression, processing, transmission and storage. The role of different types of HPC architectures depends heavily on the considered remote sensing application, cluster-based parallel computing seems appropriate for efficient information extraction from very large data archives comprising data sets already transmitted to Earth, whereas the time-critical constraints introduced by many remote sensing applications call for on-board and often real-time processing developments, only focused on data processing and compression. This includes specialized hardware architectures such as Field Programmable Gate Arrays (FPGAs) and GPUs [5].

V. Weather Forecasting

Timely weather predictions are particularly useful for severe weather events when life and property is at risk. To get weather predictions in time using latest advances in atmospheric sciences is a big challenge even on the fastest super computers. The most powerful supercomputers in the world are used for numerical weather forecasting. With the help of GPU, inherently parallel problems in weather forecasting can be solved effectively. GPUs have hundreds of parallel cores for execution of tens of thousands of parallel threads. Unlike traditional CPUs, GPUs are not optimized for a single thread performance. Instead, they are optimized for executing a large number of threads, simultaneously. Therefore, a single thread performance on a GPU is lower than that on a contemporary CPU. This difference is because the processor architectures require legacy software to be rewritten for efficient parallel execution on GPUs. GPUs have been used very successfully for numerous different computational problems. [6].

VI. K-Mean Computation

K-means algorithm is one of the famous unsupervised clustering algorithms. Nowadays, desktop computers are coming equipped with programmable GPUs with plenty powerful Single Instruction Multiple Data (SIMD) processors that can support parallel data processing and high-precision computation. With the rapid advance in GPUs performance, coupled with recent improvements in their programmability, made it possible to parallelize K-means with personal computers. In this algorithm, both data objects assignment and k centroids recalculation of traditional k-means are parallel performed on the GPU [7].

VII. Bayesian Computation

The package cudaBayesreg is used to implement a Bayesian multilevel model for the analysis of brain functional magnetic resonance imaging (fMRI) data in the CUDA environment. The statistical framework in cudaBayesreg is built around a Gibbs sampler for multilevel/hierarchical linear models with a normal prior. Multilevel modeling can be treated as a generalization of regression methods in which regression coefficients are themselves given a model with parameters estimated from data [8]. As in Statistical Parametric Map (SPM), the Bayesian model fits a linear regression model at each voxel, however uses multivariate statistics for parameter estimation at each iteration of the Markov Chain Monte Carlo (MCMC) simulation. The Bayesian model used in cudaBayesreg follows a two–stage Bayes prior approach to relate voxel regression equations through correlations between the regression coefficient vectors [9]. This model closely follows the Bayesian multilevel model proposed by Rossi, Allenby and McCulloch [10], and implemented in bayesm [11]. This approach overcomes several limitations of the classical SPM methodology. The SPM methodology traditionally used in fMRI has several important limitations mainly because it relies on classical hypothesis tests and p–values to make statistical inferences in neuroimaging [12], [13], [14]. However, as is often the case with MCMC simulations, the implementation of this Bayesian model in a sequential computer entails significant time complexity. The CUDA implementation of the Bayesian model has been able to reduce significantly the runtime processing of the MCMC simulations. The increased performance comes from the use of separate threads for fitting the linear regression model at each voxel in parallel.

VIII. Molecular Dynamics Simulation

Molecular Dynamics is a computationally intensive method for studying the natural time-evolution of a system of atoms using Newton's classical equations of motion. MD has always been limited more by the current available computing power. Researchers in this field have typically focused their efforts to simplify models and identify what can be neglected to still obtain acceptable results [15]. HPC on GPU is the key in making biologically relevant calculations tractable without compromise.

IX. Continuous Space Language Model

The Continuous Space Language Model (CSLM), introduced by Schwenk along with an open source implementation, provides an alternative to the n-gram back off model and allows "true interpolation" of the probabilities of unseen n-grams. The CSLM algorithm is highly computationally intensive and is a good candidate for implementation with CUDA. The multiplications in the hidden and output layer, both forward and backward pass, especially in bunch mode using large matrices, are highly parallel. However, there is overhead associated with using the GPU. Memory should be allocated on both the CPU as well as on the GPU. Variables used in the computation must be transferred to the GPU. The computation is then performed on the GPU, and the results must be transferred back to the host CPU. CUBLAS is a CUDA implementation of BLAS (Basic Linear Algebra Subprogram), which performs matrix multiplication operations. It is self-contained and requires no direct

interaction with the CUDA driver. Functions in the CUBLAS library provide matrix multiplications in an efficient manner and handle all overhead issues regarding programming of threads. Due to their simplicity of use, the CUBLAS libraries were used as the starting point for the introduction of CUDA to CSLM [16].

X. Vehicle Routing

Many of identical vehicles with a given capacity are located at a central depot. They need to service a set of customer orders. Each customer orders from a specific location and has specific size. Travel costs between different locations are available. The goal is to design a least-cost set of routes for all the vehicles so that all customers are visited once and vehicle capacities are adhered to. The effort is neither to propose a new, competitive Vehicle Routing Problem solution method, nor to prove once more that the GPU has high computing power. Rather, our main goal is to carefully asses how well local search implementation can fit the given hardware, identify limiting factors and to resolve them [17].

XI. Swarm Based Algorithms

Artificial Intelligence (AI) is one of the most dominating paradigms of modern research and inculcates computationally intensive algorithms. GPU is being investigated to increase performance of these AI algorithms and brief survey is given below:

A. *Ant Colony Optimization*

ACO parallel implementations can be divided into two general approaches. The first one is the parallel execution of the ants construction phase in a single colony. It aims to accelerate computations by distributing ants to computing elements[18]. The second that was introduced by Stützle is the execution of multiple ant colonies. In this case, entire ant colonies are allocated to processors in order to speed up computations as well as to potentially improve solution quality by incorporating cooperation schemes between colonies. These usually follow the message-passing and shared-memory computing paradigms. The ACO metaheuristic, the Max–Min Ant System (MMAS) algorithm and its application to the Traveling Salesman Problem (TSP). The selection of MMAS and TSP is to focus on algorithmic aspects of ACO and technical aspects of GPU computing that are not problem dependent and to compare with the work of Stützle and Hoos [19].

B. *Particle Swarm Optimization*

Advanced Driving Assistance Systems (ADAS) needs road sign detection. The published work shows that traffic signs can be first detected and then classified in video sequences in real time[]. While detection is performed using computer vision techniques based on color and/or shape matching. A novel approach based on both sign shape and color which uses Particle Swarm Optimization (PSO) for detection can be used. A single fitness function may be used both to detect a sign belonging to a certain category and, in parallel, to estimate its actual position with respect to the camera reference frame. To speed up execution time, the algorithm exploits the parallelism available with GPUs these days [20].

C. *Genetic Algorithms*

Genetic Algorithms (GA) is candidate for parallel computing since population members fitness can be evaluated in parallel. GA uses a number of other operations which, if performed in parallel, can lead to faster evolutionary performance. The binary and real-coded genetic algorithms using CUDA may be executed in parallel. The bottlenecks in a parallel GA implementation can be identified and modified suitably. The results are compared with the sequential algorithm for accuracy and clock time for varying problems by studying the different parameters population sizes, number of threads, problem sizes. Significant speed-ups results over the sequential GA [21].

XII. Conclusions

All applications, discussed, in the paper, resulted in tremendous speedup in performance. Leukocyte Tracking implementation achieved an overall speedup of 199.9x using a desktop system with an NVIDIA GeForce GTX 280 GPU, as compared to a speedup of 7.6x on the fastest available multicore CPU system. The new Molecular Dynamics algorithm using CUDA leads to a performance improvement. Weather forecasting resulted in a reduction in processing time from 16928 ms on CPU to 43.5 ms on a Graphics Processing Unit (GPU). In Remote Sensing performance gains can attain 10 to 400 times. The ACO shows speedups of up to 23.60 with solution quality similar to the original sequential implementation.

References

[1] Gingold, Robert A., and Joseph J. Monaghan. "Smoothed Particle Hydrodynamics-Theory and Application to Non-Spherical Stars." *Monthly notices of the royal astronomical society* 181 (1977):375-389.

[2] Nuli, Uday A. and Kulkarni, P. J., "SPH Based Fluid Animation Using Cuda Enabled GPU," *International Journal of Computer Graphics & Animation*,vol.2, No.4pp.47–53,October 2012.

[3] Amador, Gonçalo, and Abel Gomes. "A CUDA-Based Implementation of Stable Fluids in 3D with Internal and Moving Boundaries." In *Computational Science and Its Applications (ICCSA), 2010 International Conference on*, pp. 118-128. IEEE, 2010.

[4] Boyer, Michael, David Tarjan, Scott T. Acton, and Kevin Skadron. "Accelerating Leukocyte Tracking Using CUDA: A Case Study in Leveraging Manycore Coprocessors." In *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pp. 1-12. IEEE, 2009.

[5] Plaza, A., Q. Du, Y-L. Chang, and R. L. King. "Foreword to the special issue on High Performance Computing in Earth Observation and Remote Sensing." *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of* 4, no. 3 (2011): 503-507.

[6] Mielikainen, Jarno, Bormin Huang, H. A. Huang, and Mitchell D. Goldberg. "Improved GPU/CUDA Based Parallel Weather and Research Forecast (WRF) Single Moment 5-Class (WSM5)

Cloud Microphysics." *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of* 5, no. 4 (2012): 1256-1265.

[7] Hong-Tao, Bai, He Li-li, Ouyang Dan-tong, Li Zhan-shan, and Li He. "K-Means on Commodity GPUs with CUDA." In *Computer Science and Information Engineering, 2009 WRI World Congress on*, vol. 3, pp. 651-655. IEEE, 2009.

[8] Gelman, Andrew. "Multilevel (hierarchical) Modeling: what it can and cannot do." *Technometrics* 48, no. 3 (2006): 432-435.

[9] Da Silva, AR Ferreira. "cudaBayesreg: Bayesian Computation in CUDA." *The R Journal* 2, no. 2 (2010): 48-55.

[10] Rossi, Peter E., and Greg M. Allenby. "Bayesian Statistics and Marketing."*Marketing Science* 22, no. 3 (2003): 304-328.

[11] Rossi, Peter, and Rob McCulloch. "Bayesm: Bayesian Inference for Marketing/Micro-Econometrics." *R package version* (2010): 2-2.

[12] Friston, Karl J., W. Penny, Christophe Phillips, S. Kiebel, G. Hinton, and John Ashburner. "Classical and Bayesian Inference in Neuroimaging: theory." *NeuroImage* 16, no. 2 (2002): 465-483.

[13] Berger, James O., and Thomas Sellke. "Testing a Point Null Hypothesis: the Irreconcilability of P Values and Evidence." *Journal of the American Statistical Association* 82, no. 397 (1987): 112-122.

[14] Vul, Edward, Christine Harris, Piotr Winkielman, and Harold Pashler. "Puzzlingly High Correlations in fMRI Studies of Emotion, Personality, and Social Cognition." *Perspectives on Psychological Science* 4, no. 3 (2009): 274-290.

[15] Liu, Weiguo, Bertil Schmidt, Gerrit Voss, and Wolfgang Müller-Wittig. "Accelerating Molecular Dynamics Simulations using Graphics Processing Units with CUDA." *Computer Physics Communications* 179, no. 9 (2008): 634-641.

[16] Thompson, Elizabeth A., and Timothy Anderson. "Use of CUDA for the Continuous Space Language Model." In *High Performance Extreme Computing (HPEC), 2012 IEEE Conference on*, pp. 1-5. IEEE, 2012.

[17] Schulz, Christian. "Efficient local search on the GPU—Investigations on the Vehicle Routing Problem." *Journal of Parallel and Distributed Computing* (2012).

[18] Bullnheimer, Bernd, Gabriele Kotsis, and Christine Strauß. "Parallelization Strategies for the Ant System." (1997).

[19] Delévacq, Audrey, Pierre Delisle, Marc Gravel, and Michaël Krajecki. "Parallel Ant Colony Optimization on Graphics Processing Units." *Journal of Parallel and Distributed Computing* (2012).

[20] Mussi, Luca, Stefano Cagnoni, and Fabio Daolio. "GPU-based Road Sign Detection using Particle Swarm Optimization." In *ntelligent Systems Design and Applications, 2009. ISDA'09. Ninth International Conference on*, pp. 152-157. IEEE, 2009.

[21] Arora, Ramnik, Rupesh Tulshyan, and Kalyanmoy Deb. "Parallelization of Binary and Real-Coded Genetic Algorithms on GPU using CUDA." In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pp. 1-8. IEEE, 2010.