IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 6, June 2016

Performance Optimization of Heterogeneous Computing Environment

Touseef Golandaz

M.Tech, CSE, HKBKCE, Bangalore, India

Abstract: The applications which are running on heterogeneous computing system (HCS) with hybrid processors (CPU and GPU) often uses only one processor and the Other processor will be in ideal state for the rest of operations of the application and this results in the wastage of the available computational resources and issues related to performances. It is possible to avoid this kind of wastage of the computational resources of modern HCS by dividing work across hybrid processors of HCS. We introduce a technique to fully utilize the hybrid processors of HCS to provide the significant improvement in performance and usage of hybrid processors for matrix multiplication based applications. We are using library functions like cblas (MKL) and cublas (NVidia's CUBLAS Library function) to divide the work across the hybrid processors of HCS.

Keywords: scale, GPGPU, hybrid processors, HCS

I. INTRODUCTION

Today HCS are mostly used in solving the scientific computation like engineering, to perform the data intensive computations. Due to advancement in the technology, now HCS contains powerful hybrid processors and can process data intensive computations in parallel and correctly.The HCS contain hybrid processors i.e. CPU and GPU. When we execute the application on HCS environment, application will use either CPU to perform all of its operation or use GPU to perform all of its operations but these two processors never be used at the same time by the application to perform its operations [1] [2]. This means one processor will be in ideal state for the rest of the operations of that application and other processor will be utilized to perform the operations of that application. This results in the underutilization of the available computing resources and will significantly reduce the performance of the HCS [3]. Even the HCS environment has hybrid processors, still the operations of the applications are not distributed between the processors and not computed in simultaneously.

In today's era the matrix multiplication based applications uses HCS to perform its operations due to the presence of hybrid processors in HCS [1] [2] [3] and not only that many scientists are working to find the ideal techniques to utilize all the computational capacity of the GPU for efficient matrix multiplication andfurthermore to significantly decrease the time of execution for scientific applications which are based on the matrix multiplication.

We will distribute the operations of the matrix multiplication between the both the processors of the HCS [5] [6][7] using library functions developed by the Intel [9] and NVidia [11]. And identity the optimal load distribution ratio between these two processors of HCS.

The main objective of this research work is to identify the optimal load distribution point, fully use the computational capacity of HCS and optimize the performance of the HCS.

II. EXISTING SYSTEM

The existing optimization techniques for the HCS only uses one processor to perform all the operations of the matrix multiplication based applications. But these techniques have ability to fully utilize the computational capacity of the each processor to their maximum limits. The CPU was built to perform the general purpose operations and GPU was built to perform the graphics related operations such as gaming. With time the technology evolved and today the GPUs are also used to perform the general purpose computations. This leadsto the term called GPGPU. Today HCS with hybrid processors are used in the field of high performance computing (HPC) to solve the scientific problems. So we need new techniques, which allow us to perform the operations of the scientific computations bydistributing the operations between the CPU and GPU in order to fully utilize the computational ability of the HCS and provide the significant performance improvements.

III.PROPOSED SYSTEM

We are proposing a method which allow us to perform the matrix multiplication operations by distributing the operations of the matrix multiplication between the hybrid processors (CPU and GPU) [8] of the HCS and execute these distributed operations simultaneously on the hybrid processors(CPU and GPU). This will solve the problem of underutilization of the computational resources of the HCS. Our method uses the library functions developed by the Intel and NVidia. The cblas ( ) function of Intel MKL is being used to execute the operations of the matrix multiplication associated to CPU of the HCS and cublas ( ) function of the NVidia's CUBLAS is being used to execute the operations associated to GPU of the HCS. We are using Open MP [10] library functions to provide parallel execution environment for the HCS. With this we can easily accomplish our objectives of this research work.

A. System Architecture

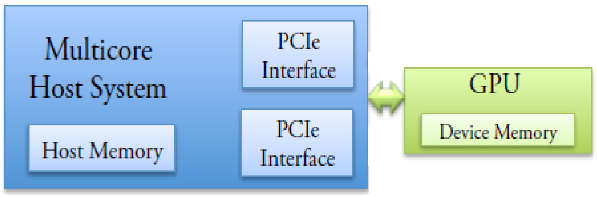

Fig. 1An Example of HCS Environment

The Fig. 1 shows the architecture of HCS with multicore host (CPU) and single GPU (device). The multicore host of the HCS is connected to single device via PCI Express connections. Features of the HCS as follows

(1) The device and the host have different memories.
(2) Device is a co-processor to the host in the HCS and the device is controlled by a thread running on the host.
(3) The host and device have different architecture.
(4) Device is specially optimized for throughput and host isspecially optimized for latency.
(5) The host contains few powerful cores and the device contains many cores but less powerful as compared to host cores in terms of clock speed.
In this research work, we are considering all these factors and try to accomplish our objectives, such as obtain high performance, a high degree of parallelism, sharing of work between the host and deviceand find optimal sharing ratio for host and device.

B. Matrix Partition

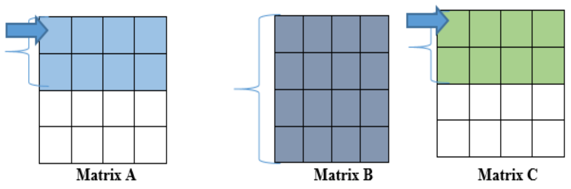

Fig 2 First Half of the Matrix Multiplication

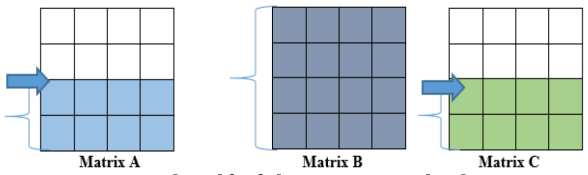

Fig 3 Second Half of the Matrix Multiplication

We have two input matrix i.e. matrix A and matrix B and one output matrix i.e. matrix C. first we will partition the matrix A into two blocks of rows and we do not partition matrix B. Then each block of matrix A will be assigned for multiplication with matrix B on either host or device.

In this method, we will compute one block of matrix A with matrix B on hostby the help of MKL [9] library function and another block of matrix A with matrix B on deviceby the help NVidia's CUBLAS [11] library functionsimultaneously. The results will be stored in the matrix C as shown in the Fig. 2 and Fig. 3.

C. System Specifications

TABLE 1
Executional Environment Specifications of HCS

No.	System	Host(CPU)	Device(GPU)
1	Processor Name	Intel Xeon E5520	NVidia Quadro FX 3800
2	Sockets	2	1
3	Cores	4	192
4	Threads Per Sockets	8	600MHz
5	Core Speed	2.26GHz	GDDR3
6	Ram Type	DDR3	1GB
7	Ram Size	12GB	51.2GB/s
8	Memory Bandwidth	25.6GB/s	NVidia Quadro FX 3800
9	L1 Bata Cache	4*32KB	1
10	L1 Instruction Cache	4*32KB	192
11	L2 Cache	4*128KB	600MHz
12	L3 Cache	8MB	GDDR3
13	Compiler	GCC/NVCC	1GB

Our HCS environment contains above mentioned specifications. We will execute our method on this computational environment and observe the results.

D. Results

Table 2
Execution time on HCS

Size of the Matrix	(Host + Device) at the ratio of 0.5	Least Execution time in Sec Only for Host	Least Execution time in Sec Only for Device
1000	0.0500	0.016	0.0185
2000	0.5010	0.096	0.0925
3000	1.1500	0.351	0.02685
4000	2.1430	0.663	0.6025
5000	4.2450	1.707	1.1545
6000	7.0680	2.352	2.0220
7000	11.0340	4.163	3.1440
8000	16.7980	4.636	4.5560
8200	5.0052	18.0720	7.506

After executing our method on above mentioned HCS environment, we can observe the results in Table 2 and Table 3, we can say that our method provides significant performance improvements by utilizing the computational capacity of host and device over the methods which only uses either host or device to perform all the operations of the matrix multiplication. The Table 3 provide information about the optimal execution time for matrix multiplication using host with device for the different matrix size with distribution ratio (scale) between the device and host. For example if the scale value is 0.8 means the 80% of the operation of matrix multiplication will be performed on device and 20% of the operations will be performed on the host.

Table 3
Optimal Execution Time in HCS

Size of the Matrix	Threads	Scale	Optimal Execution Time in Sec (Host + Device)
1000	4	0.7	0.0125
2000	16	0.7	0.0645
3000	16	0.55	0.2435
4000	12	0.65	0.4495
5000	12	0.6	1.009
6000	8	0.75	1.7245
7000	16	0.6	2.601
8000	16	0.7	3.2505
8200	12	0.6	4.4716

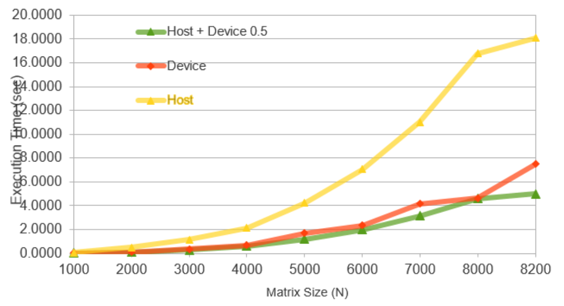

Fig. 4 Graphical Representation of Table 2

The Fig. 4 show the graphical representation for execution time given in the Table 2 and the Fig 5 shows the graphical representation of the comparison between the execution time given in the Table 2 and Table 3.

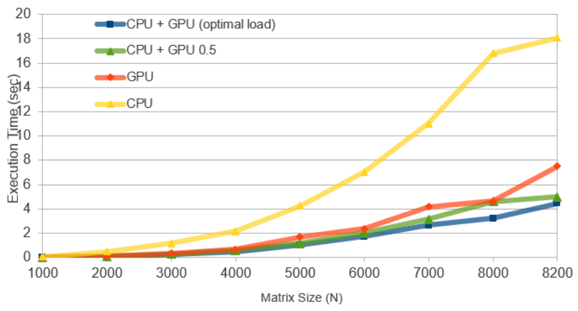

Fig. 5 Graphical Representation of the Table 2 and Table 3 to Compare the Execution Time.

IV. CONCLUSION

The HCS architecture isevolved with time, due to the advancement in the technologyi.e. from multi-processor system architecture to multicore processor architecture and to hybrid processors. One must have very good knowledge of the HCS architecture in order to effectively utilize all the processing capacity of the HCS hybrid processors to achieve maximum performance optimization for the applications.In this research work, we present workload distribution on the HCS architectures for the matrix multiplication operations and computations of matrix multiplications are performed using library functions provided by Intel and NVidia.Our obtained results shows that with the help of our methodology it is possible to increase of performance of HCS,This research work is carried out to demonstrate the comparison of the execution time on when the operationof the matrix multiplication is performed using only host (CPU), using only device (GPU) and using host (CPU) with device (GPU).

ACKNOWLEDGMENT

I would like to express my regards and acknowledgement to all who helped me in completing this project successfully. I consider it as great privilege to convey my sincere regards to Professor and Head **Dr. Loganathan.R.**, Department of CSE, HKBKCE, Bangalore for his constant encouragement throughout the course of the project.I would like to thank my professor and guide **Dr. Waseem Ahmed**, for his support and providing directions during my project work. At long last, I say thanks to Almighty, all the staff individuals from CSE Department, my relatives and companions for their steady backing and support in doing the undertaking work.

REFERENCES

[1] Jack Dongarra, Fengguang Song and StanimireTomov: Enabling and Scaling Matrix Computations on Heterogeneous Multi-Core and Multi-GPU Systems. In Preceding of 26th International Conference on Supercomputing, pp. 365-367.ACM, 2012.
[2] Song, F., S. Tomov, and J. Dongarra. Efficient Support for Matrix Computations on Heterogeneous Multi-core and Multi-GPU Architectures. University of Tennessee. Computer Science Technical, Report UT-CS-11-668 (2011).
[3] Fengguang Song, and Jack Dongarra. A scalable approach to solving dense linear algebra problems on hybrid CPU-GPU systems. Concurrency and Computation: Practice and Experience 27.14 (2015): 3702-3723.
[4] Fengguang Song, and Jack Dongarra. A scalable framework for heterogeneous GPU-based clusters. Proceedings of the twenty-fourth annual ACM symposium on Parallelism in algorithms and architectures. ACM, 2012.
[5] Muhammed Osaman, Hassan Youness and Aiman Tarek: Load Balancing on CPU-GPU Heterogeneous System: at researchgate.
[6] Bibhudatta Sahoo: Dynamic Load Balancing Strategies in Heterogeneous Distributed System. 2013.
[7] Galindo, Ismael, Francisco Almeida, and José Manuel Badía-Contelles. "Dynamic load balancing on dedicated heterogeneous systems." Recent Advances in Parallel Virtual Machine and Message Passing Interface. Springer Berlin Heidelberg, 2008. 64-74.
[8] Lee, Janghaeng, MehrzadSamadi, Yongjun Park, and Scott Mahlke. "Transparent CPU-GPU collaboration for data-parallel kernels on heterogeneous systems." In Proceedings of the 22nd international conference on Parallel architectures and compilation techniques, pp. 245-256. IEEE Press, 2013.
[9] INTELMKL'scblasdgemmavailableat https://software.intel.com/en-us /node/429920.
[10] OpenMPRelatedRecoursesareavailableathttps://openmp.org/wp/resources.
[11] CUDADocumentationAvailableathttp://docs.nvidia.com/cuda/#axzz47WyvqDZK.