GPU PARALLEL COMPUTING

Sakshi Dhingra ,Shikha Shah
15BEC102,15BEC109
Electronics and Communication, Institute of Technology Nirma University, Ahmedabad, Gujarat, India Pin-382481
15bec102@nirmauni.ac.in
15bec109@nirmauni.ac.in

Abstract: Parallel systems are widely used by multicore processors, heterogeneous cell broadband engines as well as highly parallel GPUs. parallel systems require that parallel programming is necessary if we want to make maximum use of multicores. As a result , multicore CPUs are no longer preferable because they demand a lot of programming effort in parallelizing code for their multicore architectures. To surmount such drawbacks , GPU assisted computing was thought of as an alternative for dataparallel applications. GPU s are popular and inexpensive accelerators . A floating point intensive algorithm is chosen for radar imaging applications and analysed. This way , we can analyse the computing which uses GPU.

I. Introduction

GPUs (Graphics Processing Unit) are packed with large number of floating point units that support both task and data parallelism. CPU focuses on reducing the time for a single execution stream whereas GPU aims at reducing throughput execution time for a group of instructions by increasing the speed. GPU can operate multiple threads at a time and for that , it operates on a Hardware Multithreading technique .

This technique schedules large number of active threads. The threads which are ready , occupy GPU's computing resources and thus avoid memory latency. GPU's executions are maximum when multiple data values are applied to a single instruction if the architecture is based on SIMD concept.

II. Application in Radar System

GPU parallel computing was also used in a radar system developed to detect hidden targets and obstacles. The surface of the radar has 2 antennas and an array of 16 receiver antennas. the aim was to speed up the system so that it can work on real time. According to the research results, major portion of the execution time was consumed in image formation step in radar algorithm. There is an image grid which contains pixels. Signal transmitted from transmitter goes to the pixel, gets reflected and reaches the receiver. The process is repeated for different positions of radar so that exact solution can be obtained.

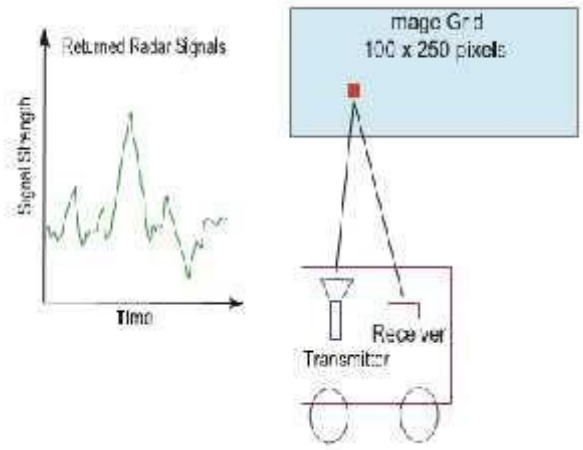

Figure 1. Image reconstruction algorithm [8]

The time taken by signal to travel from transmitter to pixel to receiver is called trip time. Output image is obtained by calculating this trip time .

When we calculate propagation time for different radar positions, a number of independent floating point calculations are involved. This leads to GPU computing.

III.GPU-CPU Communication

GPU are not used to run stand-alone programs. CPU applications manages GPU and uses it for specific communications. Main program is executed in CPU. It invokes kernel on GPU for specific task and then obtains the result once kernel is terminated. A GPU kernel is a function that executes through thousands of parallel GPU threads. GPU acts as a parallel co-processor.

GPU is connected with CPU through PCIe bus. Discrete GPU has its own physical memory. This cannot be referenced by CPU directly.Thus the input to kernel is staged into GPU memory before invoking kernel and outpur is transferred back. But the PCIe bandwidth to the main host is less than the GPU bandwidth for the local memory. As a result CPU-GPU data transfer slows down. Thus minimization of data transfer becomes necessary for optimization of the time required to process.

IV. Introduction To CUDA (Compute Unified Device Architecture)

Nvidia launched a development kit known as CUDA. This gave programmers,the facility of getting access to parallel graphics card.This strategy advertises GPU parallel computing which facilitates the use of GPU's arithmetic capabilities for solving computationally challenged problems.

The programming model is an extension to C language. CUDA provides an interface and transfers data to 2 places. One, the GPU accelerator also known as device and the other one is kernel. Execution takes place on GPU. We translate only the main program to CUDA and the remaining program is kept in C only.The portion in C is executed in serial manner and as soon as the process reaches to a kernel , parallel GPU computing begins. An important execution model difference between CPU and GPU is hardware multithreading.

The processor in CPU targets at reducing the execution time of a single instruction by computing it as fast as possible whereas in GPU, the aim is to decrease the execution time of a whole group of instructions. In other words , the active threads ready for execution are greater in number, than the resources available for computation. These large threads are kept for each core so that when switching takes place from a stalled (static) thread to an executable thread , no penalty takes place.

GPU uses hardware multithreading to hide main memory latencies. Thread stalls effect the performance and execution. When a thread is stalled , processor cannot dispatch the next instruction in an instruction stream because a standing instruction is already present there. These thread stalls last 100's of clock cycles and cause latencies. Therefore , instead of ALU being idle during this period , GPU maintains more execution contexts in a chip which can be executed simultaneously. When some instructions are stalled , instructions from other runnable threads are performed. This technique is called hardware multithreading.

Per microprocessor, maximum number of active threads is 768. CUDA threads are independent instruction streams which can be parallel executed. A unique identification number is there for each thread.

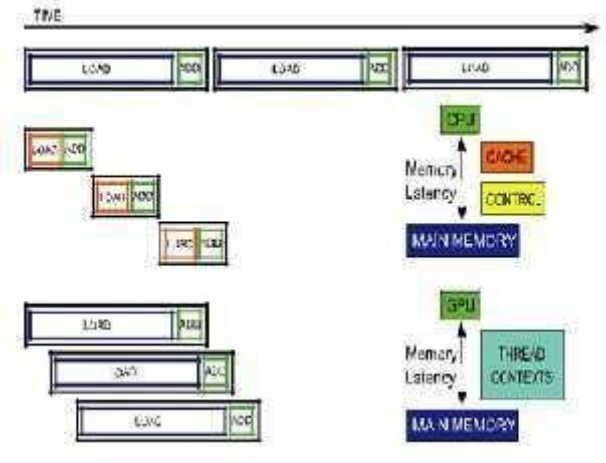

Figure 2. Addressing memory latency [8]

V. Optimization Of CUDA

The basic concept behind optimization is reducing global memory access time. One of the methods is using like CPU cache. Nvidia generally has 16KB of cache and 16KB shared memory. Shared memory is the memory which is accessible to various programs being executed, so that they can pass data among themselves. Shared memory is a memory hierarchy which is handled by user.

Another method is performing reading and writing through coalesced memory. When the memory is allocated, there are situations where two adjacent memory blocks are freed. Combining these would make them a single freed memory block which is called "coalescing".

VI. Results Of Comparison of processors:

Here, we compare 2 processors GeForce 8800 GTX (GPU) and Intel xeon 5160(CPU) on the following two grounds:1)Frequency 2)Available floating point units.

Processor	Clock Frequency	Independent Cores	Floating-point Units

GeForce 8800 GTX	1.35 GHz	16 (multiprocessors)	128
Intel Xeon 5160	3.0 GHz	2	16

Table 1. List of processor's floating-point capability [8]

The performance of various processors is shown in figure 3. It shows the execution time taken by the imaging algorithm of the radar equipment of the processing of collected data.

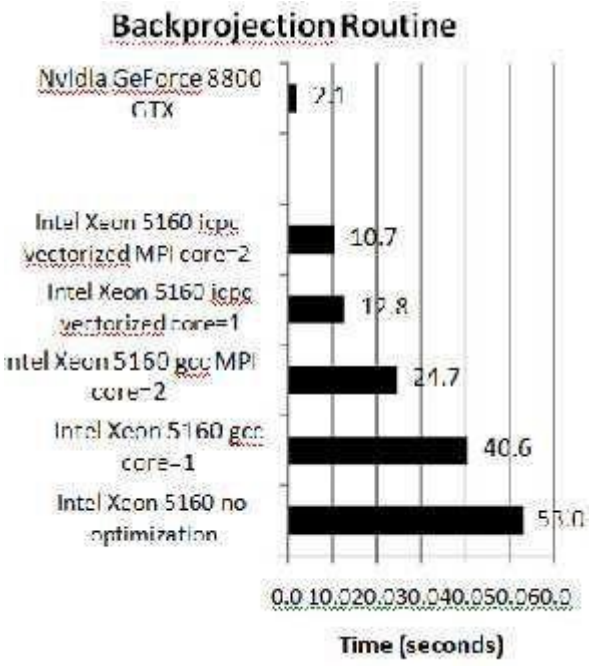

Figure 3. Performance results [8]

VII. If GPUs are faster, can CPUs be replaced by GPUs ?

GPU has certain advantages over CPU but it cannot replace CPU due to the following reasons:

1) There are more number of processor cores in GPUs but individual GPU cores run slower than a CPU core.

2) Also, everyday computing operations cannot be performed because all the features have not yet been inculcated in GPUs. For example, features required for modem operating systems are absent.

Initially, GPUs were used only for graphical purposes but then later, people made use of the large number of cores and used it for computing parallel data streams. GPU also lacks interrupt and virtual memory features.

3) CPU and GPU have architectures differing from each other and this makes each of them suitable for different tasks. GPU computes large data through number of streams and can perform only simple operations on them whereas, CPU can perform complex operations on a single stream but cannot manage data in multiple streams.

4) If we want an application to work on GPU, it needs to be specifically programmed for that. There are various techniques for this like introducing new programming languages, making changes in the existing ones.

5) AMD is coming up with Accelerated Processing Unit (APU) which will combine GPU with conventional x86 CPU cores. It will make the system compact and less costly. There would be a good multimedia performance and also, a separate GPU wouldnot be needed.

VIII. Future Work

Discrete and Hybrid GPUs are evolving faster day by day. GPU is being evolved from working as co processor to system which runs complete self-contained program. Without the involvement of CPU, GPU will be directly be able to red files in the host file system. Nvidia researchers have an insight to see multi-chip GPU designs as the future of high-performance graphics cards.

Transistor count on modern-day GPUs is increasing constantly. To fit large number of transistors on a single die has become a very expensive and complex process. With the increase of complexity, GPU production is becoming a costly process. Thus research on MultyChipModule GPU is in progress. Implementation of multichip module setup in an effective way won't be easy but, so long as they can maintain effective scaling with successive GPUs, it will be worth it.

IX. Conclusion

We have used the example of the imaging algorithm of the radar to understand GPU parallel computing in this paper.

We compared the characteristics of CPU and GPU and went through hardware multi threading. We compared the processing times for 2 processors, one GPU and the other multicore CPU. Execution time was less in case of GPU and hence, its processing results are better. Therefore,

GPU processors are very popular these days. As per observation, CUDA's performance exceeded GPU's by a factor of 5.

A high-end discrete GPU consumes about 200W power, almost double than that of high-end CPU. So, when we say about power efficient execution in GPU, we actually mean that if only a GPU is used, acceleration speedup must at least twice (2x) .

Instead of running the individual processors, if we run both CPU and GPU, a better power efficiency can be achieved [1].

References

1. Fatahalian, K. and M. Houston, "GPUs: A Closer Look." *ACM Queue, GPU Computing*, Vol. 6, Issue 2, April 2008.

2. Owens, J., M. Houston, D. Luebke, and S. Green, "GPU Computing." *Proceedings of the IEEE*, Vol. 96, No. 5, 2008.

3. Ressler, M., L. Nguyen, D. Wong, and G. Smith, "The Army Research Laboratory (ARL) Synchronous Impulse Reconstruction (SIRE) Forward-Looking Radar." *Proceedings of SPIE, Unmanned Systems Technology IX*, Vol. 6561, May 2007.

4. McCorkle, J. and M. Rofheart, "An order N2log(N) backprojector algorithm for focusing wide-angle widebandwidth arbitrary-motion synthetic aperture radar." *SPIE*, Vol. 2747, 1996.

5. Nguyen, L. and S. Jeffrey, "SAR Image Formation Using Phase-History Data from Non-Uniform Aperture." *Proc. of SPIE, Radar Sensor Technology XI*, Vol. 6547, 2007.

6. "*NVIDIA CUDA Compute Unified Device Architecture Programming Guide*," Version 2.0, 2008.

[7] M. Silberstein and N. Maruyama. An Exact Algorithm forEnergy-Efficient Acceleration of Task Trees on CPU/GPU Architectures. In *Proc. of the International Conference on Systems and Storage*(SYSTOR'04). ACM, 2011.Usually a highend discrete GPU consumes about 200W of power, which is almost two times the consumption of that of a high-end CPU. Therefore, achieving power efficient execution in GPUaccelerated systems implies that the acceleration speedups must exceed 2x if only a GPU is used. In practice, employing both a CPU and a GPU may achieve better power efficiency than running on each processor alone [1].

[8] S. J. Park, "An Analysis of GPU Parallel Computing," *2009 DoD High Performance Computing Modernization Program Users Group Conference*, San Diego, CA, 2009, pp. 365-369. Figure 1. Image reconstruction algorithm [8]Figure 2. Addressing memory latency [8]Figure 3. Performance results [8]Table 1. List of processor's floating-point capability [8]