How parallel computing improves the efficiency

**abstract**

Suppose a computer is going to do several work with computations. Traditionally, it will start the next computation only when the last one is finished. This procedure is time-consuming and inefficient. The efficiency can be improved by parallel computing. Parallel computing is a form of computation in which many calculations are carried out simultaneously. As parallel computing can save time and power, it has become the dominant paradigm in computer architecture, mainly in the form of multi-core processors. This article is going to discuss what parallel computing is, how it works and how it attributes to the improvement of efficiency in detail. The parallel technology that will be discussed in detail is the CUDA parallel technology. Finally, we will discuss the practical usage of CUDA parallel technology.

**Introduction**

Parallel computing is a form of computation in which many calculations are carried out in same time. The parallel computing can improve the efficiency of the computing. There are several different model for the parallel programing, such as task channel model and the message passing model. These two models will be discussed in detail in the following article.

In hardware terms, there are many types of parallel computing such as multicore computing, symmetric multiprocessing, and general-purpose computing on graphics processing units. In particular, general-purpose computing on graphics processing units uses GPU (graphics processing unit) to do the computation work which was traditionally done by the CPU (the central processing unit). So a GPGPU (General-purpose computing on graphics processing units) pipeline is a kind of parallel processing. More importantly, it operates on the graphical data much faster than the traditional CPU. When it comes to the GPGPU, the CUDA technology must be also mentioned. The CUDA is a parallel computing platform and program model developed by the NVIDIA company. This paper will go over parallel computing and the CUDA technology and discuss a few examples of practical use of the CUDA technology.

**Overview of Parallel Computing**

Recently, clustering and distributed computing is becoming popular because of their high computing performance. The workstations, or PCs, with high computing performance and low price, are connected together through a fast communication interface to accomplish the parallel computing.

In addition, the development of the modern network provides a much faster communication speed. This results in the parallel computing on clustered systems becoming an attractive topic. In order to use more than one processor in a program, the processors must share data with each other. One of the practical solutions is to transfer data through messages between the computers. The MPI (message passing interface) is one of the best solutions in parallel programming because it has several advantages; it is relatively simple to use the method by writing library functions or application program interface using C, C++ or Fortran language.

Parallel programming design begins with the choice of many different models which can be used to do parallel computing. They differ in the perspective of flexibility, task interaction method, support for locality, and scalability. Two examples of these models are the task channel model and the message passing model.

The task/channel model encapsulates a program and local memory and to define its interface to its environment. A channel is a message queue. A sender can place messages into the channel and a receiver can remove messages from it.

The message passing may be the most commonly used model in parallel programming. Using this model, the program can do multiple tasks, each encapsulating local data. The task interacts with other tasks by sending and receiving messages.

When it comes to the design of the parallel algorithms, four attributes need to be guaranteed: concurrency, scalability, locality, and modularity. Concurrency means many actions can be performed simultaneously. This attribute is essential because the program need to be executed in the different processors. Scalability indicates resilience to increasing processor counts. This attribute is also very important because, in most environments, the amount of processors is likely to increase. Locality means the local memory can access remote memory. This attribute is extremely important in the four attribute to the multiprocessor architecture. Modularity is also essential in parallel computing.

The parallel algorithms design has four stages, which are partitioning, communication, agglomeration, and mapping. The first stage is partition , which means divide the task into many small tasks. The next step is to communicate in order to get data required for the execution of the task. Then, the agglomeration stage means decreasing

communication and development costs while guarantee the flexibility if possible. The final step is mapping, which means map the task to different processors, and its goal is to minimize the execution time.

**Overview of the CUDA Technology**

Traditionally, data was transferred from the CPU to GPU. However, with the development of the GPU, simple data and some 2D or 3D format data could be stored as well. Then transfer the complex data to the CPU to compute. The reason why GPU operates data in this way is that it can analyze the data of the format much quicker than the CPU because the GPU can do every draw operation. On the other hand, the GPU needs to transfer some complex data back to the CPU. One of the advantages of the design of GPGPU is that it allows GPU to transfer data back to the CPU.

The development of the graphics processing units makes it has computational ability against the CPU. Researchers found out that the GPU can also be used to do computations. There are several advantages of using GPU to do computations from the perspective of parallel computing. The first advantage is the number of the cores. For example, NVIDIA GeForce GTX480 has 480 cores at most, and each of these cores can handle 768 threads. The number of the threads is another advantage of the GPU computing. However, the GPU computing also has some disadvantages. For example, the memory is transferred from the GPU to the CPU very slowly. The process of copying from the CPU to the GPU is about 10–15 times slower than one within the GPU

With the similar idea of the GPGPU, the NVIDIA company developed a parallel computer architecture CUDA. The CUDA is an architecture used to perform parallel computing, and it is easy to use. The CUDA technology can only be used in the NVIDIA GPU. Using the CUDA technology, the NVIDIA GPU can be used for general purposes. Unlike the CPU, GPU has the parallel architecture to execute many concurrent threads slowly, rather than execute a single thread.

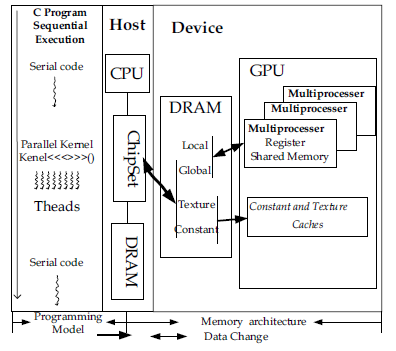


Fig 1. CDUA technology: parallel computing

Figure one shows the execution of a typical CUDA program and the CUDA memory hierarchical architecture. The whole process starts from the host (CPU) execution. When a kernel function is called, the execution is moved to a GPU. The GPU runs the code in the data parallel mode. When all threads have been executed in parallel, the corresponding grid terminates and the execution continues on CPU.

CUDA programming language is similar to the C programming language, which makes it easy for the program developer to learn and develop CUDA language. It is another advantage of the CUDA technology.

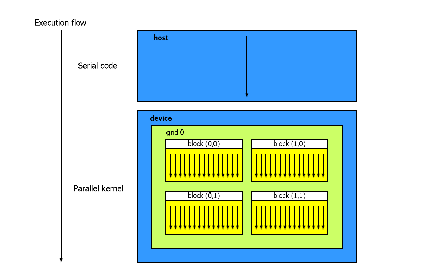


Fig 2.Programming models

Figure 2 is the programming model of CUDA. When a function is in low parallelism, the program will use the host (CPU) to execute the function. In this situation, the host (CPU) performs the task better than the GPU.

When a function is in high parallelism, the program will use parallel kernel to do the function. That is to say, the GPU can use a large number of parallel processing cores and threads to perform functions with high parallelism. And this operation makes the best use of the GPU powerful computing ability.

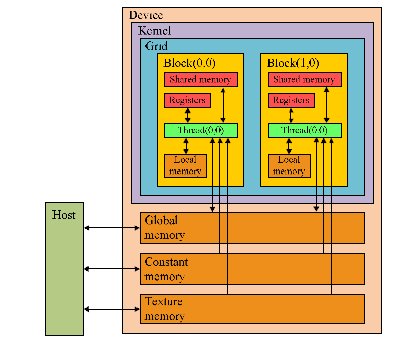


Fig 3. CUDA device architecture diagram

Figure3 is CUDA device architecture diagram. If a function is in high parallelism, the Host (CPU) will call the kernel in the CUDA to do a parallel operation. In the kernel, the smallest units to do the parallel operation is the unit called work-item. Several work-item works together is called work group. The work-items and work groups of the kernel can set the dimensions of space as one-dimensional, two-dimensional or three dimensional space. CUDA device architecture has several registers. They are private memory, local memory, constant memory and global memory.

Private memory is the memory stored in the CPU and only the work-item in the kernel can access to the private memory. The local memory is also stored in the CPU and only the work-item in the kernel can access to the local memory. And local memory has the same access speed as the private memory. Constant memory is stored in the GPU. In the GPU, any work-item can only read data, but they can not write data. However, CPU can write and read data in the constant memory. And the access speed of the constant memory is the fastest in all of the memories. Global memory has wider memory bandwidth, however, it also has higher latency, so the accessing speed of the constant memory is the slowest.

each hardware using CUDA technology has a set of multiprocessor cores. And it can execute a large number thread concurrently. For example, the NVIDIA GTX 280 has 30 multiprocessors and each multiprocessor has 8 processors.

When it comes to the framework of the CUDA, the framework of the CUDA is consist of the Runtime API and Driver API. Throughout the process of the runtime API, it is not necessary for the developers to control the hardware. Instead, the developers can call the kernel to complete the parallel computing. However, when using Driver API, not only the developers has to control the hardware, but they also has to call the kernel to complete the parallel computing.

Similarly, the programming of the CUDA can be divided into two types. One is CUDA Runtime API programing, the other one is CUDA Driver API programming.

In the CUDA Runtime API programming, the process involves several steps: set the quantity of the block and the thread, input test sequences, initialize the kernel’s memory space. After all the threads has been executed, then execute another one. Transfer the data of the CPU to the GPU (kernel), then use GPU parallel computing to execute the program in the kernel. After the execution of the program in the kernel, transfer the data back from the GPU to the CPU.

In the CUDA Driver API programming, the process involves several steps similar to the CUDA Runtime API programming: set the quantity of the block and the thread, input test sequences, initialize the kernel’s memory space. After all the threads has been executed, then execute another one. Transfer the data of the CPU to the GPU (kernel), then use GPU parallel computing to execute the program in the kernel. Set the input sequence required by the kernel. After the execution of the program in the kernel, transfer the data back from the GPU to the CPU.

**Discuss the application of the CUDA technology**

From the discussion above, the advantages of the GPU computing is obvious. Recently, Many researchers and developers are doing research and application based on the GPU computing. They take advantages of the GPU computing to do researches or practical application on many fields like image and video processing, computational biology and chemistry.

For example, the CUDA technology can be used to do Motion JEPG 2000 format’s compression. CUDA parallelism can also be used to implement Adaptive and Dynamic Data Structures.

Then, We will discuss the actual application or researches based on the CUDA technology.

1. Motion JEPG 2000

With the development of the CUDA technology, the users now can use parallel GPU resources. There were researchers tried to develop a video compression program based on the NIVIDIA’s architecture. Motion JEPG 2000 is one of the ISO standard video formats. It has several advantages. Motion JEPG 2000’s compression performance is superior and it support small scale of editing to the image content. In the research has been done, the GeForce GTX 280 was used to process Motion JPEG 2000.

Motion JEPG converts a PPM video frame into a JPEG 2000 video frame. And the encoder of the JPEG has 4 processors. They are DC level shift and Reversible Component Transform(RCT), Wavelet Transform, quantization and Embedded Block Coding with Optimized Truncation (EBCOT).

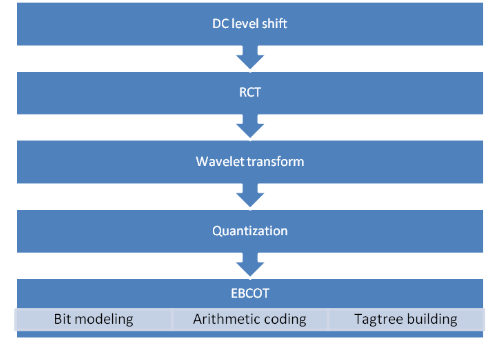


Fig4. JEPG 2000 Encoding System

In the process of research, per JPEG 2000 was developed for effective parallelization. Using the parallel computing technology of the CUDA, 25 video with the resolution of 2048\*1080 can be compressed in 4998ms, which means that it only takes one second to compute 5.1 frames, which means the speed is 20.7 times of the existing one.

1. Adaptive and Dynamic Data Structure

CUDA parallelism can also be used to implement Adaptive and Dynamic Data Structures. CUDA is a parallel programing architecture used in the NVIDIA GPUs. In CUDA, the GPU and the CPU used as co-processors, which means they both used to compute. And the kernel was called asynchronously. Synchronization is achieved by calling synchronization function or implicitly calling when the CPU tries to access memory on the device.

CUDA is scalable programming because it runs hundreds of cores and thousands of threads at the same time. And each threads can execute the same code in parallel mode in CUDA . Threads can only share the data in the same block but the threads can’t share the data in different blocks.

Then we will explain function of the dynamic data structures and its adaptive use on GPU and for that we implement minimum number finding which uses parallel method.

CUDA parallel architecture has some difference compared to the traditional compute architecture. In order to improve performance and efficiency, it is important to choose the appropriate architecture.

In the GPU supporting CUDA, Streaming Multiprocessor is used to execute thread. Each Streaming Multiprocessor may containing 8,16, or 96 stream processors. And 8 or 16 Multiprocessor can be used to execute thread concurrently. Accessing global memory of the GPU may cause longer latency, if using the global memory as the texture memory when the cache missing occurs.



Fig 5. CUDA memory model

In the figure 5, the shared memory is in the multiprocessor and shared by the threads. Both host (CPU) and device can access to the shared memory. If the device need to access to the data, the data must transform from the main memory to the global memory. Each thread has its local memory.



Fig 6.CPU-GPU Interface Model

Figure shows the interface model of the CPU and GPU. GPUs library can be linked to GPU code in order to access the host(CPU) file.

In the multithreaded applications, memory allocation functions are being widely used.

Dynamic parallel was first used in the GK110 chip in CUDA 5.5 version. In the former system, the kernel are called by the host code. And the algorithm contains recursion, loop. Dynamic parallelism makes the algorithm can use the kernel. And make it not to increase burden to the CPU.



Fig 7. The processing without or with dynamic parallelism

In the figure, we compare two different situations. One is the CUDA without dynamic parallelism (old version), the other is the CUDA with dynamic parallelism. In the CUDA without dynamic parallelism, it needs to report back to the CPU and request the CPU to launch a new kernel. The main disadvantage of the CUDA without dynamic parallelism is that it increase the burden of the CPU, while in the CUDA with dynamic parallelism, it doesn’t need to ask the CPU to launch the parallelism.



Fig 8. Completion of Parent and Child grids

The Fig 8 illustrate that the parent Grid is not considered to be completed until all the child Grids have been completed.



Fig 9. Minimum number finding

In order to improve the performance, the parallelization is used on the GPU.

Firstly, the advice allocates memory on host and device for input data and output data. Then, Initialize the input data. After that, transfer data from the host to the device. And then, launching the kernel to find the minimum value. After that, transfer the minimum value back from the device to the host. Finally, printing the minimum value.

Compute time according to the number of the elements. With the same number of elements, we compare the executing speed of serial computing and parallel computing.

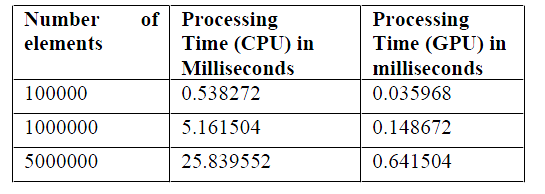


Table1. the executing speed of serial computing and parallel computing

As shown in the table 1, the executing speed of serial computing is much faster than the parallel computing.

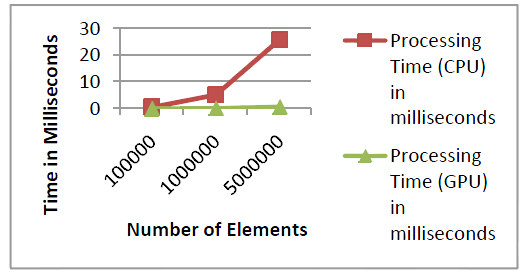


Figure 10. the processing time of CPU and the processing time of GPU

The figure 10 showed the executing time of the CPU and GPU when executing the serial computing or parallel computing. As show in the figure, we can find out that the executing time of the parallel computing is much less than the serial computing both in the CPU and in the GPU.

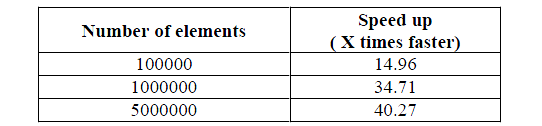


Table 2. the number of the elements and speed

As the table 2 shows, the parallel computing speed up very quickly when the number of elements increase.

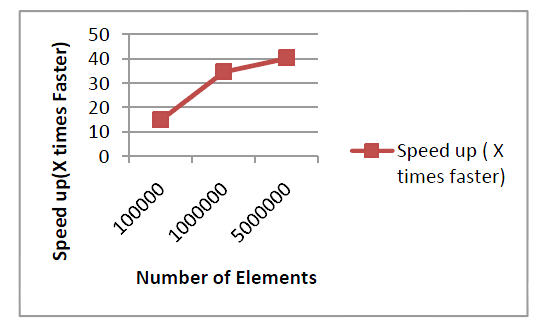


Figure 11

Similarly to the table 2, the figure 11 shows the speed increase with the increase of number of elements.

**Reference**

Ching-Lung Su; Po-Yu Chen; Chun-Chieh Lan; Long-Sheng Huang; Kuo-Hsuan Wu, "Overview and comparison of OpenCL and CUDA technology for GPGPU," *Circuits and Systems (APCCAS), 2012 IEEE Asia Pacific Conference on* , vol., no., pp.448,451, 2-5 Dec. 2012

Datla, S.; Gidijala, N.S., "Parallelizing Motion JPEG 2000 with CUDA," Computer and Electrical Engineering, 2009. ICCEE '09. Second International Conference on , vol.1, no., pp.630,634, 28-30 Dec. 2009

Garland, M., "Parallel computing with CUDA," Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on , vol., no., pp.1,1, 19-23 April 2010

doi: 10.1109/IPDPS.2010.5470378

Kumm, E.T.; Lea, R.M., "Parallel computing efficiency: climbing the learning curve," TENCON '94. IEEE Region 10's Ninth Annual International Conference. Theme: Frontiers of Computer Technology. Proceedings of 1994 , vol., no., pp.728,732 vol.2, 22-26 Aug 1994

Reyzlin, Valeriy I.; Tartakovsky, Eugene A., "Solving problems of adaptive optics using parallel algorithms based on the CUDA technology," Strategic Technology (IFOST), 2012 7th International Forum pp.1,3, 18-21 Sept. 2012

Ronghui Cheng; Yang, Eryan.; Ting Liu, "Speeding up motion estimation algorithms on CUDA technology," Microelectronics and Electronics (PrimeAsia), 2010 Asia Pacific Conference on Postgraduate Research in , vol., no., pp.93,96, 22-24 Sept. 2010

Sangale, A.L.; Devani, U.; Nikam, V.B.; Meshram, B.B., "Implementing adaptive and dynamic data structures using CUDA parallelism," Advances in Engineering and Technology Research (ICAETR), 2014 International Conference on , vol., no., pp.1,7, 1-2 Aug. 2014

Shams, R.; Barnes, N., "Speeding up Mutual Information Computation Using NVIDIA CUDA Hardware," Digital Image Computing Techniques and Applications, 9th Biennial Conference of the Australian Pattern Recognition Society on , vol., no., pp.555,560, 3-5 Dec. 2007

The principle and details of the CUDA technology

(http://en.wikipedia.org/wiki/CUDA)

The principle and details of the GUGPU technology

(http://en.wikipedia.org/wiki/General-purpose\_computing\_on\_graphics\_processing\_units)