Joint International Topical Meeting on Mathematics & Computation and Supercomputing in Nuclear Applications (M&C + SNA 2007)
Monterey, California, April 15-19, 2007, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2007)

Study on the acceleration of the neutronics calculation based on GPGPU

Yasunori Ohoka and Masahiro Tatsumi
Nuclear Fuel Industries, Ltd
Fuel Engineering and Development Department, Kumatori Works
1-950, Asashiro-Nishi, Kumatori-cho, Sennan-gun, Osaka, 590-0481 Japan
ya-ohoka@nfi.co.jp; tatsumi@nfi.co.jp

ABSTRACT

The cost of the reactor physics calculation tends to become higher with more detail treatment in the physics models and computational algorithms. For example, SCOPE2 [1] requires considerably high computational costs for multi-group transport calculation in 3-D pin-by-pin geometry. In this paper, applicability of GPGPU[2] to acceleration of neutronics calculation is discussed. At first, performance and accuracy of the basic matrix calculations with fundamental arithmetic operators and the exponential function are studied. The calculation was performed on a machine with Pentium 4 of 3.2MHz and GPU of nVIDIA GeForce7800GTX using a test program written in C++, OpenGL and GLSL on Linux. When matrix size becomes large, the calculation on GPU is 10-50 times faster than that on CPU for fundamental arithmetic operators. For the exponential function, calculation on GPU is 270-370 times faster than that on CPU. The precisions of all the cases are equivalent to that on CPU, which is less than the criterion of IEEE754 (10^{-6} as single precision).
Next, the GPGPU is applied to a functional module in SCOPE2. In the present study, as the first step of GPGPU application, calculations in small geometry are tested. Performance gain by GPGPU in this application was relatively modest, approximately 15%, compared to the feasibility study. This is because the part in which GPGPU was applied had appropriate structure for GPGPU implementation but had only small fraction of computational load. For much advanced acceleration, it is important to consider various factors such as easiness of implementation, fraction of computational load and bottleneck in data transfer between GPU and CPU.

Key Words: GPGPU, nuclear reactor physics calculation, SCOPE2

1. INTRODUCTION

Requested level of safety and economy of the nuclear power plant become higher and higher. Therefore, development of advanced methods and calculation codes for nuclear applications are actively done with powerful computing resources. However, when the calculation model becomes much more detailed, the calculation cost (ex. calculation time) becomes also higher. There are some approaches to overcome this problem such as parallel computing and/or multi-threaded programming. In this study, a different approach based on GPGPU (General-Purpose computation using on Graphics Processor Unit) is applied for the acceleration of nuclear reactor physics calculations. GPU (Graphics Processor Unit) is a unit of the computer hardware for graphics rendering. GPGPU is a way of doing computation in completely different manner with that in the conventional algorithm on CPU. Although it requires deep insight and programming skills to transform the conventional algorithm to that suitable for GPGPU, performance in successful implementation is quite attractive. That is the one of the major reason why GPGPU is becoming popular in scientific applications. There is another fact that performance rise of GPU is much steeper, evolving approximately 4 times faster, than that of CPU. In other words, GPU is one of the very promising acceleration hardware for general computations that had been exclusively done on CPU. The trend of calculation performance of GPU and CPU over the last four years is shown in Figure 1.

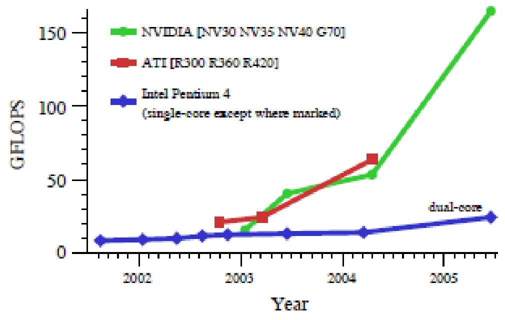

Figure 1. Calculation performance of GPU and CPU over the last four year [3]

GPU can be recognized as a vector processor that is controllable by user-defined programs. At the very first stage, programs must obey restricted data structures and limited flow control models. Recently, GPU becomes to enjoy a flexible programming scheme with structured data model and flow control such as the branch calculation ("if" in FORTRAN) and iteration ("do" in FORTRAN), but it still has a lot of limitations. Therefore, GPGPU was firstly applied in the field of computer graphics and games to simulate physical phenomena with approximated model and algorithms. In this study, however, we applied GPGPU to real application code on nuclear reactor physics calculation.

2. FEASIBILITY STUDY OF GPGPU

CPU calculation cannot be easily replaced with GPU calculation, because GPU is specific processor for the computer graphics; it is not aimed at general purpose computing. The programming style for GPU is optimized to manipulate graphics data. Therefore, if target algorithm in CPU calculation is not similar to that for manipulation of graphics data, it must be firstly transformed different algorithms that is suitable to the graphics data manipulation.

In typical computer architecture, GPU can treat only data on VRAM (Video Random Access Memory). Therefore, the data on main memory are necessary to be transferred to VRAM first for GPU to process them. And the calculation results on VRAM are transferred back to the main memory through the BUS such as PCI-express on the latest PCs. that is a data path among processors. A flowchart on data transfer between CPU and GPU is shown in Figure 2. There is a certain overhead for data transfer through the BUS because of the arbitration process.

Therefore, the users be conscious of the overhead and here to pay attention to structure and amount of data to be transferred.

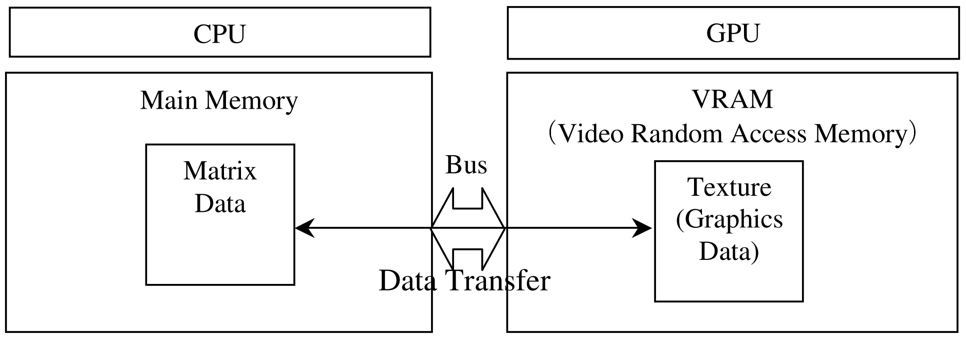

Figure 2. Data transport between GPU and CPU

The graphics data which are treated by GPU represents information for color and position of textures and vertex, respectively; it can be recognized as numerical data on VRAM to be operated by GPU anyway. Therefore, in the GPGPU, target data that is originally used in CPU calculation is transformed into data format of VRAM which is a set of ‘struct’ consists of 4 float data; RGBA (Red, Green, Blue and Alpha(transparency)) for color, XYZW (x-y-z axis and w (depth position)) for vertex. The 4 ‘struct’ data is a unit for data representation on VRAM; for example, RGBA data represents a pixel and aggregation of pixels is called texture that can be mapped on the object on the screen. In matrix calculations, numerical data to be operated are allocated as texture data in VRAM. Then, one instruction of GPU can treat one pixel at a time and all the pixel data can be processed simultaneously since GPU has many pipeline for data process to scale performance. This data operation by GPU is highly efficient and much faster than the same operation by CPU. So GPU can be recognized as a SIMD (Single-Instruction Multiple-Data) processor and this is a basic idea of GPGPU. Basically, in GPU calculation, data operation must be done within a pixel; any float data in the same pixel can be combined together such as R+G or X/Y. However data components on different pixels cannot be directly operated together because of the nature of GPU operation. Reduction of pixel data can be performed to summarize the data or to find the maximum/minimum of the data. So data structure is one of the key for successful GPGPU calculation.

A program that control GPU is called a shader program. It has specific language structure for graphics rendering. At the present, the Cg and the GLSL are provided as shader programming languages which are based on the C language. Shader programs are controlled through graphics libraries in higher level such as DirectX and Open GL.

One of the most suitable calculation style in GPGPU is calculation of NxN matrix shown in Figure 3. If the same manner of data operation can be applicable, the calculation speed becomes very fast because of inherent parallelism of GPU. For example, a diagram of the process for addition of 2x2 matrix is shown in Figure 4.

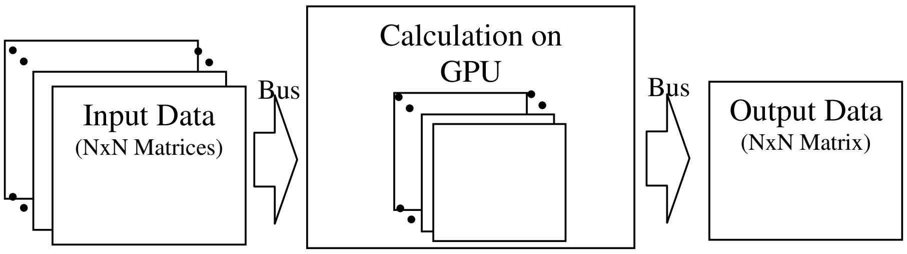

Figure 3. Matrix calculation on GPU

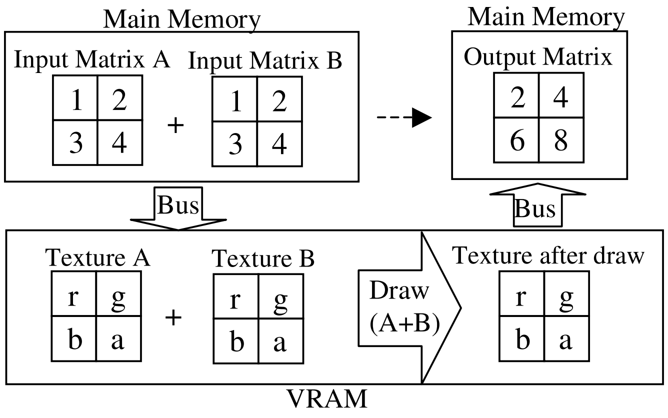

Figure 4. 2x2 Matrix calculation process on GPU

At first, the fundamental calculations based on four arithmetic operators and exponential-function was run on GPU and CPU in order to see the numerical and computational performances. The calculation was done on a machine that is shown in Table I.

Table I. Machine spec. and calculation environment

CPU	**Pentium 4 3.2GHz**
Main memory	2GB
OS	Cent OS 4.0 (kernel 2.6.9)
Compiler	gcc-3.4.3-g++ (Option –O3)
GPU	**nVIDIA GeForce 7800 GTX**
VRAM	256MB
Bus	PCI-express x16
Driver	nVIDIA 1.0-7676
API	OpenGL 2.0
Shader language	GLSL

2.1. Matrix calculations with four arithmetic operators

Calculations with four arithmetic operators (addition, subtraction, multiplication and division) were tested on CPU and GPU. The results of calculation performance on GPU and CPU are shown in Figure 5. When the matrix size becomes large, the calculation on GPU is 10 times faster than that on CPU for addition, subtraction and multiplication operators and 50 times faster at division operator. Since the precision on GPU calculation is defined as the IEEE754 (10^{-6} as single precision), calculation results agreed within the criterion between CPU and GPU cases.

When matrix size is small, the calculation on CPU is faster than on GPU. It is because that the ratio of preparation time of the calculation on GPU (ex. data structure arrangement, data transfer) to total calculation time is relatively large. When matrix size becomes larger, this ratio becomes smaller, which can be recognized in GPGPU calculation. Therefore, the calculation of small matrix on GPU has no advantage.

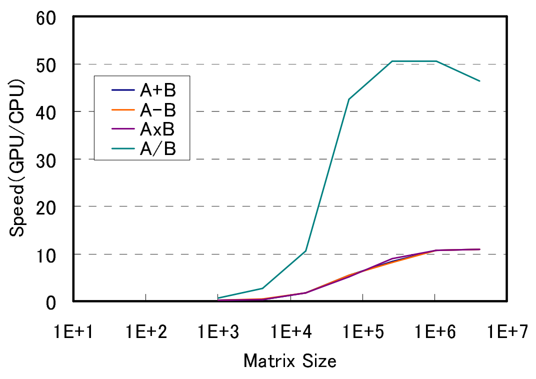

Figure 5. Four-function calculation on GPU and CPU

2.2. Calculation of exponential function

Next, calculations with exponential function were tested on CPU and GPU. The results of calculation performance on GPU and CPU are shown in Figure 6. When the matrix (the number of date for calculation) size becomes large, the calculation on GPU is 270-370-times faster than that on CPU. All the calculation results agreed between CPU and GPU cases within the criteria for single precision.

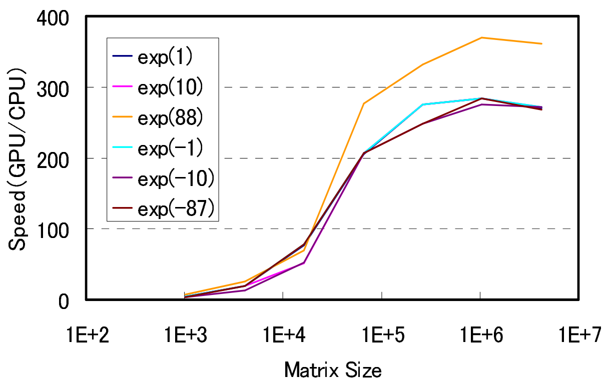

Figure 6. Exponential-function calculation on GPU and CPU

3. Application of GPGPU to SCOPE2

In the next step, we tried to apply GPGPU to accelerate the core calculation code SCOPE2 that is developed by NFI[1]. SCOPE2 solves the multi-group SP3 nodal-transport equations in 3-dimensional pin-by-pin geometry. In order to reduce computation time, SCOPE2 is fully parallelized with MPI and almost scalable running on PC clusters. In the present study, as the first step of our GPGPU application, the small geometry model of a fuel assembly is treated on single processor machine with a GPU.

SCOPE2 has several components for each functionality such as iterative calculation in fine-mesh geometry, feed-back calculation, synthesis of cross sections and so on. In selection of a target component to which GPGPU is applied, two factors were considered: easiness of implementation and fraction of computational load. If a component that has great fraction of computational load has complicated data structure, it is quite difficult to implement a program within the framework of GPGPU. This is because programmable shaders, user-defined programs that run on GPU and perform arithmetic operations on texture data, can only use limited data structure as mentioned in the previous section. Therefore it is needed to transform data structure from a CPU-fashioned manner to GPU-fashioned one. This work requires a lot of skills and efforts. It also requires help of CPU; temporary data must be transferred between CPU and GPU. However, overhead in data transfer via the PCI-express bus may spoil the performance gain by GPGPU. So one should carefully choose the target component and algorithms in GPGPU application. In our application, a component of response matrix for the multi-group SP3 equation[4] is selected for implementation of GPGPU.

One of the GPGPU application is to calculate the response matrix equation (1) which can be individually treated on each calculation mesh within fuel assembly geometry shown in Figure 7. This kind of equation are treated in SCOPE2 for different three moments within a energy group.

$$\xi_g^0 = \frac{-1+\exp(2h_x\kappa_g^0)-2\exp(h_x\kappa_g^0)h_x\kappa_g^0}{\left(-1+\exp(h_x\kappa_g^0)\right)^2\left(1+\frac{4}{h_x}D_g^{0*}\right)\kappa_g^0} \quad (1)$$

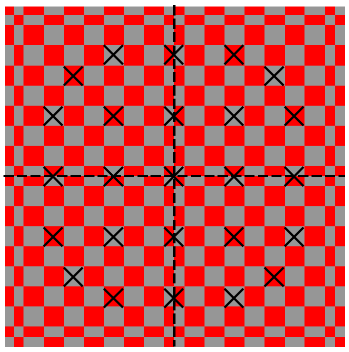
Figure 7. The calculation geometry of the fuel assembly

Total amount of response matrix calculations are very large since there are huge numbers of data to be computed: three order moments for nine group structure in 2-D (x-y) or 3-D (x-y-z) fine mesh geometry. The response equations are of linear matrix calculation and it can be independently calculated. Therefore, this application of GPGPU is very effective; there is other advantage with exponential calculation as mentioned in section 2.2. When the number of data becomes lager, the calculation speed becomes faster. It is very important to note that the calculation time of this part in SCOPE2 calculation is limited to a small fraction of all the execution time, so acceleration is relatively modest.

The result of comparison between the calculation time of CPU and GPU is summarized in Table II. When the number of data treated in the problem becomes large, the calculation on GPU is faster than that on CPU; the number of axial node becomes large, therefore, the calculation on GPU is faster than that on CPU. In the case of 3-D 12 axial nodes, the calculation on GPU is approximately 15% faster than that on CPU.

Table II. Comparison between the SCOPE2 assembly calculation time of CPU and GPU

Calculation time [s]	CPU	GPU	CPU/GPU
2-D	1.3	1.6	0.81
3-D 6 axial nodes	19.8	18.9	1.05
3-D 12 axial nodes	27.2	23.6	1.15

3. CONCLUSIONS

The applicability of GPGPU on neutronics calculation is examined. In the present study, the following facts were observed.

1. Matrix calculations with four arithmetic operators on GPU are 10-50 times faster than that on CPU.
2. Calculations of exponential-function on GPU are 270-370 times faster than that on CPU.
3. A test implementation for acceleration of SCOPE2 using GPGPU has been successfully done. The prototype code is approximately 15% faster than the original.

In the future study, application of GPGPU to other components of SCOPE2 and development of numerical library based on GPGPU will be examined.

REFERENCES

1. M. Tatsumi and A. Yamamoto, "Object-Oriented Three-Dimensional Fine-Mesh Transport Calculation on Parallel/Distributed Environments for Advanced Reactor Core Analysis," *Nuclear Science and Engineering*, **141**, pp.190-217 (2002)
2. http://www.gpgpu.org
3. J. D. Owens et al, "A Survey of General-Purpose Computation on Graphics Hardware," *EUROGRAPH 2005*, Dublin, Aug. 2005 pp.21-51 (2005).
4. M. Tatsumi and A. Yamamoto, "Advanced PWR Core Calculation Based on Multi-group Nodal-transport Method in Three-dimentional Pin-by-Pin Geometry," *Journal of Nuclear Science ant Technology*, **40**, pp. 376-387 (2003)