Emerging Technology about GPGPU

Enhua Wu[1,2]

[1]Faculty of Science and Technology
University of Macau
Macao, China
ehwu@umac.mo

Youquan Liu[1,2]

[2]State Key Lab of Computer Science
Chinese Academy of Sciences
Beijing, China
youquanliu@hotmail.com

Abstract— **By a rapid development of Graphics Processing Unit (GPU) in recent years, the programmability and highly parallel processing feature of GPU create a chance to allow the general purpose computation to be conducted on GPU, conventionally called GPGPU(General Purpose computation on GPU). A brief survey, in particular on the rationale of how the GPU architecture leads to GPGPU, is introduced in this paper, and various applications by taking advantage of GPGPU computation, including our works in this field on fluid simulation, as well as the works to be presented in this special session are also introduced.**

I. INTRODUCTION

Among so many VLSI nowadays, GPU is ubiquitous among desktops, laptops, PDAs, cell phones etc. Traditionally for many years, it is just used to accelerate some stages of the graphics rendering pipeline, which helps display the triangles onto screen. However, after the programmability available on this chip, GPU opens a door to developers to take advantage of its ALUs besides graphics processing. Compared with CPU in their architectures, GPU is more suitable for stream computations, it can process data elements in parallel with SIMD & MIMD capability. And in many cases, people gain great performance improvement on GPU over CPU. So a totally new technique called **GPGPU** (**G**eneral **P**urpose computation on GPU) emerges. And in recent years it became a hot research topic, not only in computer graphics domain, but also in other discipline areas.

In this paper, we will give a brief survey on GPGPU, and then present some application examples of GPGPU.

II. HISTORY OF GPGPU

A. *History of GPU*

When the first 3D graphics processing chip was born during the middle of 90's of last century, people were astonished to see its visual effects. And in 1999 NVIDIA began to sell its GeForce 256 processor known as NV10, which for the first time supported hardware-accelerated T&L(Transform and Lighting). From that time, the processor began to be called GPU (Graphics Processing Unit). We can find more details about the history of GPU from Wikipedia[1].

With the programmability of GPU emerging in GeForce 3 by NVIDIA from 2001, GPU entered into a new high speed era. At first people can write vertex shaders which run on the hardware, and then pixel shaders were supported quickly. And Read/write memory, floating point precision, and conditional execution were supported sequentially. At this time, GPGPU enters into a new era[2], and research progress has been made on the subject [3,4,5,6].

The generations of GPU actually follow the developments of Direct3D, one key component of DirectX SDK from Microsoft, which is devoted to game developments. So here we can also classify GPUs into the different generations in accordance with DX.

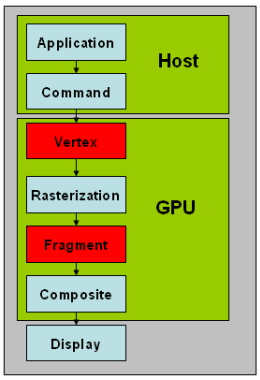

Figure 1. Pre-DirectX 10 hardware pipeline

As shown in Figure 1, the Pre-DX8 GPUs are not programmable beyond advanced texture blending capabilities; DX8 GPUs are the first to include an assembly language for vertex processing, as shown in red block of "Vertex". Such kind of GPUs includes the ATI Radeon 8500 and NVIDIA GeForce 3. Then DX9 GPUs extend vertex programs to enable data-dependent branching, and add a fragment program assembly language, as shown in red block of "Fragment", with floating point fragment precision supported. From Figure 1, we can see that the whole pipeline is still targeted for graphics tasks. By its working procedure, the application issues the rendering command, and sends the vertices data to GPU, then the vertex processor will do transform and lighting, followed

Sponsored by Grant of University of Macau, NSFC(60773030) and China Fundamental Research of Science and Technology 973 Fund (2002CB312102)

by rasterization of triangles into fragments, and with fragment processor, the generated fragments could be processed in some way. So here with these vertex or fragment processors, people could use them for some general-purpose computations.

In 2006, DX10[7] unified the whole pipeline as shown in Figure 2, in other words, there are no distinctions between vertex processor and fragment processor any more, for different tasks, these processors will be balanced between vertices-bound and fragment-bound processing. Furthermore, geometry shaders got supported from such kind of hardware. Actually in this way, GPU could be used as a floating-point coprocessor for any other non-graphics tasks, such as physics *etc*.

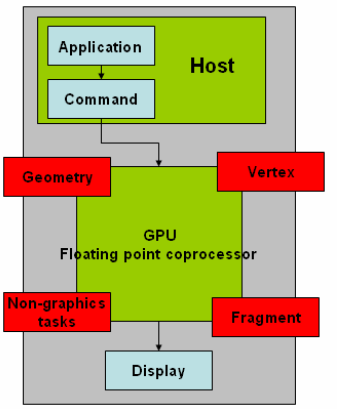

Figure 2. Latest hardware pipeline

B. *Why use GPU for general-purpose computation*

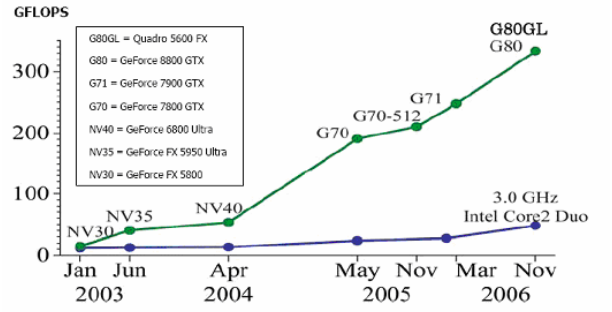

Figure 3. Floating-point operations on CPU and GPU(from NVIDIA)

Why so many researchers turn to GPU for some general-purpose computations? Before we think about this question, we should take a look at the latest GPU's architecture from NVIDIA. As seen from Figure 4, we can find many transistors on GPU are devoted to data processing (the green blocks stand for ALU).

From Figure 3, we can also find that the floating-point operations per second of GPU exceed that of CPU greatly. In other words, the computation power of GPU is stronger than that of CPU. And GPUs annual growth is greater than 2.0X, much faster than Moore's law. Due to its ubiquity, the computation power is more reachable than other hardware, and in terms of the computation cost, GPU for per GFLOPS is much lower than CPU.

Another important reason is regarded to the software platform progress. At the initial stage of GPGPU development, researchers had to write assembly instructions to conduct computation on GPU. Then after CG, HLSL, OpenGL Shading language came up, it turns out easier for people to write the hardware code. With CUDA interface available on DX10 GPU, it becomes more convenient for non-graphics people to take advantage of the power of GPU in their computation applications.

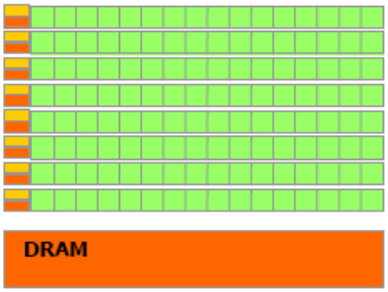

Figure 4. The architecture of GeForce 8800(from NVIDIA)

C. *Programming model of GPGPU*

1. Early stage GPGPU model

In early days, some researchers realized that GPU could be used to solve some non-graphics problems. At that time, GPU supported depth buffer and color blending operations which got some hardware acceleration. So, on such kind of chips, these researchers tried to map their problems to a depth buffer operation or color blending operation.

2. GPGPU Shaders model

After GPU chips open a programmable interface, people began to write shader codes to run on the hardware. Some researchers use vertex programs to solve some general-purpose problems, but most researchers use pixel programs to do so. The reason behind is that often the pixel processors are more powerful than the vertex ones, and it is also easier to fetch textures serving as the memory. However, during this stage, people still need to use graphics API such as OpenGL, Direct3D. In another word, people still need to map the problems to a graphics rendering procedure as shown in Figure 5.

GPU is often used as stream processor with the following steps:

- Draw a screen-sized quad
- Run kernel over each fragment (fragment program)
- Read stream data from textures
- Write results to frame buffer

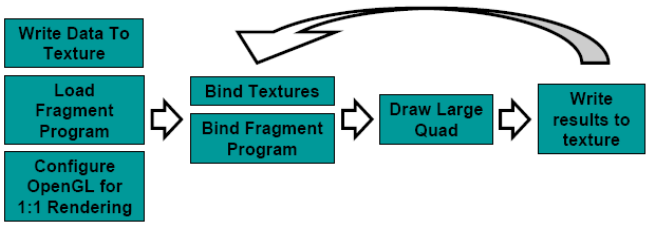

Figure 5. Programming model(from SIGGRAPH COURSE 2004)

Generally, data streams are represented as textures, Cg[8], OpenGL Shading Language (GLSL), or High Level Shading Language (HLSL) is used to write the kernel programs to operate the data on the graphics hardware.

3. Latest GPGPU model

As mentioned above, if people want to harness the computation power of GPU, they have to learn the graphics pipeline. To overcome these problems, NVIDIA unveiled the **C**ompute **U**nified **D**evice **A**rchitecture (CUDA) [9] in November 2006 which allows the use of the C programming language to code algorithms to execute on the GPU. CUDA-enabled GPUs include data parallel cache, which allows 128 processor cores in the GeForce 8 Series GPUs, providing up to 12,288 concurrent threads. Besides the flexible interface for programming, it also supports memory scatter which is impossible on pre-DX10 hardware, bringing more flexibilities to GPU.

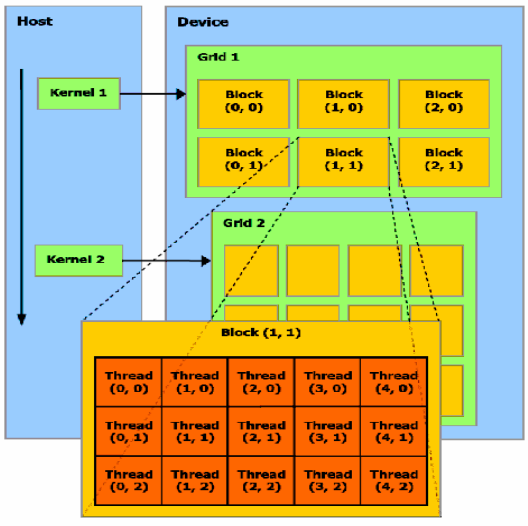

Figure 6. CUDA programming model(from NVIDIA)

As shown in Figure 6, people can write C code to run on GPUs just with certain configuration of thread management.

Similarly, AMD provides **C**lose **T**o **M**etal (CTM) SDK [10] and its successor AMD Stream SDK interface to help researchers and developers to use their GPU platform to accelerate the computation.

Actually before these two commercial SDKs are available, some researchers had made effort to hide the graphics concept when using GPU for general-purpose computation by steam languages such as Brook-for-GPU[11], and Sh[12]. Compared with Cg, GLSL, HLSL, such kind of codes are closer to those on CPU.

Most recently, the Khronos Group, which was founded by a number of leading media-centric companies, including 3Dlabs, ATI, Intel, NVIDIA, SGI, announced that they will work on **Open** **C**omputing **L**anguage (OpenCL)[13] specification to enable any application to tap into the vast gigaflops of GPU and CPU resources through an approachable C-based language. With such kind of interface, the developers would no longer worry about the variety of hardware platforms. With the release of DirectX 11.0 of Microsoft recently, compute shader joins GPGPU group. And Intel also released its many-core architecture-Larrabee[14], which is based on its Pentium architecture, not only targeted for graphics processing, but also targeted for general tasks.

III. APPLICATIONS OF GPGPU

Along with the development of GPGPU, we can find variety of GPGPU applications[2] coming up, including those in the fields of computational geoscience, finance, physics, which are all involved with computation-intensive, highly parallel processing tasks.

In this special session, the works from 4 different areas on GPGPU are reported. The investigations are respectively on FEM deformation simulation, image spatial diffusion, dynamic simulation of large scale forest, and Marching-Cube algorithm in medical image applications. Among from others, here we select several representative example applications to illustrate what GPU can do nowadays in the general-purpose domain instead of the traditional computer graphics area.

A. Basic Linear Algebra Operation

Basic linear algebra operations are the most common requirement for the general computation. With textures as the data streams, shader programs as the kernels, Bolz [15] & Krüger[16] provided solutions to represent sparse or dense matrix and vector by textures, and then execute the computation on GPU with shaders. For example, as shown in Figure 7, a dense matrix is represented as a list of 2D textures, used as the input data in the pixel shaders.

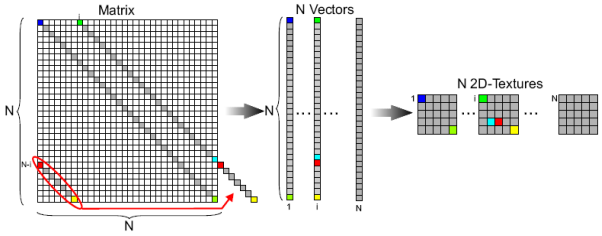

Figure 7. The representation of a 2D matrix, which is converted to 2D textures (from [15])

B. Fluid simulation

From Figure 8-10, we present our own work on fluid simulation accelerated with GPU[17,18]. Generally, to simulate the fluid motion physically, we have to solve the Navier-Stokes Equations,

$$\nabla \cdot \mathbf{u} = 0$$

$$\partial \mathbf{u}/\partial t = -(\mathbf{u} \cdot \nabla)\mathbf{u} + v\nabla^2 \mathbf{u} - \nabla p + \mathbf{f}$$

By using the GPU, the performance on computation speed up achieved more than one degree of magnitude.

C. Digital Signal Processing

Researchers from the Chinese Univeristy of Hong Kong presented a SIMD algorithm that performs the convolution-based Discrete wavelet transform (DWT) completely on a GPU, which brings significant performance gain on a normal PC without extra cost[19].

D. Database operation

Researchers from UNC used GPU for database operations[20] including relational query, conjunctive selection, and aggregation operations. For a million-scale database, GPU has the advantages over CPU in most situations.

E. Molecular Dynamics

Christopher *et. al* [21] used GPU to accelerate the calculation of cutoff pair potentials, one of the most prevalent computations required by many different molecular modeling applications, and the result shows 12 to 20 times faster than optimized CPU-only code.

F. HPC(High Performance Computing)

As early as 2004, some researcher declared that GPU has entered into the mainstream of computing[22].

HPC is very important to some large scale simulations, such as earth climate prediction, nuclear explosion simulation *etc*. In previous days, researchers have to turn to mainframe computer for help. However such kind of hardware is very expensive, highly unsuitable for most researchers and institutes. With the evolving of the architecture of GPU and its according software platform, GPU begins to enter HPC market. In 2007 NVIDIA began to sell its Tesla desktop system, which provides a sustainable price with a good enough performance.

Recently, France's CEA announced it will set up a supercomputer based on 1068 CPUs and 96 GPUs, bringing the total number of horsepower to 300 TFLops, here 192 TFLops from GPUs.

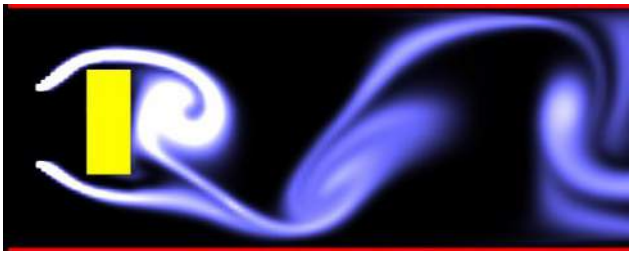

Figure 8. 2D Von Kamann Vortex Street

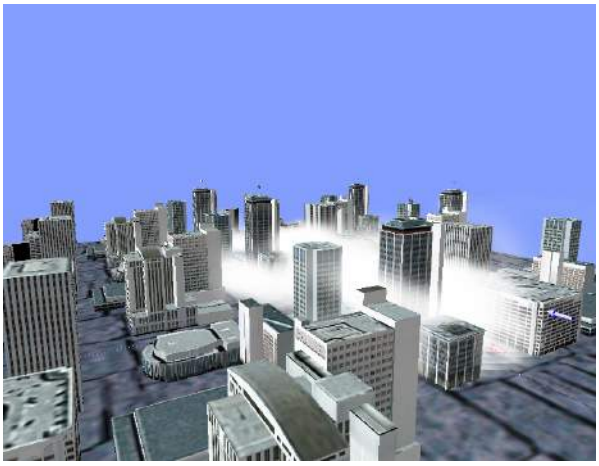

Figure 9. Complex flow inside a city

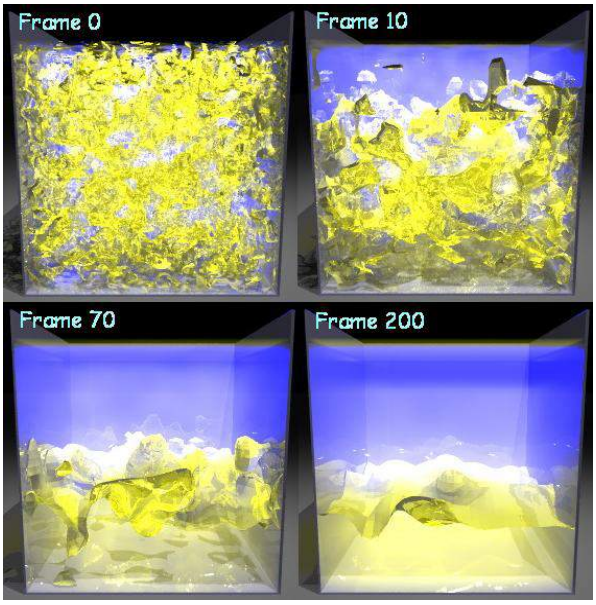

Figure 10. Oil-water decomposition

IV. Conclusion

In this paper we give a brief survey to GPGPU development, and present the programming models based on the evolving generations of GPUs. We explain the rationale on why so many researchers turn to GPU for variety of computation tasks, even non-graphics problems. To illustrate the power of GPU for general-purpose computation, we provide several examples in different applications. Through such kind of examples, we can surely see that the great advantage of GPU is able to be taken for computation-intensive, highly parallel number-crunching tasks.

Acknowledgments

The work has been supported by the Grant of University of Macau, NSFC(60773030) and China Fundamental Research of Science and Technology 973 Fund (2002CB312102).

REFERENCES

[1] GPU.Available:http://en.wikipedia.org/wiki/Graphics_processing_unit .

[2] GPGPU. Available: http://www.gpgpu.org .

[3] J. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A.E. Lefohn, T.J. Purcell, "A survey of general-purpose computation on graphics hardware", in Proceedings of Eurographics 2005, State of the Art Reports, Dublin, Ireland, pp.21-51, 2005.

[4] E.H. Wu, Y.Q. Liu, "General-purpose computation on GPU", Journal of Computer Aided Design and Computer Graphics, 16(5), May, 2004. pp.601-612.(in Chinese)

[5] E.H. Wu, "State of the art and future challenge on general purpose computation by graphics processing unit". Journal of Software, 15(10), Oct.2004. pp.1493-1504.(in Chinese)

[6] J.D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "GPU Computing", in Proceedings of the IEEE, 96, pp. 879-899, 2008.

[7] D. Blythe, "The Direct3D 10 system", ACM Transactions on Graphics (Proceedings of SIGGRAPH2006), 25(3), pp.724-734, 2006.

[8] W. R. Mark, R. S. Glanville, K. Akeley and M. J. Kilgard, "Cg: a system for programming graphics hardware in a C-like language", ACM Transactions on Graphics(Proceedings of SIGGRAPH2003), 22(3), pp.896-907, 2003.

[9] CUDA. Available: http://www.nvidia.com/object/cuda_home.html.

[10] CTM. Available: http://www.amd.com/us-en/Corporate/VirtualPressRoom/0,,51_104_543~114147,00.html

[11] I .Buck, T. Foley, D. Horn, J. Sugerman, K. Fatahalian, M. Houston, P. Hanrahan, "Brook for GPUs: stream computing on graphics hardware", ACM Transactions on Graphics, 23(3), ACM Press (Proceedings of SIGGRAPH2004), pp. 777-786, 2004.

[12] M. McCool, SD. Toit, T. Popa, B. Chan, K. Moule, "Shader algebra", ACM Transactions on Graphics, 23(3), ACM Press (Proceedings of SIGGRAPH2004), pp. 787-795, 2004.

[13] OpenCL. Available: http://www.khronos.org/news/press/releases/khronos_launches_heterogeneous_computing_initiative/

[14] L. Seiler, D. Carmean, et.al., "Larrabee: A many-core X86 architure for visual computing," ACM Transactions on Graphics (Proceedings of SIGGRAPH2008), 27(3), 2008.

[15] J. Bolz, I. Farmer, E. Grinspun, and P. Schreoder, "Sparse matrix solvers on the GPU: conjugate gradients and multigrid," ACM Transactions on Graphics (Proceedings of SIGGRAPH2003), 22(3), pp. 917-924, 2003.

[16] J. Krüger and R. Westermann, "Linear algebra operators for GPU implementation of numerical algorithms", ACM Transactions on Graphics (Proceedings of SIGGRAPH2003), 22(3), pp. 908-916, 2003.

[17] E.H. Wu, Y.Q. Liu, X.H. Liu, "An improved study of real time fluid simulation on GPU" (Conf. Invited Speech & Invited Paper at CASA2004: 17th International Conference on Computer Animation & Social Agents, Geneva). Computer Animation & Virtual World, John Wiley & Sons, 15(3-4), pp.139-146, 2004.

[18] E.H. Wu, "Fluid dynamics and real time simulation by contemporary GPU", Keynote Speech at Cyberworld 2006 EPFL(Swiss Federal Institute of Technology) with ACM/SIGGRAPH, EUROGRAPHICS , pp. 28-30, 2006.

[19] T. T. Wong, C. S. Leung, P. A. Heng and J. Wang, "Discrete wavelet transform on consumer-level graphics hardware", IEEE Transactions on Multimedia, 9(3), pp. 668-673, 2007.

[20] N.K. Govindaraju, B. Lloyd, W. Wang, M. Lin, D. Manocha, "Fast computation of database operations using graphics processors", in: Proceedings of SIGMOD, pp. 215-226, 2004.

[21] C. I. Rodrigues, D. J. Hardy, J. E. Stone, K. Schulten, and W.M W. Hwu, "GPU acceleration of cutoff pair potentials for molecular modeling applications", in Proceedings of the 2008 conference on Computing frontiers, pp. 273-282, 2008.

[22] M. Macedonia, "The GPU enters computing's mainstream", IEEE Computer Society, 36, pp. 106-108, 2003.