01 Oct 2007

A Survey of Neural Computation on Graphics Processing Hardware

Ryan J. Meuth

Donald C. Wunsch
Missouri University of Science and Technology, dwunsch@mst.edu

Recommended Citation

R. J. Meuth and D. C. Wunsch, "A Survey of Neural Computation on Graphics Processing Hardware," *Proceedings of the 22nd IEEE International Symposium on Intelligent Control, 2007*, Institute of Electrical and Electronics Engineers (IEEE), Oct 2007.
The definitive version is available at https://doi.org/10.1109/ISIC.2007.4450940

22nd IEEE International Symposium on Intelligent Control
Part of IEEE Multi-conference on Systems and Control
Singapore, 1-3 October 2007

A SURVEY OF NEURAL COMPUTATION ON GRAPHICS PROCESSING HARDWARE

Ryan J. Meuth, Donald C. Wunsch II
University of Missouri- Rolla
Dept. of Electrical & Computer Engineering
1870 Miner Circle,
Rolla, MO, 65401

***Abstract* - Modern graphics processing units (GPU) are used for much more than simply 3D graphics applications. From machine vision to finite element analysis, GPU's are being used in diverse applications, collectively called General Purpose Graphics Processor Utilization. This paper explores the capabilities and limitations of modern GPU's and surveys the neural computation technologies that have been applied to these devices.**

I. INTRODUCTION

In recent years consumer graphics processing units have experienced significant increases in performance, driven by increasingly realistic game simulations and popular multimedia demands. As a result, the graphics industry has leveraged a parallel processing model to provide a doubling of graphics computing capability every six months, as opposed to the 18 month doubling rate general computing processors, a trend that is illustrated in Figure 1. As these graphics processors become more capable and flexible, they have become desirable platforms for general computation. Owens [1] provides a comprehensive overview of the industry of general purpose computation on GPU's. However, Owens neglects to mention neural network applications on graphics processing units. Here, we provide an overview of these techniques, with associated challenges and limitations.

Figure 2 shows an overview of the graphics processing pipeline. On the host system side, the application generates a data structure to be rendered, consisting of a set of verticies and their corresponding colors that define a polygon.

Manuscript received January 30, 2007. This work was supported in part by The Boeing Company, M. K. Finley Endowment, and the National Science Foundation.
Ryan Meuth is with the Department of Electrical and Computer Engineering at the University of Missouri – Rolla, Rolla MO, 65401, USA. (phone: 573-341-4521; e-mail: rmeuth@umr.edu).
Donald C. Wunsch II is with the Department of Electrical and Computer Engineering at the University of Missouri Rolla, Rolla, MO, 65401, USA. (e-mail: dwunsch@umr.edu).

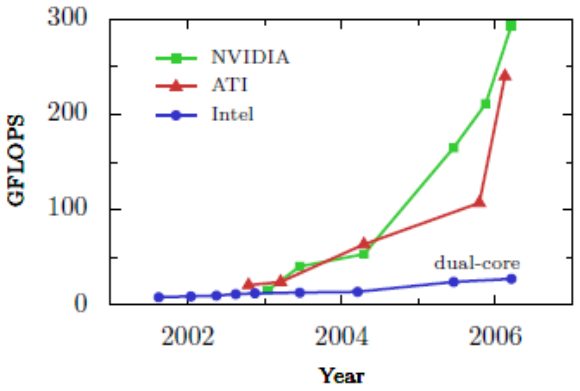

Fig 1. The exponential increase in performance of graphics processing units compared to the performance of Intel processors over the last 4 years. Figure Courtesy of Owens [1].

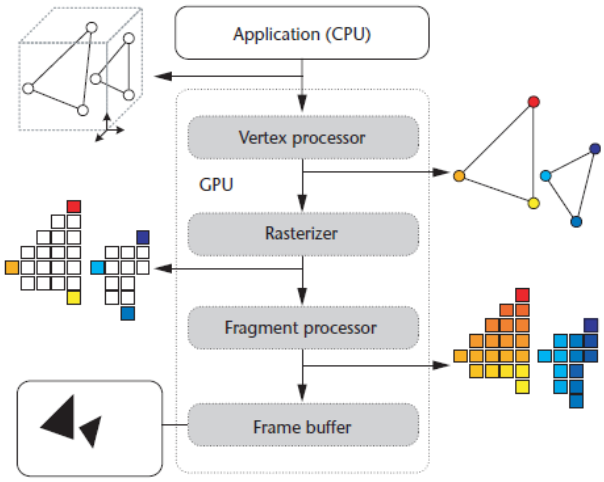

Fig 2. The graphics processing pipeline. Modern GPU's combine the vertex and fragment processor into a unified shader unit that is able to perform either of these functions. Currently, GPU's can include up to 128 unified shader units. Figure Courtesy of Goodnight [2].

This data structure is passed to the vertex processor, which is the first programmable unit in the graphics pipeline, which typically applies transformations to the vertices. The rasterizer then maps these coordinates to pixel locations, generating a set of fragments. These fragments are then passed to the fragment processor, the second programmable unit in the pipeline. The Fragment processor determines which fragments are to be drawn to the frame buffer, and then fills pixels with color information based on a program called a shader. Shader programs allow complex lighting and texture information to be mapped onto pixels. The frame buffer holds the completed image for output to a display.

To maintain high frame rates under increasingly graphically intensive applications the vertex and fragment processors have been implemented as a single-instruction, multiple-data parallel processing architecture. Modern graphics processors combine vertex and fragment processors into a generalized unified shader unit. At this writing, GPU's can include up to 128 unified shader units, operating at up to 1.3Ghz. As the entire pipeline is based on the 32-bit floating point data type, this yields a significant processing capability on the order of hundreds of GFLOPS in a single desktop frame. Additionally, bus enhancements now allow multiple graphics cards to work together in the same system [3].

For general purpose computing, the GPU architecture lends itself well to applications where the same calculations are repeatedly performed on large blocks of data [4]. In this way, particle systems, finite element analysis, image processing, and other numerical computation are well suited to utilize the GPU. However, the shader units of GPU's do not yet include efficient branching hardware, so algorithms utilizing data-dependant operations are difficult to implement effectively. Also, the data bus that hosts the GPU is often inefficient for small data transfers, so to achieve a reasonable speedup data must be operated on in batches [5]. Many of these difficulties have been overcome by creative algorithm design and implementation on the target systems. The widespread availability of these devices has allowed inexpensive high performance computing environments to be constructed that leverage both CPU and GPU capability to create a 'cluster of clusters' [6].

GPU shader programs are written in a language similar to assembly, and can be developed through a graphics programming interface, such as OpenGL or DirectX. High-level languages such as Brook, Sh, and RapidMind allow developers to use C-based languages to write shader programs, providing data abstraction and useful functions, reducing the learning curve of these devices.

II. GAME CONSOLES

Driven by increasingly complex video games and graphics as well as new entertainment media demands such as internet, digital photography and video playback, consumer video game consoles have become powerful general purpose machines. At the same time, these systems must be brought to the public at an affordable price point. Figure 3 compares the ratio of Floating Point Operations per Second (FLOPS) per dollar of several game consoles and Intel Pentium based systems. Here we can see that the cost-effectiveness of the latest generation of game consoles is an order of magnitude higher than that of any Intel-based system. These features make gaming consoles a highly desirable platform for inexpensive high performance computing systems.

Though the performance per dollar ratio of these systems is attractive, they are not without limitations, most notably in their interconnect ability. Only the last two generations of consoles have included networking capabilities, and then only one port is provided, limiting the efficiency of interconnects architectures in console-based clusters.

Until the latest generation of game consoles, the technology embedded in these products have often lagged behind the capability of personal computers at the time of release. However, the selling price of these devices makes them very competitive. In the previous generation of game consoles, this was recognized, and several attempts were made to utilize inexpensive game consoles as nodes in a super computing cluster. Very little success was made with the original Xbox [7], but researchers at the University of Illinois – Urbana Champaign succeeded in developing a 65-node computing cluster based off of the popular Playstation 2 video game console. This cluster was used for chemical simulations, and with a price point of $15,000 for the entire cluster, the system provided a high level of performance per dollar [8].

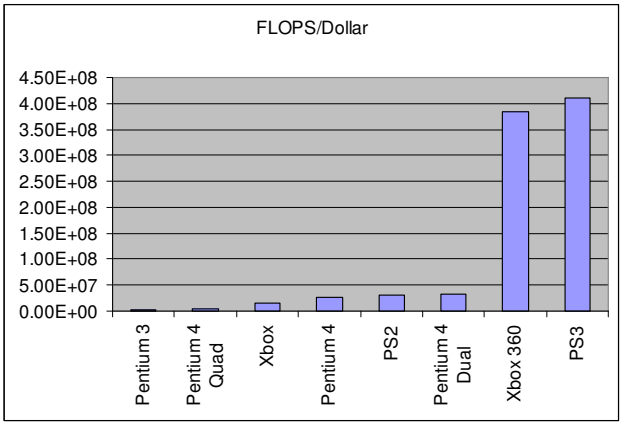

Fig. 3. Shows the FLOPS per dollar ratio of the past two generations of game consoles and Intel Processor-based systems. We can see that the latest generation of game consoles is several orders of magnitude more cost efficient than the latest Pentium-based systems.

The latest generation of game consoles differs from the former in that Microsoft's Xbox 360 and Sony's Playstation 3 both include new technologies that greatly surpass what is available in the home PC market. The Xbox 360 includes a tri-core Power PC processor operating at 3.2Ghz, theoretically providing a peak processing capability of 115.2 GFLOPS. The Xbox 360 surpasses PC-based computing capability by an order of magnitude, at a quarter of the cost [9]. Additionally, the Playstation 3 is capable of 205 GFLOPS provided by a nine-core processor called the Cell Broadband Engine cooperatively developed by Sony, IBM and Toshiba. The Cell consists of a single Power PC (PPE) based processor that manages 8 Synergistic Processing Elements (SPE) connected by an extremely high speed interconnect bus and shared memory. The PPE controls the SPEs like a cluster master node, implementing job queue, shared memory, and bus management. The Cell is unique in that the device is capable of managing 8 independent threads at full processor speed, with full branching and floating point operations available on each SPE [10, 11].

The Cell is also interesting in that it can be programmed using existing tools for graphics processing units, making much existing GPGPU work directly portable to these platforms.

Currently, no projects have been undertaken to develop high performance computing clusters based on the current generation game consoles. However, IBM will be using the Cell Processor in its next generation super computer, codenamed "RoadRunner." The machine will consist of 16,000 AMD Opteron cores matched with 16,000 Cell Broadband Engines, collectively rated at over 1 peta-FLOP/s. This will make it the most powerful super computer in the world by several orders of magnitude. It is to be built for the Department of Energy at Los Alamos National Laboratory in New Mexico. [12]

II. UNSUPERVISED LEARNING

The repetitive operations of self-similar data clustering methods have allowed many of these algorithms to be ported to the GPU, but the iterative and data-dependant nature of the algorithms have prevented large performance gains. Bohn implements Kohonen feature maps on GPU hardware. Even in '98 performance was improved by 5 times on large data sets using the GPU's of the time [13]. Hall uses a combination of the GPU and CPU to implement K-means clustering, using the GPU to perform distance computations and the CPU to perform template updates. Using this method, Hall was able to triple the performance of the algorithm over the CPU alone [14]. Similarly, Harris implements the Fuzzy C-Means algorithm, doubling its performance [15].

III. SUPERVISED NEURAL NETWORKS

Artificial neural networks attempt to capture the adaptability of biological neurons in a mathematical model for information processing. ANNs are very powerful tools that are highly parallelizable but also computationally expensive and match well with the GPU computing architecture. As a workhorse of the computational intelligence field, there exists a high demand for this acceleration. As a highly analytic structure, neural networks can be reduced to a series of matrix operations, and thus are easily parallelized, as the GPU is highly optimized to perform these kinds of operations. Zhongwen achieves a massive 200 times increase in performance of a multilayer perceptron implemented on graphics hardware over a typical CPU, enabling real-time soccer ball tracking on commodity hardware. Zhongwen uses the GPU to first extract a set of characteristics from image data, then applies a pre-trained MLP to these characteristics for classification. Zhongwen also provides several tips for ensuring efficient implementation of algorithms on GPU's. These tips include minimizing the pass count (or number of times a program must be applied to data), and minimizing data transfers between CPU and GPU sides [16].

Bernhard takes a different approach, implementing spiking neural networks for image segmentation, which achieves up to a 20 times increase in performance. Bernhard utilized a special counter on the GPU called the Occlusion Query, which tracks how many times a memory location has been modified by a shader program. Using this counter, he was able to efficiently compile the activations of neurons in a spiking neural network [17].

IV. CONCLUSIONS

It can be easily seen from this review that significant performance gains can be elicited from implementing neural network algorithms on graphics processing units. However, there is an amount of art to these implementations. In some cases the performance gains can be as high at 200 times, but as low as 2 times or actually less than CPU operation. Thus it is necessary to understand the limitations of the graphics processing hardware, and to take these limitations into account when developing algorithms targeted at the GPU. It should also be noted that all the reviewed papers in this document were operating on last-generation hardware. As of the end of 2006, the next generation graphics hardware has been released, which include an order of magnitude more shader units per processor, as well as improved branching capabilities. One can envision the possible capability of 256 programmable shader units working in parallel at 1.3 GHz each, in a single desktop box. Unfortunately none of the previous work has analyzed the performance of their algorithms relative to the number of computational units involved, which makes it uncertain exactly how new hardware will effect the performance of these algorithms. Additionally, efficient implementations of several neural network techniques have yet to be realized, including Adaptive Resonance Theory, making this an open and exciting area of research.

ACKNOWLEDGMENT

The authors are grateful to Rui Xu for helpful discussions.

REFERENCES

[1] John D. Owens, David Luebke, Naga Govindaraju, Mark Harris, Jens Krüger, Aaron E. Lefohn, Timothy, "A Survey of General-Purpose Computation on Graphics Hardware," *J. Purcell Computer Graphics Forum,* Volume 26

[2] Goodnight, N., Wang, R., Humphreys, G., "Computation on programmable graphics hardware" *IEEE Computer Graphics and Applications,* Volume 25, Issue 5, Sept.-Oct. 2005 Page(s):12 – 15

[3] "GeForce 8800 specifications," http://www.nvidia.com/page/geforce_8800.html, accessed November 9, 2006.

[4] Ekman M., Warg F., Nilsson J., "An in-depth look at computer performance growth." *ACM SIGARCH Computer Architecture News* 33, 1 (Mar. 2005), 144–147.

[5] Trancoso, P.; Charalambous, M.; "Exploring graphics processor performance for general purpose applications" *8th Euromicro Conference on Digital System Design, 2005. Proceedings.* 30 Aug.-3 Sept. 2005 Page(s):306 – 313

[6] Zhe Fan; Feng Qiu; Kaufman, A.; Yoakum-Stover, S. "GPU Cluster for High Performance Computing Supercomputing," *Proceedings of the ACM/IEEE SC2004 Conference 2004* Page(s):47 – 47

[7] "12 Node Xbox Linux Cluster," http://www.xl-cluster.org/index.php accessed January 19, 2007.

[8] "65 Node PS2 Linux Cluster," http://arrakis.ncsa.uiuc.edu/ps2/cluster.php accessed January 19, 2007. University of Illinois, Urbana-Champaign.

[9] "Comparison of Seventh Generation Game Consoles", www.wikipedia.com, http://en.wikipedia.org/wiki/Comparison_of_seventh-generation_game_consoles, Retrieved January 19, 2007.

[9] Pham D., Asano S., Bolliger M., Day M. N., Hofstee H. P., Johns C., Kahle J., Kameyama., Keaty J., Masubichi Y., Riley M., Shippy D., Stasiak D., Wang M., Warnock J., Weitzel S., Wendel D., Yamazaki T., Yazawa K.: "The design and implementation of a first-generation CELL processor." *Proceedings of the International Solid-State Circuits Conference* (Feb. 2005), pp. 184–186.

[10] Brown, Jeffry (2005-12-06). "Application-customized CPU design." IBM. http://www-128.ibm.com/developerworks/power/library/pa-fpfxbox/?ca=dgr-lnxw07XBoxDesign Retrieved on 2006-09-30.

[11] "Introduction to the Cell multiprocessor", *IBM Journal of Research and Development*, http://researchweb.watson.ibm.com/journal/rd/494/kahle.html, September 7, 2005.

[12] "IBM to Build World's First Cell Broadband Engine Based Supercomputer." IBM http://www-03.ibm.com/press/us/en/pressrelease/20210.wss (2006-09-06). Retrieved on 2006-09-11.

[13] Bohn C.A.: "Kohonen feature mapping through graphics hardware." *In Proceedings of the Joint Conference on Information Sciences* (1998), vol. II, pp. 64–67.

[14] Jesse D. Hall, John C. Hart: "GPU Acceleration of Iterative Clustering" *Manuscript accompanying poster at GP^2: The ACM Workshop on General Purpose Computing on Graphics Processors, and SIGGRAPH 2004 poster*, Aug. 2004

[15] Harris, C.; Haines, K.; "Iterative Solutions using Programmable Graphics Processing Units." *The 14th IEEE International Conference on Fuzzy Systems, 2005. FUZZ '05.* May 22-25, 2005 Page(s):12 – 18

[16] Zhongwen Luo; Hongzhi Liu; Xincai Wu: "Artificial neural network computation on graphic process units," *IEEE International Joint Conference on Neural Networks, 2005. IJCNN '05. Proceedings.* Volume 1, 31 July-4 Aug. 2005 Page(s):622 - 626 vol. 1

[17] F. Bernhard and R. Keriven. "Spiking neurons on GPUs." *In International Conference on Computational Science. Workshop General purpose computation on graphics hardware (GPGPU): Methods, algorithms and applications,* Reading, UK, May 2006.