

Data Visualization and Mining using the GPU

Sudipto Guha* Shankar Krishnan † Suresh Venkatasubramanian‡

April 17, 2005

Abstract

An exciting development in the computing industry has been the emergence of graphics processing units (the GPU) as a fast general purpose co-processor. Initially designed for gaming applications, today's GPUs demonstrate impressive computing power and high levels of parallelism and are now being used for a variety of applications far removed from traditional graphics rendering settings.

Perhaps the most powerful use of the GPU has been in visualization, which couples the raw computing power of the GPU with its extensive capabilities for rendering scenes. The GPU provides the required computing power and real-time interactive rendering capabilities and there are now GPU-assisted algorithms for many fundamental problems in data visualization and analysis, including such basic primitives as matrix operations, FFTs, wavelet transforms, clustering and mining data streams.

This is an exciting and fast developing area, and the tools and technique are now mature enough that researchers with no experience in using the GPU can use it to develop new data mining tools. The purpose of this tutorial is to introduce the KDD audience to the GPU and the programming model it represents, describe the ways in which one can program the GPU, and demonstrate a set of data mining primitives that have been implemented effectively on the GPU.

1 Introduction

The computing industry and community is seeing the emergence of a powerful subsystem: the graphics processing unit (GPU). These chips demonstrate impressive computing power (the NV40 chip contains over 225 million transistors), parallelism in pixel rendering, and rendering bandwidth. Modern GPUs are also cheap and ubiquitous; they are commodity hardware cards and almost all PCs today have some kind of advanced video graphics card. As a consequence, the GPU is now becoming a platform for high-speed general purpose computation, in many applications providing orders of magnitude better performance than CPU computations. To harness the raw computational power of the GPU, high level languages like Cg [2], HLSL [4], Sh [5], GLSL [3] and Brook [1] are being developed – these allow users to program the GPU quite easily, and in many cases *without needing a detailed knowledge of graphics*. The flexibility and ease of use of the GPU

*University of Pennsylvania, email:sudipto@cis.upenn.edu

†AT&T Labs – Research, email:krishnas@research.att.com

‡AT&T Labs – Research, email:suresh@research.att.com

has led to its use in a plethora of applications far removed from its graphics origins, in areas as diverse as image processing, robotics, geometry, scientific computing, GIS, and computer vision¹.

1.1 Visualization And Data Mining

Although the GPU has been used for a variety of applications, its strength lies in its ability to draw on its visualization capabilities (its fast rendering engine) and its fast processor. Some of the most effective uses of the GPU have been in the solution of problems in data analysis and visualization. Today, there are fast GPU implementations for many traditional data mining operators like the Fast Fourier Transform [24, 17, 6], matrix operations [21, 13, 7, 20], the discrete wavelet transform [15, 16, 28], nearest neighbors [18, 12], and others, leading to new algorithms for problems in clustering [12], mining of frequent item sets and computing heavy hitters [10], depth contours [19] performing reverse nearest neighbor operations and doing statistical correlations[11].

The key idea that unites GPU algorithms with data mining is the streaming nature of the GPU. As volumes of data have increased, researchers have been forced to model information as a stream of data where the processor is allowed to inspect the data in the order these items arrive. The memory of the processor is a small fraction of the memory required to store the entire data. Any data not explicitly stored cannot be accessed unless a new pass over the data is initiated. Thus, the processor is constrained to compute in passes (sometimes restricted to one) over the data. In a way a model of data streams forces random access to be considered as a valuable resource in dealing with massive data. Graphics chips operate in a fashion similar to streams; it renders a collection of triangles streamed in a typical pipelined fashion. The GPU can be viewed as a stream processor with pixel-level parallelism [14].

Today, we see an array of algorithms that use the GPU for analyzing and visualizing data streams. Viewed in totality, we see a collection of *tools*, like FFT, matrix-vector ops, wavelets, nearest neighbors, frequency moments etc, that have been used to build *applications* in clustering, frequent itemsets, RNN, regression analysis, nonparametric statistics, and other areas. Moreover, the growing body of theory and practice of GPU programming has fueled an array of courses, tutorials and workshops that address specific aspects of GPU programming, as well as many open source libraries that provide functionality for GPU-enhanced computations. Recently, researchers have also proposed a high level language **SCOUT** [23] designed specifically for data visualization on the GPU.

We believe that this is the right time to expose the KDD community to the capabilities of the GPU. Some of the reasons are:

- Visualization is an important and ongoing issue in analysis and data mining. A study of how programmable GPUs help us in fast visualization will be timely and useful to the community.
- The connections between GPU and stream processors allow the algorithms implemented on a GPU to have a "natural" streaming or pipelined flavor. Given the enthusiasm in the KDD community over data streams, visualization of streams will be a topic of interest, as already demonstrated by works such as StatStream [29].
- Several tutorials have been designed on using GPU for general purpose computing [22, 25] along with tutorials which focus more on graphics issues. A full semester course is being

¹A detailed listing of GPU applications is far beyond the scope of this proposal; see the website <http://www.gpgpu.org> for more details.

taught by one of the authors at UPenn [27] on using the GPU for general purpose computing. An abstraction of paradigms and algorithmic framework of how to use the GPU for data analysis tasks is emerging from this body of work which will be of interest to the mining community.

- A systems level picture is also emerging from the various implementations that use the GPU. Several data analysis primitives have been shown to work well with the GPU and several applications have used the GPU successfully as well. In general, there is now a growing understanding of when the GPU is effective for accelerated computations, and equally important, when it is not.
- There is a growing code-base of GPU tools and online resources, as well as intense activity in higher-level language development for the GPU. These allow more expressive and complex computations to be performed in the GPU via high-level programming abstractions that require only a minimal (or no) knowledge of graphics, thereby simplifying implementation efforts considerably.

2 Intended Audience

This tutorial will be directed at a general KDD audience. No background in graphics or GPU programming will be assumed, and the lectures will focus on a diverse set of applications, with detailed self-contained examples. The examples will naturally bring forth the programming paradigms used to implement data mining algorithms on the GPU.

3 Tentative Lecture Plan

The planned length of this tutorial is 3 hrs, to be broken into three units.

3.1 An Overview

In the first unit, we will start by present a brief overview of broad trends in GPU design that have resulted in the fast processing speeds of modern graphics cards. We will establish the connection between the streaming nature of the pipeline and its performance, drawing on material first discussed by Pat Hanrahan [14]. We will also be drawing from the course at Penn [27].

We will follow this with a series of brief motivating examples demonstrating how the GPU has been used for data analysis. We will cover topics ranging from the k -means algorithm of Hall and Hart [12], the streaming data mining work of Govindarajuet al [10], the use of matrix-vector operations for fluid flow simulation [7, 20], and the work on computing correlation, lines of regression and reverse nearest neighbors by the lecturers [11].

We will conclude this unit with a detailed look at the graphics pipeline, demonstrating how its various stages can be viewed as parts of a generic computational pipeline.

3.2 Tools

In this unit, we will focus on various data analysis primitives, and sketch their implementations on the GPU. The tools we will consider are matrix operations [21] (multiplication, addition,

matrix-vector multiplication, inner-products), convolutions, Fourier transforms [17] and the discrete wavelet transform [28], nearest neighbor calculations [18], and quantile estimation on data streams [10].

In all cases, we will use GPU primitives to describe the implementation of the tool. The tools will be used also to illustrate the manner in which computational concepts like variables, control flow, data structures and arithmetic operations are mapped to the GPU.

3.3 Applications and Evaluation

Finally, we will put the pieces together, demonstrating how the basic tools developed in the previous unit can be used to build general data analysis and visualization applications. We will return to the examples described in the first unit, describing in greater detail how the GPU may be used to do k -means clustering [12], compute frequent item sets and heavy hitters [10], compute non-parametric data estimates like depth contours [19], perform regression and correlation operations on data streams (and replicate the functionality in tools like StatStream) [11], and do medical image analysis via the use of the FFT[26].

Another key topic that we will discuss in this unit is performance evaluation and benchmarking. Profiling the behavior of GPU applications is a difficult process, and we will make use of tools like GPUBench [8] to explain how this is done. We will also discuss the limitations of the GPU. There are many applications that are not suitable to implement on the GPU (either because of their inherent sequentiality, or because of excessive non-local data access, or for other reasons), and a good understanding of when one should **not** use the GPU is as important as understanding the scope of its applicability. One of the papers we will discuss in this respect is the work by Fatahalian, Sugerman and Hanrahan on the limits of matrix operations on the GPU [9].

3.4 Additional Tutorial Material

To aid in the presentation of the tutorial, we will draw on material from the various courses mentioned above. We will also use multimedia demonstrations (videos, code) to assist in our presentations. Many of these are open-source tools available on the web (GPGPU.org is a valuable resource), and others are taken from paper presentations and demos.

4 Biographies

4.1 Sudipto Guha

Sudipto Guha is an assistant professor in the Computer Information Sciences Department, University of Pennsylvania. He has previously worked for AT&T Labs – Research from 2000 to 2001 as a member of the technical staff after receiving his PhD from Stanford University. His research interests are primarily in design and analysis of algorithms for computation under constrained resources. He has worked in the fields of graph approximation algorithms for NP-hard problems, randomized algorithms and combinatorial optimization, efficient optimization in database query and mining, multi-pass data stream algorithms. He has authored several papers on data streams and is investigating the connections between mining data streams and the GPU.

4.2 Shankar Krishnan

Shankar Krishnan is a member of the Information Visualization department at AT&T Labs – Research. He received his Bachelor’s degree in Computer Science from the Indian Institute of Technology, Madras in 1991. He received his M.S. and Ph.D. degrees in Computer Science from the University of North Carolina, Chapel Hill in 1993 and 1997, respectively. His main research interests include 3D computer graphics, data visualization, computational geometry, and graphics hardware-based algorithms. He has authored several papers and has given a number of technical presentations and tutorials on these topics in leading conferences in computer graphics, computational geometry and visualization. Shankar has presented (as part of a team) full-day courses on the topic of graphics hardware-based algorithms at ACM SIGGRAPH in 2002 and 2003.

4.3 Suresh Venkatasubramanian

Suresh Venkatasubramanian is a member of the Information and Visualization Department of AT&T Labs – Research. Prior to joining AT&T, he did his Ph.D at Stanford University in 1999. His research interests include computational geometry, algorithms, data mining and data streams, algorithms for the GPU, and information geometry. He has written a number of papers on GPU algorithms that have been presented at the Symposium on Computational Geometry, the Symposium on Discrete Algorithms, the Symposium on Interactive 3D Graphics, and the Workshop on General Purpose Computing with the GPU (GP2). He has given numerous talks on applications of the GPU, and is currently teaching a graduate level course at the University of Pennsylvania on GPU programming.

References

- [1] BrookGPU. <http://graphics.stanford.edu/projects/brookgpu/>.
- [2] Cg. <http://graphics.stanford.edu/projects/brookgpu/>.
- [3] GLSL. <http://www.opengl.org/documentation/oglsl.html>.
- [4] HLSL. http://msdn.microsoft.com/library/en-us/directx9_c/directx/graphics/programmingguide/hlslshaders/programmablehlslshaders.asp.
- [5] Sh. <http://libsh.org/>.
- [6] ANSARI, M. Y. Video image processing using shaders, 2003.
- [7] BOLZ, J., FARMER, I., GRINSPUN, E., AND SCHRÖDER, P. Sparse matrix solvers on the gpu: conjugate gradients and multigrid. *ACM Trans. Graph.* 22 (July 2003), 917–924.
- [8] BUCK, I., FATAHALIAN, K., AND HANRAHAN, P. GPUBench. <http://graphics.stanford.edu/projects/gpubench/>.
- [9] FATAHALIAN, K., SUGERMAN, J., AND HANRAHAN, P. Understanding the efficiency of gpu algorithms for matrix-matrix multiplication. In *Graphics Hardware* (2004).
- [10] GOVINDARAJU, N., RAGHUVANSHI, N., AND MANOCHA, D. Fast and approximate stream mining of quantiles and frequencies using graphics processors. In *Proc. SIGMOD* (2005).

- [11] GUHA, S., KRISHNAN, S., AND VENKATASUBRAMANIAN, S. Are pictures worth a thousand queries ? visualizing data streams with graphics hardware. Manuscript, 2005.
- [12] HALL, J., AND HART, J. C. Gpu acceleration of iterative clustering. 2004.
- [13] HALL, J. D., CARR, N. A., AND HART, J. C. Cache and bandwidth aware matrix multiplication on the gpu. Tech. rep., University of Illinois, Urbana-Champaign, 2003.
- [14] HANRAHAN, P. Why Is Graphics Hardware So Fast. <http://www.graphics.stanford.edu/~hanrahan/talks/why/>.
- [15] HOPF, M., AND ERTL, T. Hardware based wavelet transformations. In *Workshop on Vision, Modeling, and Visualization* (1999), pp. 317–328.
- [16] HOPF, M., AND ERTL, T. Hardware accelerated wavelet transformations. In *Proc. TCVG Symposium on Visualization* (2000), pp. 93–103.
- [17] JANSEN, T., RYMON-LIPINSKI, B. V., HANSEN, N., AND KEEVE, E. Fourier volume rendering on the gpu using a split-stream fft. In *Vision, Modelling and Visualization* (2004).
- [18] KENNETH E. HOFF, I., KEYSER, J., LIN, M., MANOCHA, D., AND CULVER, T. Fast computation of generalized voronoi diagrams using graphics hardware. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (1999), ACM Press/Addison-Wesley Publishing Co., pp. 277–286.
- [19] KRISHNAN, S., MUSTAFA, N. H., AND VENKATASUBRAMANIAN, S. Hardware-assisted computation of depth contours. In *13th ACM-SIAM Symposium on Discrete Algorithms* (2002).
- [20] KRÜGER, J., AND WESTERMANN, R. Linear algebra operators for gpu implementation of numerical algorithms. *ACM Trans. Graph.* 22 (July 2003), 908–916.
- [21] LARSEN, E. S., AND MCALLISTER, D. Fast matrix multiplies using graphics hardware. In *Supercomputing '01: Proceedings of the 2001 ACM/IEEE conference on Supercomputing (CDROM)* (2001), ACM Press, pp. 55–55.
- [22] MANOCHA, D., AND GOVINDARAJU, N. COMP290-058: GPGP: General Purpose Computation using Graphics Processors. <http://gamma.cs.unc.edu/GPGP/>.
- [23] MCCORMICK, P. S., INMAN, J., AHRENS, J. P., HANSEN, C., AND ROTH, G. Scout: A hardware-accelerated system for quantitatively driven visualization and analysis. In *IEEE Visualization* (2004), pp. 171–178.
- [24] MORELAND, K., AND ANGEL, E. The FFT on a GPU. In *SIGGRAPH/Eurographics Workshop on Graphics Hardware* (2003), pp. 112–119.
- [25] MOSEGARD, J., AND SANGILD, T. Gpgpu. http://www.daimi.au.dk/~mosegard/GPGPU_E04.
- [26] PHARR, M., AND FERNANDO, R. *GPU Gems 2 : Programming Techniques for High-Performance Graphics and General-Purpose Computation*. Addison-Wesley Professional, March 2005.

- [27] VENKATASUBRAMANIAN, S. CIS 700/010: GPU Programming and Architecture. <http://www.cis.upenn.edu/~suvenkat/700/>.
- [28] WANG, J., WONG, T.-T., HENG, P.-A., AND LEUNG, C.-S. Discrete wavelet transform on gpu. <http://www.cse.cuhk.edu.hk/~ttwong/demo/dwtgpu/dwtgpu.html>.
- [29] ZHU, Y., AND SHASHA, D. Statstream: Statistical monitoring of thousands of data streams in real time. In *VLDB* (2002), pp. 358–369.