System Analyses and Algorithmic Considerations in CUDA Implementations for Complex Simulations

Robert F. Lucas, Ke-Thia Yao, Gene Wagenbreth & Dan M. Davis
Information Sciences Institute, USC
Marina del Rey, California

ABSTRACT

Complex equation-based simulations of both man-made and natural phenomena present a wide spectrum of system architectures and algorithmic designs. Many of these require significant computational resources and many cause unacceptable delays in analysis. More critically, these delays often preclude interactive participation by Humans-In-The-Loop, an important capability when seeking a higher-level, system wide appreciation of the value of the object of the test and evaluation. Heterogeneous programming, e.g. *using Graphics Processing Units as accelerators, has shown theoretical promise. The advent of the Compute Unified .Device Architecture (CUDA) has, to a large degree, obviated the programming constraint. The authors relate their experiences with CUDA, discuss it implementation of various algorithms, and set forth some lessons learned. This should give programmers insights into which segments of their code would show significant speed-up and help project the improvement in program performance. This will be discussed in relation to algorithms often found in Hyper-spectral analyses.*

AUTHORS

Robert F. Lucas is the Director of the Computational Sciences Division of ISI at the USC. There he manages research in computer architecture, VLSI, compilers and other software tools. He's been the PI on the JESPP project since 2002. Prior to joining ISI, he was the Head of the High Performance Computing Research Department for NERSC at LBNL. Earlier he was the Deputy Director of DARPA's ITO, and a member of the research staff of IDA's Center for Computing Sciences. Dr. Lucas received B.S., M.S., and Ph.D. degrees in Electrical Engineering from Stanford University, Palo Alto, California. rflucas@isi.edu

Ke-Thia Yao is a project leader and research scientist at the University of Southern California Information Sciences Institute. His research has been centered on the JESPP project, which has the goal of supporting very large-scale distributed military simulations involving millions of autonomous agent entities. He has developed a suite of monitoring/logging/analysis tools to help users better understand the computational and behavioral properties of large-scale simulations and has designed the Scalable Data Grid for distributed data management of large archives. He received his B.S. degree in EECS from the University of California, Berkeley, and his M.S. and Ph.D. degrees in Computer Science from Rutgers University, New Brunswick, New Jersey. kyao@isi.edu

Gene Wagenbreth is a Systems Analyst for Parallel Processing at the Information Sciences Institute at the University of Southern California, doing research in the Computational Sciences Division. Prior positions have included Vice President and Chief Architect of Applied Parallel Research and Lead Programmer of Pacific Sierra Research, where he specialized in parallelization of Fortran programs. He is active in benchmarking, optimization and porting of software and has programmed on CRAY, SGI, Hitachi, Fujitsu, NEC, networked PCs, networked workstations, IBM SP2, as well as conventional machines. He received a B.S. in Math/Computer Science from the University of Illinois, Champagne-Urbana. genew@isi.edu

Dan M. Davis is the Director, JESPP Project, Information Sciences Institute (ISI), University of Southern California (USC), focusing on large-scale distributed DoD simulations. As the Assistant Director of the Center for Advanced Computing Research at Caltech, he managed Synthetic Forces Express, bringing HPC to DoD simulations. Prior experience includes work as a Software Engineer at the Jet Propulsion Laboratory and at a classified project at Martin Marietta. He saw duty in Vietnam as a Cryptologist in the USMC and he retired as a Commander, Cryptologic Specialty, U.S.N.R. He received B.A. and J.D. degrees, University of Colorado in Boulder. ddavis@isi.edu

Robert F. Lucas is the Director of the Computational Sciences Division of ISI at the USC. There he manages research in computer architecture, VLSI, compilers and other software tools. He's been the PI on the JESPP project since 2002. Prior to joining ISI, he was the Head of the High Performance Computing Research Department for NERSC at LBNL. Earlier he was the Deputy Director of DARPA's ITO, and a member of the research staff of IDA's Center for Computing Sciences. Dr. Lucas received B.S., M.S., and Ph.D. degrees in Electrical Engineering from Stanford University, Palo Alto, California. rflucas@isi.edu

System Analyses and Algorithmic Considerations in CUDA Implementations for Complex Simulations

Introduction and Background

With the 2007 release of an early version of the Compute Unified Device Architecture (CUDA) software developers kit, the rapid development and programmability of General Purpose Graphics Processing Units (GPGPUs) has become a practicality. The recognition that GPGPUs can accelerate the execution of a wide range of algorithms and that their costs are kept low has made this a viable path. (Lastra *et al.*, 2004) The low cost has been made possible by mass marketing of this computational technology to the untold multitudes of "gamers" in the consumer market. (Davis, 2010) This technology is held to be especially attractive to the Test and Evaluation (T&E) community.

One of the major issues is the time constraints put upon a very desirable computer paradigm, Real Time Hyper-spectral Scene Generation (RTHsSG). The T&E community has a continuing and critical need for advanced hyper-spectral sensor systems with the ability to generate, synthetic imagery in real time . These systems must be able to provide a compatible, repeatable test environment. This paper addresses the technologies that putatively can respond to that need in a way that is achievable an sustainable.

CUDA

CUDA is the path the authors have followed and are recommending. It is an easily mastered parallel computing architecture which was developed by NVIDIA. However, the authors caution against assuming that mastering the code will allow all journeyman programmers to become effective parallel programmers. It can be argued that parallel programming is more of an art than a science. Witness the fact that parallelizing compilers are still very primative and all involved parallel programming is hand coded.

CUDA is the computing engine in NVIDIA graphics processing units (GPUs) that is accessible to software developers through industry standard programming languages. Programmers who are familiar with the C programming paradigms will find CUDA very accessible.

Code developers will find access the native instruction set and access to the memory registers of the parallel computational elements in CUDA-capable GPUs. Even inexpensive NVIDIA GPUs have effectively become open architectures like CPUs. GPUs have a parallel "many-core" architecture, each core capable of running thousands of threads simultaneously - if an application is suited to this kind of an architecture, the GPU can offer large performance benefits. This approach of solving general purpose problems on GPUs is known as GPGPU.

While primarily for graphics functions, GPGPUs are used to perform various functions like physics calculations (physical effects like debris, smoke, fire, fluids) CUDA has also been used to accelerate non-graphical applications in computational biology, cryptography and other fields by an order of magnitude or more. An example of this is the BOINC distributed computing client.

CUDA developers with soon find that it provides both a low level API and a higher level API. This approach is available for all of the major platforms and development environments. Linux, Windows and Mac versions are downloadable from the NVIDIA web site.

RTHsSG

The GPUs power potential is large and growing at a faster rate than the CPU for which they would act as accelerators. However, while it is true that performance may often be measured in peak GigaFLOPS (billions of Floating Point Operations Per Second), true utility is better measured by productivity that takes into account many other issues, *e.g.* programming ease, availability of trained programmers, maintainability of code, durability of technology, *etc.* (Kepner, 2004) To respond to some of these needs and to gain insights into their magnitude and impact, the authors have presented three courses and a graduate student symposium on the programming of GPGPUs for accelerated computation.

Gene Amdahl's Law

Professor Gene Amdahl propounded a law to describe parallel speed-up. (Amdahl, 1967) The Amdahl's law is a model for the relationship between the expected speedup of parallelized implementations of an algorithm relative to the serial algorithm, under the assumption that the problem size remains the same when parallelized. For example, if for a given problem size a parallelized implementation of an algorithm can run 12% of the algorithm's operations arbitrarily quickly (while the remaining 88% of the operations are not parallelizable), Amdahl's law states that the maximum speedup of the parallelized version is $1/(1 - 0.12) = 1.136$ times as fast as the non-parallelized implementation.

More technically, the law is concerned with the speedup achievable from an improvement to a computation that affects a proportion P of that computation where the improvement has a speedup of S. (For example, if an improvement can speed up 30% of the computation, P will be 0.3; if the improvement makes the portion affected twice as fast, S will be 2.) Amdahl's law states that the overall speedup of applying the improvement will be:

$$\frac{1}{(1-P)+\frac{P}{S}}$$

To see how this formula was derived, assume that the running time of the old computation was 1, for some unit of time. The running time of the new computation will be the length of time the unimproved fraction takes, (which is 1 − P), plus the length of time the improved fraction takes. The length of time for the improved part of the computation is the length of the improved part's former running time divided by the speedup, making the length of time of the improved part P/S. The final speedup is computed by dividing the old running time by the new running time, which is what the above formula does.

Parallelization

In the case of parallelization, Amdahl's law states that if P is the proportion of a program that can be made parallel, *i.e.* benefit from parallelization), and (1 − P) is the proportion that cannot be parallelized (remains serial), then the maximum speedup that can be achieved by using N processors is:

$$\frac{1}{(1-P)+\frac{P}{N}}$$

In the limit, as N tends to infinity, the maximum speedup tends to 1 / (1 − P). In practice, performance to price ratio falls rapidly as N is increased once there is even a small component of (1 − P).

As an example, if P is 90%, then (1 − P) is 10%, and the problem can be speed up by a maximum of a factor of 10, no matter how large the value of N used. For this reason, parallel computing is only useful for either small numbers of processors, or problems with very high values of P: so-called embarrassingly parallel problems. A great part of the craft of parallel programming consists of attempting to reduce the component (1 – P) to the smallest possible value.

The Amdahl Fraction

The maximum speedup in an improved sequential program, where some part was sped up p times is limited by inequality:

$$\text{maximum speedup} \le \frac{p}{1 + f \bullet (p-1)}$$

where f ($0 < f < 1$) is the fraction of time (before the improvement) spent in the part that was not improved.

Therefore, making A twice as fast is better than making B five times faster. The percentage improvement in speed can be calculated as:

$$\text{percentage improvement} = \left(1 - \frac{1}{\text{speedup factor}}\right) - 100\%$$

- Improving part A by a factor of two will increase overall program speed by a factor of 1.6, which makes it 37.5% faster than the original computation.

- However, improving part B by a factor of five, which presumably requires more effort, will only achieve an overall speedup factor of 1.25, which makes it 20% faster.

Many have disputed this set of predictions. For example, popular author and computer veteran, Danny Hillis has criticized Amdahl's Law as being unnecessarily pessimistic in its assumption that the sequential portion of a program as being 5% (or 50%) of the execution time. In many applications, particularly with very large data sets, the amount of sequential code is closer to 0%, as essentially every data element can be processed independently.

Types of Algorithms found in RTHsSG

With this background, it may be useful to consider some of the more specific issues of interest to this paper. Enabling test and evaluation professionals to investigate various phenomena and evaluated specific systems is arguably more effective if it can be done interactively, especially with actual service members.

Hyper-spectral imaging collects and processes information from across the electromagnetic spectrum. Unlike the human eye, which just sees visible light, hyper-spectral imaging is more like the eyes of the mantis shrimp, which can see visible light as well as from the ultraviolet to infrared. Hyper-spectral capabilities enable the mantis shrimp to recognize different types of coral, prey, or predators, all which may appear as the same color to the human eye.

Humans build sensors and processing systems to provide the same type of capability for application in agriculture, mineralogy, physics, and surveillance. Hyper-spectral sensors look at objects using a vast portion of the electromagnetic spectrum. Certain objects leave unique 'fingerprints' across the electromagnetic spectrum. These 'fingerprints' are known as spectral signatures and enable identification of the materials that make up a scanned object. For example, having the

spectral signature for oil helps mineralogists find new oil fields.

Many algorithms are applicable to these analyses (Lugo-Beauchamp, 2004). Common subroutines are need to manipulate matrix data and perform various transforms, not uncommonly Fourier Fast Transforms. (Manolakis, 2002). These types of algorithms are very amenable to the power of the GPU and many functions such as this have already been written, tested and optimized by the research group up at NVIDIA.

Dimensions of Scalability

For the ... end-users.

The JFCOM Experience with JSAF

The Joint Forces Command (JFCOM) has the mission of leading the transformation of the defense establishment of the United States and to enable the U.S. to exert broad-spectrum dominance as described in Joint Vision 2010 (CJCS, 1996) and 2020 (CJCS, 2000). The Joint Concept Development and Experimentation Directorate, J9, is JFCOM's research arm. Having a research activity lodged within an operation command is a unique situation. That leads to experiments in which warfighters in uniform are staffing the consoles during interactive, HPC-supported simulations.

Experiments to model and simulate the complexities of urban warfare use well-validated entity-level simulations, e.g. Joint Semi-Automated Forces (JSAF) and the Simulation of the Location and Attack of Mobile Enemy Missiles (SLAMEM). These need to be run at a scale and resolution adequate for modeling all the convolutions of urban combat.

There is a long lineage of entity-level synthetic battlefield codes. They consist of representations of terrain that are populated with intelligent-agents representing: friendly forces, enemy forces and civilian groups. Experience has shown that to produce their behaviors in at timely manner, significant compute resources are required (Messina, 1997). Much of this is true because a major computational load is imposed in the performance of line-of-sight calculations between the entities. The inherently onerous "n-squared" growth-characteristics of an all-see-all program have been identified previously (Brunett, 1998). If several thousand entities need to interact with each other in an urban setting with vegetation and buildings obscuring the lines of sight, inter-node communications chaos and failure are often observed. This situation has been successfully ameliorated, but only partially so, by the use of an innovative interest-managed communication's architecture (Barrett, 2004).

Because of this, JFCOM required an enhanced Linux cluster of adequate size, power, and configuration to support larger and more sophisticated simulations, especially when there is a requirement for more than 2,000,000 entities within high-resolution insets on a global-scale terrain database. The cluster has been used occasionally to interact with live exercises, but more often has been engaged interactively with users and experimenters while generating only virtual or constructive simulations. (Ceranowicz, 2005) It had to be robust to reliably support hundreds of personnel committed to the experiments and it had to be scalable to easily handle small activities as well as larger global-scale experiments with hundreds of live participants, many distributed trans-continentally, as shown in Figure 1 below.

To enable JFCOM to meet these goals, a J9 team designed and developed a scalable simulation code that has been shown capable of modeling more than 1,000,000 entities.

This effort is known as the Joint Experimentation on Scalable Parallel Processors (JESPP) project (Lucas, 2003.) This work was begun under an earlier DARPA/HPCMP project named SF Express. (Messina, 1997) The early JESPP experiments on the University of Southern California Linux cluster showed that the code was scalable well beyond the 1,000,000 entities actually simulated, given the availability of enough nodes (Wagenbreth, 2005).

Early on in the JESPP Project, JSAF, was successfully fielded and operated using JFCOM's HPCMP-provided compute assets hosted at ASC-MSRC, Wright Patterson AFB, and at the Maui High Performance Computing Center (MHPCC) in Hawai'i. The dedicated compute power provided additionally allows for the easy identification, collection, and analysis of the voluminous data from these experiments, enabled by the work of Dr. Ke-Thia Yao and his team (Yao, 2005).

A typical experiment would find the JFCOM personnel in Suffolk Virginia interfacing with a "Red Team" in Fort Belvoir Virginia, a civilian control group at SPAWAR San Diego, California, participants at Fort Knox Kentucky and Fort Leavenworth Kansas, all supported by the clusters on Maui and in Ohio. The use of interest-managed routers on the network has been successful in reducing inter-site traffic to low levels.

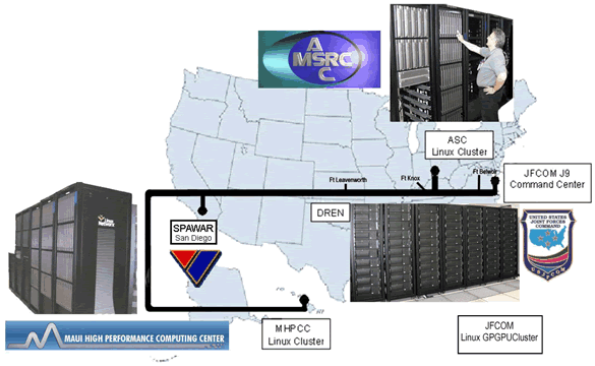

Figure 1 JFCOM Experimental Network

Forces Modeling and Simulation (FMS) is a field often driving new computer science due to its virtually unlimited needs and open-ended requirements. In a similar way, interactive computing is a new frontier being explored by the JESPP segment of FMS, coordinating with a few other user groups. Along these lines, the newly enhanced Linux Cluster capability has provided significant synergistic possibilities with other computational areas such as signals processing, visualization, advanced numerical analysis techniques, weather modeling and other disciplines or computational sciences such as SIP, CFD, and CSM.

Potential Algorithms

The use of existing DOD simulation codes on advanced Linux clusters operated by JFCOM was the implementation method chosen as being most efficient. This effort supplanted the previous JFCOM J9 DC clusters with a new cluster enhanced with 64-bit CPUs and nVidia 8800 graphics processing units (GPUs). Further, the authors were able to modify a few legacy codes. As noted above, the initial driver for the FMS use of accelerator-enhanced nodes was principally the faster processing of line-of-sight calculations. Envisioning other acceleration targets is easy: physics-based phenomenology, CFD plume dispersion, computational atmospheric chemistry, data analysis, *etc.* The computer was given the *Joshua*, the Hebrew commander.

Figure 2
Joshua

The first experiments were conducted on a smaller code set, to facilitate the programming and accelerate the experimentation. An arithmetic kernel from an MCAE "crash code" (Diniz, 2004) was used as vehicle for a basic "toy" problem. This early assessment of GPU acceleration focused on a subset of the large space of numerical algorithms, factoring large sparse symmetric indefinite matrices. Such problems often arise in Mechanical Computer Aided Engineering (MCAE) applications. It made use of the SGEMM (Single precision GEneral Matrix Multiply) algorithm (Whaley, 1998) from the BLAS (Basic Linear Algebra Subprograms) routines (Dongarra, 1993)

Figure 3. ... Looks Over the Shoulder

This ...

Figure 4. Cour...HPCMP

Any good instructor will advise being very careful about picking a good setting for instruction. All four of the efforts covered here were in excellent settings. The had sufficient room for each student to use their own ... stale.

Figure 5. Class Setting for First Course in Suffolk

A ... NVIDIA

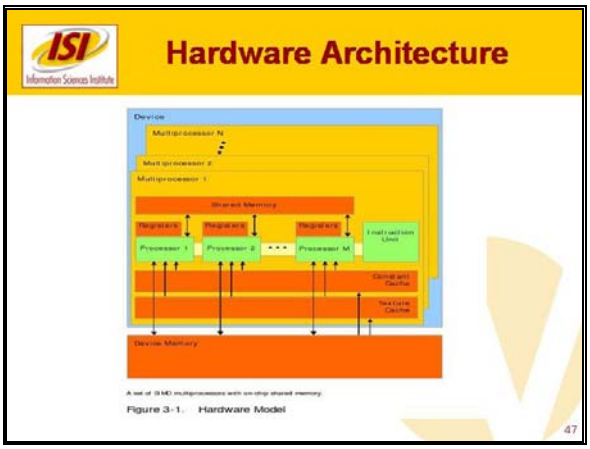

Figure 6. Slide showing NVIDIA Architecture

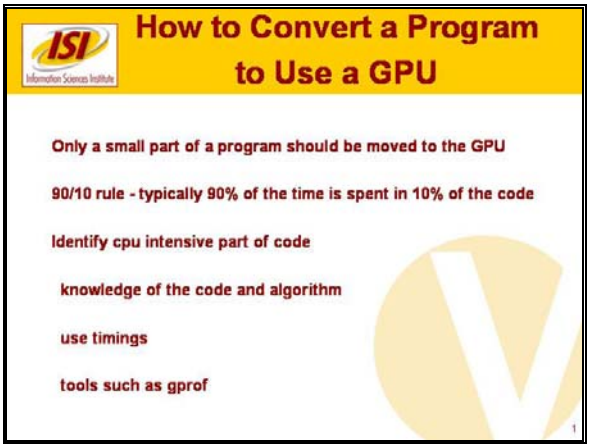

Figure 7. Slide Laying out Recommended Porting Approach

In ... space.

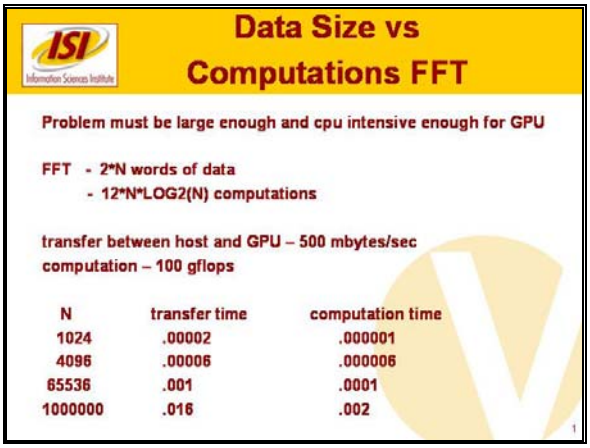

Figure 8. Slide Presenting Computation Parameters

The... site

Analysis and Lessons Learned

.... has been recorded.

Conclusions

The authors feel safe in concluding that the approach described above, as structured, achieved the immediate goals:

- Introduction of the new capability to users
- Establishing basic skills in CUDA
- Providing an overview of system analysis skills for heterogeneous programming

It seems clear that the courses were well received and appreciated. It similarly appears to be that a senior programmer or, better yet, a team of two or three senior programmers, could easily master CUDA programming, tailor the NVIDIA material for their own particular community, and effectively present a similar course to other programmers.

Whether this would be superior to just letting each individual programmer "boot-strap" themselves up via on-line and commercial materials is a less clear analysis. It would seem to the authors that this would depend on the size of the technical audience, the real heterogeneous or parallel programming experience held by the prospective instructors and other factors too numerous to name. In any case, the authors stand ready to discuss these or other issues that may be encountered by the prospective instructors.

Acknowledgements

This work was directed and funded by the Joint Experimentation Group at the Joint Forces Command and the authors wish to thank Major General Woods, Anthony Cerri, Jim Blank, and the entire J9 staff. The authors especially would like to acknowledge the direction, support and counsel of Rae Dehncke who has been the program manger and guiding light for this project. The authors would like to acknowledge the excellent technical support provided by the members of the Computational Sciences Division of the Information Sciences Institute, Gene Wagenbreth and John Tran, as well as the guidance and assistance from members of the Scalable Systems Division there, the Dr.s Ke-Thia Yao and Robert Neches. None of this could be accomplished without the help of the JSAF team Dr. Andy Ceranowicz, Mark Torpey, Bill Hellfinstine and many others.

References

Amdahl, G.M. (1967), Validity of the single-processor approach to achieving large scale computing capabilities. In *AFIPS Conference Proceedings,* Atlantic City, N.J.

Brooks, F.P., (1995) *The Mythical Man-Month: Essays on Software Engineering*, Anniversary Edition (2nd Edition), Addison Wesley Professional, Reading Massachusetts

CUDA (2010), *CUDA GPUs*, retrieved form the internet via IE on 12 May 2010 from: http://www.nvidia.com/object/cuda_gpus.html

Ceranowicz, A. & Torpey, M., (2005), Adapting to Urban Warfare, Journal of Defense Modeling and Simulation, 2:1, January 2005, San Diego, Ca

Davis, D.M., (2010), privately purchased a CUDA capable GPU at Fry's Electronics in Manhattan Beach California for $39 in January of 2010

Kepner, J., (2004), HPC Productivity: An Overarching View, *International Journal of High Performance Computing Applications*, Volume 18, Issue 4, Pages: 393 - 397

Kernighan, B.W. & Ritchie, D.M., (1998), *C Programming Language (2nd Edition)*, Prentice -Hall, Upper Saddle River New Jersey

Lastra, A., Lin, M., and Minocha, D., (2004 *ACM Workshop on General Purpose Computations on Graphics Processors*

Lugo-Beauchamp, W., Carvajal-Jimenez, C., & Rivera, W., (2004), Performance of Hyper-spectral Imaging Algorithms on IA-64, in the Proceedings of the IASTED International Conference on Circuits, Signals, and Systems, San Francisco, CA

Lucas, R.. F. & Davis, D. M., (2003) Joint Experimentation on Scalable Parallel Processors, in the *Proceedings of the Interservice/Industry Training, Simulation, and Education Conference*, Orlando Florida

Manolakis, D., (2004), *Detection Algorithms for Hyper-spectral Imaging Applications,* Project Report HTAP-8, Lincoln Labs, Massachusetts Institute of Technology, Lexington Massachusetts

Messina, P. C., Brunett, S., Davis, D. M., Gottschalk, T. D., (1997, April) Distributed Interactive Simulation for Synthetic Forces, In J. Antonio, (Chair), Mapping and Scheduling Systems, *International Parallel Processing Symposium*, Geneva, Switzerland.

U.S. News and World Reports, (2010), *The Best Engineering Schools*, retrieved from the internet on 12 May 2010 from: http://grad-schools.usnews.rankingsandreviews.com/best-graduate-schools/top-engineering-schools/rankings