PATH TO PETASCALE: ADAPTING GEO/CHEM/ASTRO APPLICATIONS FOR ACCELERATORS AND ACCELERATOR CLUSTERS WORKSHOP REPORT

VERSION 1.01

Last Updated: 09/03/09

NSF Award IDs: 0540688, 0626354

Document Authors
Volodymyr Kindratenko, Wen-mei Hwu, Robert Wilhelmson, Robert Brunner, Todd Martinez
NCSA
1205 W Clark Street, Urbana, IL 61801
kindr@ncsa.uiuc.edu
1 Overview

The Path to Petascale: Adapting GEO/CHEM/ASTRO Applications for Accelerators and Accelerator Clusters Workshop, was held at NCSA on April 2-3, 2009. Invites included researchers from the geosciences, astronomy and astrophysics, and computational chemistry communities with a keen interest in developing applications that can take advantage of the newly emerging processor and computational accelerator architectures. There were 102 registered attendees, with the vast majority from outside the University of Illinois at Urbana-Champaign. The 31 presentations included several technology presentations made by the technology representatives working in the area of application accelerators and many talks by application developers attempting to use the application accelerators and accelerator clusters. The workshop also included a poster session (consisting of 15 posters) and a hands-on session on obtaining access and using several accelerator-based systems. Participants were requested to fill in a survey about their involvement with application accelerators and the issues related to using such accelerators in practice.

1.1 Workshop Goals

• To review the roadmaps for application accelerators and software development ecosystems for accelerators
• To learn about new tools and frameworks that reduce development efforts and time to results
• To learn about using application accelerators within an HPC cluster environment
• To identify key challenges/issues in using application accelerators and clusters with accelerators
• To gain a better understanding how algorithm/application needs can be mapped into different types of application accelerators
• To encourage communication and sharing among groups attempting to use application accelerators

These goals were addressed through invited presentations, attendee presentations, and discussion. The agenda is given in the appendix, and all presentations are available at:

http://www.ncsa.illinois.edu/Conferences/accelerators/

2 Questionnaire Summary

There were 102 registered attendees; their names and affiliations are listed in the appendix. The attendees were primarily application code users and developers representing a diverse set of fields in which application accelerators are becoming increasingly popular.

A questionnaire was filled out by attendees before the end of the event. It total, 50 questionnaires were filled in and were returned. A compilation of responses appears in the appendix. The answers reveal that presently the vast majority of users favor GPU-based accelerator systems, with the Cell processor being a distant second choice. This is not surprising due to the wide availability and relatively low cost of NVIDIA GPUs that support general-purpose computations. The answers also show that the major roadblocks in using application accelerators remain in programming models and tools. Mapping algorithms to new processor architectures and rewriting existing codes using new programming languages and methodology to include application accelerators was frequently cited as the challenge
that stops computational scientists from widely adopting application accelerators. Other issues cited in this context were the lack of a long-term roadmap for application accelerators from the vendors and the risk in investing time and resources in technologies that are still immature and may turn out to be short-lived.

Attendees expressed interest in and support for the idea of an organization, such as NCSA, providing services to the scientific computing community to help application developers in using the newly emerging computing technologies. Such services could range from on-site training to the full application port. More than half of the attendees that filled in the questionnaire replied that they will definitely or very likely use such services, and just 11 attendees replied that they will not likely use such services.

3 Hands-on Session Summary
This session was designed to provide the information necessary for users to obtain access and to start using three accelerator-based clusters: the National Center for Supercomputing Applications GPU clusters, the Arctic Region Supercomputing Center Cell blade cluster, and the Laboratory for Computational Science and Engineering Cell tri-blade cluster. Users interested in having access to these systems were required to request the access following the procedures posted at http://www.ncsa.illinois.edu/Conferences/accelerators/systems.html before the event. The session was designed to help them get started using the systems. Presentations were given by Galen Arnold of NCSA, Anton Kulchitsky of ARSC, and Jagan Jayaraj and Pei-Hung Lin of the University of Minnesota.

4 Summary of Technology-oriented Talks
Wen-mei Hwu from the University of Illinois at Urbana-Champaign stressed the fact that tools are needed to do heavy lifting in parallel programming for application accelerators. He presented the status of work on GPU programming tools that simplify application development and performance tuning for NVIDIA GPUs and application portability between GPU and multi-core platforms. Tools under development at Illinois include CUDA-auto for implicitly parallel programming with data structure and function property annotations to enable auto parallelization, CUDA-lite for locality annotation programming to eliminate the need for explicit management of memory types and data transfers, CUDA-tune for parameterized CUDA programming using auto-tuning and optimization space pruning, and CUDA/OpenMP, a first-generation CUDA program with explicit, hardwired thread organizations and explicit management of memory types and data transfers.

Nathan Bell from NVIDIA focused on the CUDA platform evolution and new tools that NVIDIA is working on to simplify application development. One such tool is a C++ template library, Komrade, that will provide a simple way to implement commonly used algorithms. He also gave an overview of the latest effort to create a faster sparse matrix-vector multiplier that is frequently used in many scientific codes.

John Stone from the University of Illinois at Urbana-Champaign provided a detailed list of issues and solutions for multi-GPU application development, addressing such challenges as static workload distribution, load balancing, and multi-GPU runtime support. The examples were given in the context of VMD application port to a multi-GPU system. The multi-GPU framework developed in the course of this work will be released for use in other applications.

Vipin Sachdeva from IBM presented slides on the Cell processor and software development roadmap
and ecosystem.

Ben Bergen from Los Alamos National Lab presented a talk on the RoadRunner architecture and its programming models: host-centric and accelerator-centric. Electromagnetic Particle-In-Cell (PIC) code implementation on the RoadRunner was used as an example to demonstrate the issues with application porting and performance tuning. The bottom line of the presentation was that in order to achieve high performance on an architecture like RoadRunner, all parts of the system need to be tightly coupled and highly optimized.

Paul Woodward from the University of Minnesota gave an overview of first-hand experience in Cell programming for a fluid dynamics code. The approach used by Woodward’s team consisted of auto-translating existing Fortran code to Cell intrinsics. The auto-translation tools developed in the course of the work will be released to the public. The caveat, however, is that the original Fortran codes need to be written following a specific programming model and programming style. Nevertheless, impressive performance results were reported.

Michael Wolfe from the Portland Group, Inc., gave a status report on the PGI-GPU compiler. The compiler is designed to use OpenMP-like pragmas to separate the GPU code from the host code. Application implementation efforts using this approach appear to be similar to those when using OpenMP.

Shujia Zhou from NASA presented slides on an effort to accelerate climate and weather simulations with the Cell processor. He discussed porting strategies, such as data decomposition.

Mark Govett from NOAA Earth System Research Laboratory presented a talk on the status of Fortran → CUDA translator under development at NOAA for porting NOAA weather models to run on GPU-based systems. The need for such a translator was motivated by the fact that CUDA tools did not support Fortran and NOAA has a very large code base written in Fortran. The translator uses OpenMP-like pragmas for describing compute kernels for GPU implementation. Using the automated translator resulted in speedups of ~20x for several kernels used as test cases.

Jim Phillips from the University of Illinois at Urbana-Champaign talked about his experience with porting the NAMD molecular dynamics software to a GPU cluster. NAMD is written using the Charm++ distributed execution framework and its port to the GPU cluster required dealing with the Charm++ scheduler and load balancer in addition to porting the kernel itself.

Scott Pakin from Los Alamos National Lab presented slides on a programming model developed specifically for the RoadRunner system, called reverse acceleration, in which application execution is Cell SPE-centric and the application calls for CPU execution support only when the SPE is not capable of executing one or another task, such as invoking MPI communication. This approach has proven to be appropriate for hybrid architectures, such as RoadRunner.

Matthew Knepley from the University of Chicago presented slides on tree-based algorithms for GPUs.

Jonathan Cohen from NVIDIA presented a talk on the design of a CUDA-based CFD library.
5 Summary of Application-oriented Presentations

5.1 Geosciences
Mi Yan from the IBM Future Technology Solution Design Center presented a talk on IBM’s efforts in implementing seismic applications on the Cell Broadband Engine. Two particular techniques used in seismic imaging of interest were Kirchoff depth migration and wave equation algorithms. Algorithm reformulation, data decomposition, and various optimization techniques were applied in order to port these two algorithms to Cell co-processors, resulting in 15x and 3x speedups, respectively, when compared to a 2.8 GHz quad-core Intel Xeon microprocessor. Optimization techniques included manual code vectorization, data alignment, overlapping data transfers with computation, pre-computing lookup tables, etc.

Gordon Erlebacher from Florida State University presented slides on an effort to port a high-order finite-element earthquake modeling application, SPECFEM3D, to NVIDIA graphics cards using CUDA. Two implementation approaches were considered. For smaller models, the entire dataset is loaded and stored in the GPU-attached memory, while for larger models parts of the dataset are loaded and processed by the GPU in sequence. The latter approach has proven to be less efficient, resulting in speedups on the order of 5x as compared to 15x speedup from the small model implementation. The study also suggested that using mixed-precision calculations should be sufficient for this code and that there are numerous limitations in the current GPU hardware that prevent even higher performance.

John Michalakes from the National Center for Atmospheric Research (NCAR) discussed his effort to accelerate kernels from WRF on GPUs. WRF—a large collaboratively developed community non-hydrostatic weather model—has been ported to a number of HPC systems. In this study, several kernels have been targeted for GPU acceleration: WSM5 microphysics, tracer advection, and atmospheric chemistry. Speedups of 40x, 4x, and 2x, respectively, have been achieved and Michalakes pointed out that a considerable conversion effort was undertaken to achieve these speedups on the GPU.

Rory Kelly from NCAR presented a talk on an effort to implement solar radiation calculations on GPUs. The work is based on RADDEDMX subroutine taken from NCAR CAM 3.0 code. The subroutine is small in terms of the number of lines of code, about 1,000 lines of Fortran, but it is responsible for a large portion of calculations—a desirable code characteristic for implementing on computational accelerators. The code ported to GPU has proven to be bandwidth-limited due to the low PCIe interface bandwidth. Even though it runs about 11 times faster than a dual-core CPU version, it only utilizes about 2% of the peak FLOPS on the GPU card. Kelly concluded that achieving a large fraction of peak performance for a problem of low computational intensity is not realistic at this time.

David Yuen from the University of Minnesota presented a talk on tsunami wave propagation models implemented on GPUs. In this work, radial basis functions (RBF) are used to model 2D linear waves. The overall application is implemented in Matlab. To port the Matlab-based code, the researchers used the Jacket toolkit. The reported speedup is 8x for a large grid size and 3x for a small grid size. Future work includes development of a GPU version of a nonlinear shallow water code and the coupling of a tsunami to gravity waves in the ionosphere for early detection by satellites.

Chris Hill from MIT discussed some preliminary results in porting a parallel finite volume shallow water code to GPUs. So far simple 2D shallow-water kernels have been ported to GPUs, resulting in 8x speedup. The biggest issue the research team is faced with is the need for maintaining a common
source between the traditional multiprocessor code and the new GPU implementations. Also, limitations in the current GPU software stack make it more difficult to write MPI-based codes.

Shujia Zhou from NASA continued his earlier presentation on climate and weather simulations with Cell.

### 5.2 Astronomy and Astrophysics

Adrian Pope from Los Alamos National Lab presenting for Salman Habib described the RoadRunner Universe project which aims at accelerating cosmology simulations on the RoadRunner supercomputer. Currently code is being assembled and tested on development machines with the expectation of soon running the full code at full scale. Several cosmological simulation techniques were implemented and a few more are under consideration.

Richard Edgar from Harvard University presented results from an effort to develop a GPU-based data processing pipeline for the Murchison Widefield Array radio telescope. Computational requirements and the geographical location of the array make the use of conventional microprocessors impractical due to power restrictions. Several benchmark kernels have been ported to GPUs, demonstrating a substantial speedup. The work is under way to move the kernels to multi-GPU systems and to implement the entire data processing pipeline on these systems.

Jose Garcia from NCAR presented a talk on an effort to implement the Voigt Function used in the computation of theoretical spectro-polarimetric signals on IBM Cell and NVIDIA GPUs. He reported speedups of 3- to 5x for the Cell and 2x for the GPU. He also pointed out that the algorithm's performance is bound by the memory bandwidth and that this is reflected by the higher performance obtained on the Cell. He further noted that code porting to the GPU is somewhat simpler than code porting to the Cell processor.

Anshu Dubey from the University of Chicago discussed issues related to moving well-established large-scale codes, such as FLASH, to new architectures. Communities working with mature codes, such as FLASH, frequently decide to stay on the side while sustainable platforms emerge and tools for code translation to these platforms become available. For them, code portability is one feature that cannot be compromised. Also, code scalability to a large number of processor cores is something that the communities worked hard to achieve, and thus they are very reluctant to take a look at platforms that introduce new dimensionality to code scalability and portability.

Adrian Pope from Los Alamos National Lab presented some results on the implementation of a new cosmological n-body code for the RoadRunner Universe project presented earlier in the session.

### 5.3 Computational Chemistry

Ivan Ufimtsev from Stanford University presented a talk on an on-going effort to re-implement *ab initio* electronic structure codes on a multi-GPU system. Speedups as high as 650x have been reported for some of the models when comparing the execution time of the GPU implementation and the CPU-based GAMESS application. He noted that instead of porting kernels taken from an existing application, as commonly done, the entire application was developed from scratch with the GPU architecture in mind. They also made an effort to validate the developed code with regards to the limited numerical resolution on GPUs.

Andrey Asadchev from Iowa State University presented slides on an effort to develop an algorithm for computing electron repulsion integrals over uncontracted high angular momentum Gaussian function on GPUs.
Joshua Anderson from Iowa State University presented slides on the status of HOOMD, a general
purpose molecular dynamics code specifically developed from scratch for running on GPUs. Several
kernels have been hand-optimized to run different stages of the computational pipeline. Current work
focuses on extending the application to run on multiple GPUs.

Alistair Rendell from the Australian National University presented a talk on an effort to implement
sparse matrix vector multiplication on GPUs. The group systematically evaluated a number of
techniques on a range of sparse matrices. Then they empirically derived a decision tree that can be
used to choose an algorithm based on sparse matrix structure and size to yield the best performance.
The presentation also included a comparison of GPU and Cell implementations of a particle simulation
with Lennard-Jones interactions, concluding that Cell generally outperforms the GPU implementation by
a factor of 2. Rendell concluded that it was easier to get their codes to run initially on a GPU, but getting
them to run well was just as challenging.

Alan Aspuru-Guzik from Harvard University presented a talk on the use of GPUs for correlated quantum
chemistry calculations.

Peter Eastman from Stanford University presented slides on the implementation of OpenMM molecular
dynamics simulation on GPUs. OpenMM is a library for running molecular simulations on high-
performance architectures. The latest work includes GPU support for non-bonded interactions
computation. Non-bonded interaction kernels that use neighbor lists are particularly difficult to
optimize on GPUs due to out of order memory access. The presentation focused on optimizing such
algorithms.

6 Summary of Poster Session

6.1 Geosciences
Some optimization strategies for running weather codes efficiently on GPUs, Mark Govett, NOAA Earth
System Research Laboratory

GPU implementation of multiphase lattice-Boltzmann simulations, Stuart Walsh, Department of Geology
and Geophysics, the University of Minnesota

Modeling of Tsunami Equations and Atmospheric Swirling Flows with a Graphics Processing Unit (GPU)
and Radial Basis Functions (RBF), Jessica Schmidt, College of Saint Scholastica

Evaluating the Jaccard-Tanimoto Index on Multi-Core Architectures, Vipin Sachdeva, IBM Future
Technology Solution Design Center

Auto-tuning sparse matrix-vector "stencil" computations for GPUs, Chris Hill, MIT

Visualization and Computational Steering of Level Sets on the GPU for Seismic Interpretation, Benjamin
Kadlec, University Corporation for Atmospheric Research

6.2 Astronomy and Astrophysics
Accelerating Cosmology codes, Robert Brunner, Department of Astronomy, University of Illinois at
Urbana-Champaign
Solar wind properties at 1 AU, big time series analysis on the Cell cluster at ARSC, Anton Kulchitsky, Arctic Region Supercomputing Center, UAF

High-Performance Large Eddy Simulations (LES) of Turbulent Flows using Graphics Processing Units (GPUs), Aaron Shinn and Pratap Vanka, University of Illinois at Urbana-Champaign

Porting GENESIS (GENetic-Evolutionary Stokes Inversion Strategy) to NVIDIA's CUDA Architecture, Brian Harker-Lundberg and Kenneth Mighell, National Optical Astronomy Observatory

6.3 Computational Chemistry
Quantum Monte Carlo on Graphical Processing Units, Amos Anderson, Caltech

Liouville-von Neumann Molecular Dynamics, Jacek Jakowski, Emerson Center for Scientific Computation and Department of Chemistry, Emory University

Two-Electron Integral Evaluation on FPGA, Cell and GPGPU accelerators, Guochun Shi, NCSA

GPU acceleration of continuum quantum Monte Carlo Simulations, Ken Esler, NCSA

Real-time Registration of 3D Medical Images, Ramtin Shams, The Australian National University

7 Workshop Conclusions

7.1 Observations
The attendees were split between early adopters of accelerators and skeptics. The skeptics' main concern was that new computing technologies are introduced frequently and domain scientists simply do not have time to follow up with all the technology developments. From their perspective, existing HPC resources are adequate for solving many problems.

Early adopters tend to start writing codes from scratch, frequently stepping back and re-evaluating the suitability of the existing algorithms for the accelerator architectures. The best performance improvements reported so far tend to be from this type of efforts rather than from porting existing codes.

The majority of early adopters were still at the stage of technology evaluation and initial kernel porting to a single accelerator. A few efforts were starting to look beyond simple kernels and a single GPU or Cell accelerator. A few large-scale accelerator clusters became available recently for this work, such as Cell-based RoadRunner (LANL) and GPU based Lincoln (NCSA).

New users of application accelerators fall into one of three categories:

- The first user type is unwilling to invest any resources in porting. Only when an almost effort-free port can be made, e.g., with existing libraries, will these users consider application accelerators. Most users will start in this category and some will not progress beyond it. Libraries such as CUBLAS, CUFFT, and GPULib enable these users to leverage accelerators. Not all applications benefit from this approach as some may not use standard libraries and others may not use them in a way that enables a significant performance boost.

- The second user type is willing to experiment with accelerators but is concerned with code portability. For these users, the availability of advanced compiler technology to obtain the acceleration benefit from reasonably portable code is the key. Examples of such emerging tools...
include OmpeMP-style compilers for Cell and GPUs. Few companies offer alternatives for programming accelerator-enhanced systems using techniques such as pragmas to identify kernels or advanced compiler analysis to identify parallel operations. While this approach may not match the gains possible using native compilers and explicit parallelization, code portability is preserved. As with the first user type, some developers will stop at this point.

- The third user type is willing to go to great lengths to get the highest possible performance. For these users, native tools are the main development platform.

Researchers working with mature codes and with large-scale codes in particular tend to be very reluctant to practice on the bleeding edge of computing technologies. Mature codes have the advantage of being extensively validated and trusted in the community and porting them to newly emerging accelerator architectures will require another round of re-validation. Also, porting large-scale codes to a new architecture is a non-trivial task.

Long-term viability of the application accelerators is perhaps the most important issue that prevents many teams from porting their codes. Neither NVIDIA nor IBM was willing to discuss their future product roadmaps, making even the early adopters uncomfortable with regards to the future of their work.

It has been pointed out a number of times that many of the codes have been around much longer than the machines they were originally designed to run on. This is possible because the codes are written using languages that are portable across a range of systems. With the introduction of the application accelerators, a set of new languages and programming models is emerging such that code portability is no longer an option. The community fears that these new architectures will result in the creation of many code branches that are not compatible and portable.

The community is concerned with the complexity of the task of porting existing codes to new architectures. In many cases, significant code transformations are necessary, and frequently algorithmic transformations are required to achieve any meaningful performance improvements.

The maturity of the programming and debugging tools for application accelerators, and the availability of libraries, is another concern in the community. The community is enthusiastically awaiting the release of OpenCL for major GPU and multi-core processor architectures.

In some cases, reasons other than performance improvements have been cited as important for using accelerators. This is particularly true for applications in which compute resources must be located near scientific instruments and thus may have additional constraints, such as power requirements. An interesting point was the importance of flops/watt benefit of some of the accelerator technologies.

An interesting comment from NOAO was that collaboration with NCSA allowed a pilot study to be performed, which could then be leveraged to go to their sponsoring agency for further funding. This never would have been possible without a group that has the system and the knowledge necessary to help others get started with exploring the technology.

### 7.2 Recommendations

The potential of accelerators and accelerator clusters to enhance the performance of many scientific codes beyond what is possible on conventional architectures is evident. This potential may remain unrealized for many scientific applications unless directed efforts by the community, technology vendors, and funding agencies are implemented. These efforts should include:
• **Teaming application scientists and domain application developers with computer scientists or application specialists who are knowledgeable about accelerators and the computer systems that use them.** So far the majority of the efforts have been undertaken by domain application developers. The drawback of this approach is that application developers spend a lot of time on the idiosyncrasies of the new systems and programming tools and do not always manage to take full advantage of the accelerator capabilities. Teaming up with computer scientists or application specialists intimately familiar with these systems will help to produce better code and achieve better performance.

• **Developing performance models and application porting guides for application accelerators.** At present, many application developers are reluctant to start porting their codes to application accelerators because they cannot predict performance benefits without an actual prototype implementation, which by itself is a substantial effort. Performance models against which candidate application can be evaluated for their suitability for application accelerators need to be developed. Also, a set of guides on how to port applications to different accelerator platforms need to be created.

• **Developing better tools for implementation, debugging, and performance analysis on application accelerators and accelerator clusters.** Application developers are bogged down in hardware-specific coding. At present, programming application accelerators is much harder than it should be. Existing tools for application performance analysis, such as code profilers, are not suitable for accelerators; tools with similar capabilities for accelerators are nonexistent. Debugging tools do not provide a sufficiently accurate inside view of the accelerators. This state of the art in application development for accelerators makes it difficult for computational scientists to port applications. Substantial investments should be made in the development of tools that would enhance the productivity of application developers working with accelerators.

• **Implementing enhancements to the existing cluster software and run-time systems to provide a better support for accelerators.** At present, application accelerators are not treated as independent computing resources by the cluster management software or by the run-time systems used for parallel application execution on a cluster, e.g., MPI run-time. System administrators and application developers have to provision and use these resources manually, imposing a limited range of usage scenarios and programming models supported on the accelerator clusters. Enhancements to the existing run-time systems need to be introduced to allow for a more efficient and easier accelerator resources utilization and sharing among multiple users.

• **Setting up shared resources available for the application community at large.** While many teams have in-house access to a few compute nodes outfitted with accelerators, access to large-scale accelerator clusters is generally not available for most of them. Large-scale shared resources that are up to date and support a variety of usage scenarios are necessary for code development, performance tuning, and production runs.

• **Making available community training, education, and resource-sharing mechanisms.** At present, application developers typically refer to the training materials and resources provided by the technology vendors in order to get up to speed with using the accelerators. This approach, however, has the potential for a biased view on the technology and its applicability to solve a given problem. Instead, unbiased experience-based training materials should be created and made available to the end-user community.