High-Performance Computing with Accelerators

This issue of CiSE is based on work presented at the US National Science Foundation workshop, “Path to Petascale: Adapting Geo/Chem/Astro Applications for Accelerators and Accelerator Clusters,” held at the US National Center for Supercomputing Applications (NCSA) in early 2009. The workshop was designed to raise awareness about the emergence of accelerator-based high-performance computing (HPC) resources among computational scientists from the geosciences, computational chemistry, and astronomy and astrophysics communities and to help them get started in using these resources.

Volodymyr Kindratenko, Robert Wilhelmson, Robert Brunner, Todd J. Martínez, and Wen-mei Hwu
The State of Affairs
In the past few years, a new class of HPC systems has emerged. These systems employ unconventional processor architectures—such as IBM’s Cell processor and graphics processing units (GPUs)—for heavy computations and use conventional central processing units (CPUs) mostly for non-compute-intensive tasks, such as I/O and communication. Prominent examples of such systems include the Los Alamos National Laboratory’s Cell-based RoadRunner (ranked second on the December 2009 TOP500 list) and the Chinese National University of Defense Technology’s ATI GPU-based Tianhe-1 cluster (ranked fifth on the same TOP500 list).

Currently, there’s only one large GPU-based cluster serving the US computational science community—namely, Lincoln, a TeraGrid resource available at NCSA. This will be augmented in the near future by Keeneland, a Georgia Institute of Technology system funded by NSF Track 2D HPC acquisition program. On the more exotic front, Novo-G cluster, which is based on Altera field-programmable gate array (FPGA), is deployed at the University of Florida’s NSF Center for High-Performance Reconfigurable Computing (CHREC). By all indications, this trend toward the use of unconventional processor architectures will continue, especially as new GPUs, such as Nvidia’s Fermi, are introduced.

Despite hardware system availability, however, the computational science community is currently split between early adopters of accelerators and skeptics. The skeptics’ main concern is that new computing technologies are introduced frequently, and domain scientists simply don’t have time to chase after developments that might fade away quickly.

In particular, researchers working with mature and large-scale codes are typically reluctant to practice on the bleeding edge of computing technologies. From their perspective, the accelerator-based systems’ long-term viability is a key question that prevents them from porting codes to these systems. Many such codes have been around much longer than the machines they were originally designed to run on. This continues to be possible because the codes were written using languages (C and Fortran) supported by a range of HPC systems.

With the introduction of application accelerators, new languages and programming models are emerging that eliminate the option to port code between “standard” and “non-standard” architectures. The community fears that these new architectures will result in the creation of many code branches that are not compatible or portable. Mature codes have also been extensively validated and trusted in the community; porting them to newly emerging accelerator architectures will require yet another round of validation.

In contrast, early adopters argue that existing HPC resources are insufficient—at least for their applications—and they’re willing to rewrite their codes to take advantage of the new systems’ capabilities. They’re concerned about (but willing to endure) the complexity of porting existing codes or rewriting them from scratch for the new architectures. They’re also concerned about (but willing to deal with) the limitations and issues with programming and debugging tools for the accelerators.

Early adopters aren’t overly concerned about code portability, because in their view, efforts such as OpenCL and the development of standard libraries will eventually deliver on cross-platform portability.

Early adopters aren’t overly concerned about code portability, because in their view, efforts such as OpenCL and the development of standard libraries will eventually deliver on cross-platform portability.
The second user type is willing to experiment with accelerators, but is concerned about code portability and long-term sustainability. For this group, the availability of advanced compiler technology to obtain the acceleration benefit from reasonably portable code is key. Examples of such emerging tools include OpenMP-style compilers for Cell and GPUs. A 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 might not match the gains possible using native compilers and explicit parallelization, it preserves code portability. 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. The best performance improvements reported on the accelerator-based HPC systems so far are from efforts in which computational scientists stepped back and re-evaluated existing algorithms’ suitability for accelerator architectures, and then re-implemented them from scratch using native accelerator languages and tools.

The main reason computational scientists consider using accelerators is because of the need to increase application performance to either decrease the compute time, increase the size of the science problem that they can compute, or both. However, reasons other than pure performance improvements are starting to influence the deployment of HPC resources. As the size of conventional HPC systems increase, their space and power requirements and operational cost quickly outgrow the available resources and budgets. Thus, metrics such as flops per machine footprint, flops per watt of power, or flops per dollar spent on the hardware and its operation are becoming increasingly important. Accelerator-based HPC systems look particularly attractive considering these metrics. For example, the top eight systems on the November 2009 Green500 list of the world’s most energy-efficient supercomputers are accelerator-based.

Realizing the Potential

Although the potential of accelerators in HPC is evident for many (but not all) applications, it might remain unrealized unless the scientific computing community, computer scientists, technology vendors, and funding agencies do their part to advance the technology. Several existing challenges can point the way forward.

First, many current efforts to move scientific codes to accelerators are undertaken by the domain application developers themselves. Although domain experts’ application insights are hard to replicate, they often end up spending too much time on the idiosyncrasies of the new systems and programming tools and don’t always obtain the full benefits of the accelerator capabilities. Teaming up with computer scientists or application specialists intimately familiar with these systems can help produce better code and achieve better performance.

Second, application developers are bogged down in hardware-specific coding. Currently, programming application accelerators is harder than it should be. Existing tools for application performance analysis, such as code profilers, are unsuitable for accelerators and tools with similar capabilities for accelerators are only starting to emerge. Debugging tools don’t provide a sufficiently accurate inside view of the accelerators. Substantial investments should be made to develop tools that could enhance the productivity of application developers working with accelerators.

Third, many application developers are reluctant to start porting code to application accelerators because they can’t predict performance benefits without an actual prototype implementation, which by itself is a substantial effort. We therefore need to develop performance models against which developers can evaluate the suitability of candidate applications for application accelerators. We also need to create a set of guides on how to port applications to different accelerator platforms.

Finally, application accelerators aren’t currently treated as independent computing resources by the HPC management software or by the runtime systems used for parallel application execution (such as MPI runtime). System administrators and application developers must therefore provision and use these resources manually, imposing a limited range of usage scenarios and programming models supported on the accelerator HPC systems. We need to enhance existing runtime systems and HPC management software stacks.
to allow for more efficient and easier accelerator utilization and sharing among multiple users.

**In this Issue**

The articles in this special issue cover three different scientific domains (cosmology, atmospheric science, and computational chemistry) and two application accelerator platforms (Cell and GPU). The approaches described also cover all three types of accelerator users, demonstrating their thought processes and justifications for the selected approach. We hope readers will find these articles interesting and relevant to their domain, experience, and goals.

Adrian Pope, Salman Habib, Zarija Lukic, David Daniel, Patricia Fasel, and Katrin Heitmann, from the Los Alamos National Laboratory, and Nehal Desaiy, from the Aerospace Corporation, discuss the development of a new hybrid cosmology simulation code for RoadRunner, the world’s first PetaOp platform. The authors designed their application around the machine, taking into account the Roadrunner architecture’s unique characteristics. Their approach is driven by the need to reduce communication among the computational kernels running on the Cell-based accelerators. To achieve this, they use a mirrored particle-cache technique and simplify computational templates via digital filtering and differencing in the spectral domain, resulting in a simple spatial deposition of the simulated domain with minimal communication overhead. The resulting code’s weak scaling is nearly ideal up to 6,000 processors.

Rory Kelly from the US National Center for Atmospheric Research describes the use of GPU computing in atmospheric modeling. Working with the Community Atmosphere Model code base, Kelly ported to a GPU a subroutine that approximates the effects of shortwave radiation on the Earth's atmosphere. The GPU-accelerated subroutine alone is 14 to 20 times faster than the original code, depending on the CPU architecture it’s compared to. Kelly also analyzed the impact of accelerating this subroutine for the overall application and found that the speedup translated into only a 3 percent performance improvement for the full application. He concludes that a full atmospheric model to make truly efficient use of HPC systems with GPUs should be designed for a GPU from the ground up, rather than retrofitting an existing code.

Peter Eastman and Vijay Pande from Stanford University present OpenMM, a framework for molecular simulations. OpenMM was engineered as a layered architecture: the higher level provides a user-accessible platform-independent API, while the lower level provides hardware-specific implementation of the computational blocks. Thus, to enable GPU use, developers provide the lower-level layer implementation for a particular GPU platform, while the higher-level API remains unchanged. The current OpenMM implementation provides basic molecular dynamics subroutines for three platforms: a reference platform written in C++, a CUDA-based platform for Nvidia GPUs, and an OpenCL-based platform for a variety of GPUs and CPUs.

Mark Watson, Roberto Olivares-Amaya, Richard Edgar, and Alan Aspuru-Guzik, from Harvard University, and Tomás Arias, from Cornell University, provide results of accelerating correlated quantum chemistry calculations using GPUs. Their approach is based on replacing standard matrix multiplications routines ($\text{GEMM}$) from the BLAS library with the newly developed mixed-precision subroutines ($\text{MGEMM}$) to accelerate key matrix multiplications used in legacy-correlated quantum chemistry codes. The authors developed and evaluated two such mixed-precision algorithms: bitwise $\text{MGEMM}$ and heterogeneous $\text{MGEMM}$. The article provides implementation details as well as detailed benchmarking results for the two algorithms, showing their advantages versus single-precision matrix multiplication code provided by Nvidia.

**HPC** is experiencing a new cycle of innovation in which high-performance serial and parallel processing cores are tightly coupled with special-purpose hardware accelerators to enable unprecedented performance levels. Tightly integrated Intel Larrabee and Advanced Micro Devices (AMD) Fusion are just over the horizon, while loosely coupled multicore systems with many-core GPUs attached are being assembled into large HPC systems with impressive demonstrated capabilities for specific applications. The challenge now is to migrate many more applications to these systems as well as to develop new algorithms that can take full advantage of them while ensuring portability to new generations of platforms. The articles presented in this special issue shed some light on the use of application accelerators in HPC and the challenges the scientific computing community is facing.
Volodymyr Kindratenko is a senior research scientist at the US National Center for Supercomputing Applications and a lecturer in the Department of Electrical and Computer Engineering at the University of Illinois at Champaign-Urbana. His research interests include high-performance computing and special-purpose computing architectures. Kindratenko received a DSc in analytical chemistry from the University of Antwerp. He is a senior member of IEEE and the ACM. Contact him at kindr@ncsa.illinois.edu.

Robert Wilhelmson is chief scientist at the US National Center for Supercomputing Applications and professor of atmospheric sciences at the University of Illinois at Champaign-Urbana. His research interests include modeling severe storms and tornadoes, numerical methods, flow visualization, and high-performance computing. Wilhelmson has a PhD in computer science from the University of Illinois. Contact him at bw@ncsa.illinois.edu.

Robert Brunner is a research scientist at the US National Center for Supercomputing Applications and an associate professor of astronomy at the University of Illinois at Champaign-Urbana. His research interests include applying data mining and machine learning algorithms to large astrophysical data sets with the goal of better understanding the nature and evolution of our universe. Brunner has a PhD in astrophysics from the Johns Hopkins University. Contact him at bigdog@illinois.edu.

Todd J. Martinez is David Mulvane Ehrsam and Edward Curtis Franklin Professor of Chemistry at Stanford University and a professor of photon science at SLAC National Accelerator Laboratory. His research interests include quantum mechanical simulation methods for ground and excited electronic states of molecules. Martinez has a PhD in chemistry from University of California, Los Angeles. Contact him at toddmtz@stanford.edu.

Wen-mei Hwu is a professor and the Sanders-AMD Endowed Chair in the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign. His research interests include architecture, implementation, and programming tools for parallel computer systems. Hwu has a PhD in computer science from the University of California, Berkeley. He is a fellow of IEEE and the ACM. Contact him at w-hwu@illinois.edu.

Selected articles and columns from IEEE Computer Society publications are also available for free at http://ComputingNow.computer.org.