

Report on NSF Workshop on Center Scale Activities Related to Accelerators for Data Intensive Applications

31 October, 2010

Viktor K. Prasanna, University of Southern California

David A. Bader, Georgia Institute of Technology

This workshop is supported by NSF Grant Number 1051537, in response to the Call for Exploratory Workshop Proposals for Scientific Software Innovation Institutes (S2I2).

1. Executive Summary

This report is the outcome of the *Planning meeting for Center Scale Activities related to Accelerators for Data Intensive Applications*, held at Arlington, VA, on October 14-15, 2010, and supported by the National Science Foundation (NSF) as part of the Call for Exploratory Workshop Proposals on Scientific Software Innovation Institutes (S2I2) [NSF 10-050]. The workshop, chaired by Viktor K. Prasanna, University of Southern California, and David A. Bader, Georgia Institute of Technology, invited domain scientists from the application areas of biology, computational social science, and security, and computer science researchers from the technology areas of multi-core processors, general-purpose graphics processing units (GPGPU), and field programmable gate arrays (FPGA). The workshop included over 25 participants, with broad representation from academia and industry.

The goal of the workshop was to outline the requirements and make recommendations to the NSF as input for a future solicitation on Scientific Software Innovation Institutes (S2I2) [NSF 10-551]. The expected outcome was two-fold:

- To better understand the opportunities of multi-core, GPGPU, FPGA, and other accelerator technologies for the biology, computational social science and security application domains; and
- To use this improved understanding to provide recommendations to the NSF for a solicitation establishing center-scale institutes for sustainable software infrastructure to promote interdisciplinary research.

The first day of the workshop focused on outlining the opportunities from accelerator technologies in several broad application areas, through a format of presentations and facilitated discussion, and a presentation by Manish Parashar, NSF program manager, on the S2I2 program. The second day included four breakout sessions focusing on center organization, software infrastructure, community understanding, and sustainability. These discussions produced a number of concrete proposals that address both goals of the workshop.

This executive summary presents our main recommendations that address the above goals; the remaining sections of this report provide further supporting detail. All the participants provided inputs in preparing this report and also reviewed the report before submission to the NSF.

1.1. Recommendation on Center Organization, Goals and Sustainability

The S2I2 center's operational model must balance research, development, and integration activities to best achieve the overall goal of software creation, maintenance, and sustainability. The center is best organized through a strong management model headed by a center director with academic credentials and management experience. An S2I2 center should establish an external advisory board representing target communities, including industry, investigators from Scientific Software Elements (SSE) and Scientific Software Integration (SSI) projects [NSF 10-551], and notable scientists.

The S2I2 center management should identify and engage with relevant small-scale SSE's and medium-scale SSI's. Shared funding for aligned activities can provide an SSE/SSI incentive to collaborate. The center management should focus on a set of key projects and activities with high impact on the targeted communities, including software development, maintenance, user education, and training. Dedicated staff with software development expertise in accelerators needs to provide in-depth support for domain scientists, up to several months at a time, to facilitate their leveraging new accelerator technologies autonomously.

The primary goal of the center must be to investigate, build, distribute, and maintain software that will spur innovation in scientific computing over an extended period of time. A strong software design and development team should build and maintain reliable, usable, and extensible software that are created organically and contributed by user, developer, and business communities.

A sustainable software infrastructure requires building the "software right". The workshop participants recommend that a center should adopt portable, open, extensible software development frameworks and practices to allow easy intake, sharing, and sustained maintenance of code. Software should be modular, with stakeholders associated with each module. Growth of new software must be encouraged while balancing the need to maintain the existing codebase. The center must actively plan for the lifecycle of software, transitioning from management to maintenance and even to deprecation. Maintaining and evaluating an evolving set of benchmarks and "hero" codes is important to the community. There is also agreement that any center-maintained software must include and enforce guidelines for code contribution.

Documentation of both software and best practices for specific domains and applications are essential. Availability of trained technical writers in the center would be of immediate value to the community.

Choosing the "right software" to maintain in the center is important. The center should target science domains and applications that have impact in the near and long term (2-3 years). A balanced selection of high risk software applications that may advance sciences radically, as well as software with incremental, sustained improvements should be chosen. The center must provide avenues for computer science investigators to participate, to ensure rapid incorporation of advanced software tools.

The workshop participants consider education and training of external visitors as a major aspect of attaining sustainability as it can spread expertise at their home institution and beyond. Invitations to long-term visitors could be handled by a bidding mechanism, any form of peer review, or optionally in return for donating their application code. The center needs good instructors, both for large groups and for one-to-one consulting basis.

The center's lifespan should go through phases: an initial start-up phase to collect code and best practices, followed by delivery of major artifacts to the wider community, and then a push of knowledge outward through education and training while still curating the wider community's knowledge. Any center must prove value not only to the NSF but also to community stakeholders like vendors and users.

The workshop participants recommend the NSF to encourage specific letters of collaboration from the broadest possible community of scientific software researchers, developers, and domain scientists in support of an S2I2 proposal, but suggest they be limited to at most 10 letters.

The workshop participants advocate for a center model that includes a core group at a single site, and a limited number of partners at remote sites. Such a center requires significant coordination and effort, and splitting of the center into many small, distributed parts would be counter-productive. The participants do not find it necessary to co-locate the center with other major cyber-infrastructure entities.

1.2. *Motivation for a Center on Software for Accelerators*

High-end computing systems are moving to extreme scales by augmenting architectural designs with hardware-based application accelerators. The workshop participants identify reconfigurable hardware such as FPGAs, heterogeneous processors like the upcoming Intel Knights family and AMD Fusion platforms, highly multi-core and multi-threaded processors from Cray and Oracle/Sun, and general-purpose graphics processing units (GPGPUs) from NVIDIA and AMD, among others, as exemplars of these accelerators. Current software repositories and tools do not address development for these systems sufficiently for productive scientific use because the application space is diverse, leading to an immense performance optimization design space.

Data-intensive computational kernels in biology, network security, and social sciences exemplify emerging and future workloads and represent challenging cases that can benefit from high performance accelerators. For example, next generation DNA sequencing instruments flood individual investigators with terabytes and community projects with petabytes of data. High-throughput instrumentation is also making a presence in other areas, such as materials science where atomic-scale images are provided by Atom Problem Tomography (APT). The workshop participants see a combination of multi-core processors, and accelerator technologies such as GPUs and FPGAs as providing a more affordable and accessible alternative to large clusters and supercomputers. Another application for accelerators is the area of high throughput malware scanning, matching large data quickly against sophisticated malware signatures. Compute-intensive numerical methods, such as the molecular dynamics and Fast Multi-pole Methods, can also benefit from accelerator-driven parallelization.

Working through practical details of how applications can take advantage of such accelerators will enable breakthroughs in these domains. An iterative development process is required to understand not only the features and capabilities, but also the limitations and idiosyncrasies of new technologies useful in solving science problems efficiently. The workshop participants see a vital need and present opportunity to develop a community-based collaborative software infrastructure that aims to accelerate the design, implementation, optimization, and dissemination of a wide variety of codes tuned for accelerator-based systems, as well as methodologies, education and training materials.

1.3. Mission Statement for an S2I2 Center

Following an engaging discussion among the workshop participants, a mission statement to capture the goals of the S2I2 center was proposed:

“The S2I2 program envisions that the Scientific Software Innovation Institutes will provide the expertise, processes, resources and implementation mechanism to transform computational science and engineering innovations and community software into a robust and sustained software infrastructure for enabling accelerator-enhanced science and engineering”

2. Motivation

Knowledge has often been forgotten during the course of recorded history. The historical record is quite clear that important knowledge is often “rediscovered” as part of the human experience.

The modern computer era has been a double-edged sword in regard to this subject. On one hand, data can now be stored for long periods of time and reproduced on demand. On the other hand, creators of knowledge beyond the bleeding edge, in particular academia, will discover and innovate in some area and then move on. Being beyond the bleeding edge, the lone researcher (or small research group) finds it hard to provide an infrastructure that can sustain innovation beyond that edge for very long.

Acceleration technology has long suffered from losing cutting-edge research knowledge. In the 1970's commercial general purpose computers had very low floating point performance because accounting was the main application in use at that time. Universities as well as the military had a need for high performance floating point mathematics to solve vector and matrix calculations. As a result, SIMD, MIMD, systolic, and vector machines were conceived and high level languages were designed to program them. Many roadblocks were overcome to learn to program these exotic machines. Ironically the winning technology was a CISC/RISC single core von-Neumann architecture called the microprocessor. In the process of choosing a winner, knowledge from many of the hard won programming battles was lost in the 1980's. We must now relearn how to overcome the roadblocks of the past as well as address the current roadblocks created by the high performance computing requirements of present day science. We must also engage researchers who recall past successes and failures.

This recurrent problem can be addressed by creating a framework that will allow researchers to express their innovations in a reusable, sustainable fashion. By creating centers staffed by personnel capable of “institutionalizing” the researchers’ innovations, the academic community can leverage and keep safe and useful the hard earned knowledge that will make our nation stronger.

One problem has been that visionary research discovers algorithms proven correct but infeasible for existing computational infrastructure.

2.1. Accelerator Community

Accelerators (as defined by a non-general purpose CPU and general register architecture) have demonstrated that they will be of continuing importance to scientific computing because of their greater energy efficiency and performance – sometimes captured in a figure of merit energy-delay product (EDP). As it is widely accepted that mainstream computing technologies based on CMOS will be increasingly energy/power limited [DARPA-Exascale], the shift to greater reliance on accelerator computing structures is likely to continue and perhaps accelerate. As such, accelerators must be a significant element of the S2I2 program if it is to have a broad impact on scientific software.

3. Technologies

3.1. Field Programmable Gate Array (FPGA)

Field Programmable Gate Arrays (FPGAs), invented in the mid 1980s, were predominantly used for connecting two or more different logical subsystems. Researchers have taken advantage of the parallel architecture of FPGAs and have been successful in accelerating certain applications. Such research is often forgotten, as discussed in Section 2. In recent years, there has been a significant increase in the size of the FPGAs. FPGA architectures containing multi-million, and occasionally over a billion, transistors have been manufactured by vendors. The large size and architectural enhancements, including computing domain-specific feature sets, provide the needed versatility to accelerate a wide range of applications on modern FPGAs.

FPGA architecture typically consists of an array of logic blocks, IO pads, routing channels, DSP blocks and internal RAM. Depending on the application, these architectural elements are programmed to perform various functions. The logic and DSP blocks can implement arithmetic and logic operations such as add, multiply, and compare. With modern FPGA capacities, hundreds of thousands of these logic blocks can accelerate science applications with wide parallelism. The logic and DSP blocks' results flow through the FPGAs using configurable routing channels. The ability to create a massive array of operators coupled with the capability to route the data between operators make FPGAs an excellent platform for data flow engines that accelerate applications.

Configuring data flow between operators explicitly removes overheads, like cache misses, required in general purpose processors. These operators can be configured to have point-to-point dedicated interconnects, thereby efficiently pipelining the execution of operators. The parallelism offered by current generation FPGA architecture can be easily seen in a few examples of HPC-relevant parameters [HPC-FPGA]:

- Internal bandwidth to move the operands and results of application-specific ALUs is in the order of terabytes/sec (TB/s)
- Throughput of integer operations is in the order of Tera-operations/sec (TOPS)
- Throughput of floating point operations is in the order of gigaflops/sec (GFLOPS)

Several FPGA-based accelerator and server solutions are available from HPC system vendors [FPGA-Vendors]. These server companies also provide the compiler tools, libraries and services to program these systems. The science community can tap into the benefits of the FPGAs by accelerating their application on these FPGA-based HPC systems, but will need to restructure their algorithms to achieve the potential gains [Storaasli2007, Fu2009, Shafer2010, Convey, Baker2007, Yi2010, Putnam2009, Ganegedara2010, Hua2008].

3.2. Graphics Processing Units (GPU)

GPU stands for Graphics Processing Unit. Traditionally, GPUs have acted as co-processors within a computer system providing acceleration of 3D graphics rendering. However, in recent years there has been significant interest in exploiting the enhanced memory bandwidth and raw floating point horsepower of GPUs for general purpose computation (GPGPU). GPUs offer extremely high FLOPS rates at relatively low cost and power consumption. This makes them attractive targets for hardware acceleration of many scientific applications including molecular dynamics (MD) simulations. As high-end GPUs increasingly become standard equipment in scientific workstations, many research labs begin to have them and they can be purchased easily with new equipment. This makes GPUs accessible to researchers and thus tempting instruments for running computational experimentations.

The nature of GPU hardware has made its use in general purpose computing challenging [yang2007]. However, the development of application programming interfaces (APIs) targeted at general purpose scientific computing has reduced this complexity to the point where GPUs are starting to be accepted as serious tools for the economically efficient acceleration of an extensive range of scientific problems. Recently, vendors are providing platform-specific programming toolkits for general purpose computations, such as NVIDIA's Compute Unified Device Architecture (CUDA) [CUDA] and AMD's ATI Stream [ATI-Stream].

However, algorithmic paradigm shifts are often required in existing code to maximize the performance offered by the massively parallel GPU hardware. A key strategy in improving wall clock time to scientific problem solution is to recast an algorithm in a way that makes it computationally palatable for the nature of the hardware that it is being executed on. An algorithm that performs poorly on a CPU may perform many orders of magnitude better on a GPU and vice versa. However, when dealing with scientific problems, it is essential that alternative approaches to solving the underlying physics yield the same solution, albeit via different paths. Changing the nature of the problem being solved to suit GPU acceleration, without a thorough understanding of the implications this has on the scientific results, can have serious consequences.

Despite being more difficult to optimize for than CPUs, GPGPUs are being used for a variety of data-parallel computations, such as sorting, FFTs, wavelets, and dense matrix multiply; sparse conjugate gradient and multi-grid; bioinformatics; relational joins for databases; quantum chemistry; and the fast multi-pole method [CUDA-Community]. However, productive and efficient use of these highly-capable GPGPUs requires improved development environments that expose better programming models, handle streaming data movement, and reduce scheduling overheads [Jeon2010].

3.3. Multi-core & Parallel

For the purposes of this report, we assume that multi-core processors are composed of cores that are capable of executing not only independent streams of instructions but independent operating systems. This distinguishes them from other many-core accelerators such as DSPs or GPUs, although these devices may be converging as GPUs become more programmable and multi-cores add more cores. Since 2004, microprocessor companies have been adding cores rather than increasing the performance and clock speed of single cores or uniprocessors. This is true for both general-purpose host microprocessors for workstations, servers, and laptops, and for embedded and accelerator processors. High-end microprocessors have now reached eight to sixteen cores for workstations and servers [Shin2010, Wendel2010, Rusu2010], and are up to 64 cores for embedded accelerator processors [Bell2008], which tend to be more power-efficient and include specialized instructions for accelerating software for specific domains.

As the number of cores increases, software development becomes more challenging, especially for applications for which performance is critical. Interconnects, topology, and locality become more complex as the number of cores increase beyond what can be connected via a single bus. The memory hierarchy becomes deeper as memory associated with remote cores is accessed and techniques are used to avoid consuming limited off-chip memory bandwidth. A variety of programming models that expose parallelism must be considered, and the ability of compilers and libraries to exploit parallelism will affect software innovation significantly. Reliability becomes a more significant issue as the number of components comprising a device continues to increase and as software that exploits parallelism becomes more complex. Debugging also becomes more challenging as parallelism is added, and programming models and debugging may need to be considered together. As Moore's Law delivers more cores per processor, sustained scalability will be important for software, to exploit not only the number of cores available at a given time but over a period of time as new processors with increased number of cores are released.

Multi-core processors fill an important niche in scientific processing because of their flexibility, programmability, and ability to execute control-intensive algorithms efficiently [Yoo2009, Datta2008, Miller2010]. However, their ability to support continued software innovation is endangered if the software community does not find sustainable ways to support the exploitation of parallelism that will be provided over the next decade by multi-core processors.

3.4. Multi-threading

Multi-threaded acceleration focuses on memory bottlenecks. These accelerators tolerate latency from accessing unpredictably scattered data by maintaining many execution threads. An execution thread issues computational instructions once all outstanding memory references are satisfied. With a deep enough queue of memory references and enough threads available, a processor can keep all its execution units busy and achieve astounding efficiency. Another feature sometimes included in multi-threaded acceleration is a fine-grained synchronization facility.

The common-case cost of fine-grained locking and atomic operations on massive data can be amortized into the memory latency and tolerated with the same mechanisms

One example of this architecture is the Cray XMT [XMT]. By tolerating latencies all the way to its distributed memory units, the Cray XMT achieves remarkable efficiency on massive graph analysis. Fine-grained locking is “free” because it is amortized into the memory latency, and this locking permits automatic optimization and software composition otherwise impossible. Simple software often achieves high efficiency on the Cray XMT's multi-threaded architecture.

A less-successful version of multi-threaded acceleration is within Intel's Hyper-Threading [HT]. Hyper-Threading only tolerates the latencies to the outermost cache level and includes only simple atomic operations to support synchronization. Taking advantage of Hyper-Threading requires great care in software engineering, showing how seemingly minor variations in the low-level acceleration architecture can affect the software architecture.

4. Applications Areas & Communities

In our scope for this workshop, the organizers selected three broad applications areas as representatives of emerging science disciplines that could take ready advantage of accelerator technologies and potentially see high-payoff scientific rewards from this new opportunity. This report describes in detail areas from biological sciences, security, and computational social sciences, as exemplars of emerging computational science areas with synergistic needs for the use of accelerators.

4.1. Biological Sciences

High-throughput instrumentation is rapidly advancing the rate and scale of data generation in modern biological research. This in turn is fundamentally altering the research landscape by enabling novel applications and driving cost-efficiencies. Advances in DNA sequencing have caused sequence reads to dramatically increase from 96 reads per run in 2006 by the ABI 3700 sequencer, to the present day 300 million paired reads per run by the Illumina Genome Analyzer. This quantum leap in throughput creates many technical challenges:

- Instrumentation-specific challenges such as error correction;
- Challenges to reinvent methodologies for classic applications – e.g., assembling large mammalian and plant genomes with next-generation sequencing is still an open challenge;
- Challenges due to rapid scale up in scope and number of research projects driven by lower economic costs; and
- Challenges caused by an expanding repertoire of applications – e.g., personalized genome sequencing, and digital expression analysis for systems biology.

The rapid development of next generation sequencing and the richness and diversity of applications it makes feasible have created an enormous gulf between the potential of this technology and the development of computational methods to realize this potential. This creates unprecedented opportunities for research with the possibility for broad impact. Furthermore, a number of competing

technologies for next-generation sequencing, each with its own and distinct technical characteristics, add another dimension to the set of research challenges. Finally, other high-throughput instrumentations beyond next-generation sequencing, including technologies for expression and metabolomic measurements, are also commonly used in modern biology research.

High-throughput instrumentation is shifting individual investigators to the data-intensive applications space (terabytes) and community projects to the extreme-scale (petabytes). Not long ago, community projects were terascale, and it was sufficient to use large clusters and supercomputers. However, affordability and accessibility rule them out for widespread use by present day individual investigators, while they are proving inadequate for extreme-scale projects. These problems can be effectively addressed by a combination of emerging computing technologies such as multi-core processors, and accelerator technologies such as GPUs and FPGAs. As a concrete example, development of high-throughput computational biology tools for an inexpensive workstation with a few accelerators, with the intent of widespread adoption, can significantly impact research in the biological sciences.

4.2. Security

Accelerator technology can enhance performance in many areas of computer security. Here, we briefly outline several of those areas and explain how the processing requirements of the domain benefit from acceleration. The potential security domains for acceleration have expanded rapidly both as scientists figure out how to map problems onto accelerators, and due to the expanding capability of accelerators. In particular, recent trends to improve I/O performance of accelerators have opened several new applications from the security domain.

Before discussing the use of accelerators to help computer security, we first note that accelerators can themselves introduce potential security problems into systems. While to date accelerator based malware has been limited, this possibility exists in the future. Possible attack surfaces include:

- Using the accelerator instructions to execute malware via pre-compiled exploits. A likely initial threat is the use of accelerators to help obfuscate CPU-based malware.
- Attacking accelerator code to gain execution. A particular concern here includes the large scale parallelism of accelerators and the likelihood of latent synchronization errors in existing accelerator code.
- Using the just-in-time translation features of environments such as CUDA to target malware to multiple accelerator systems.

An integral part of a center devoted to accelerators will be to help ensure that accelerators remain secure and do not become the preferred method to attack their host systems.

One security application for accelerators is malware analysis. The process of understanding malware requires reverse engineering and de-compiling code that has often been encrypted and obfuscated. Binary analysis requires algorithms that recover both control flow and data flow from programs. Traditionally, these techniques use algorithms that are not well suited for accelerators (especially GPUs) due to their relatively low computational intensity and focus on integer operations. However, recent

results [Hall2010] have demonstrated that some types of program analysis that are computational intensive ($O(N^3)$) can be effectively performed as sparse matrix floating point operations, and see gains of up to 70x on GPUs as compared to well-tuned CPU-based algorithms. While this initial work is in the area of traditional compilation, there is reason to expect that it will translate to similar gains for reverse engineering.

Another security application for accelerators is for high throughput malware scanning. Malware scanning requires rapid processing of large amounts of data (suspect binaries) and the ability to quickly match the data against signatures of known malware. To achieve high performance on conventional processors, malware companies resort to using very simple patterns. Typical patterns often include matching against a relatively small number of bytes. Using accelerator technology will allow using more sophisticated patterns. This is one application area that depends on significant I/O performance from accelerators since even sophisticated pattern matching will likely retain a relatively high ratio of I/O to computation.

Algorithms that monitor network data for malware are very computationally intensive. Due to the intensive nature of these algorithms, an application must limit the depth of the scan to meet the real time requirements of the network. This limited depth scan provides an opportunity for viruses and other forms of malware to escape detection. An FPGA based accelerator will provide an ideal solution for network monitoring, potentially allowing the full depth of scan required to ensure malware free networks. FPGAs will allow hardware based scanning algorithms to be updated as new malware signatures are detected through FPGA reconfiguration capabilities.

4.3. Computational Social Sciences

The growth of data, both structured and unstructured, outpaces current analysis frameworks' capacities. Every month, Facebook adds 25 billion connections between users and content, forming a wealth of information for marketing and advertising. Every day, The New York Stock Exchange processes 1.5 TB of new data into an 8PB long-term archive that could potentially come under regulatory scrutiny. Both network security and scientific instrument monitoring tools produce gigabytes to terabytes of data daily. Widely available tools for analyzing the graph structure within these datasets focus on static analysis, but users require dynamic tracking of properties for business optimization, anomaly detection, and other applications. A single QPI memory channel at a (unidirectional) 12.8 GB/s can transfer only 1.1 PB of data per day, and the overhead of carefully distributing fine-grained parallelism within the analysis kernels often reduces this amount significantly. The current practice of repeatedly applying static analysis tools does not scale even to current data sizes and forgoes many opportunities for parallel performance.

The modeling and analysis of massive, dynamically evolving semantic networks also raises new and challenging research problems to respond quickly to complex queries. Empirical studies on real-world systems such as the Internet, socio-economic interactions, and biological networks have revealed that they exhibit common structural features – a low graph diameter, skewed vertex degree distribution, self-similarity, and dense sub-graphs. Analogous to the small-world (short paths) phenomenon, these real-world datasets are broadly referred to and modeled as small-world networks. Several types of computations may become feasible at large scale using accelerator technology.

One of the fundamental problems in network analysis is to determine the importance (or the centrality) of a given entity in a network. Some of the well-known metrics for computing centrality are *closeness*, *stress* and *betweenness*. Of these indices, betweenness has been extensively used in recent years for the analysis of social-interaction networks, as well as other large-scale complex networks. Some applications include lethality in biological networks, study of sexual networks and AIDS, organizational behavior, supply chain management processes, as well as identifying key actors in terrorist networks. Betweenness is also used as the primary routine in accepted social network analysis (SNA) algorithms for clustering and community identification in real-world networks.

Path-based queries: Several common analysis queries can be naturally formulated as path-based problems. For instance, while analyzing a collaboration network, we might be interested in chains of publications or patents relating the work of two researchers. In a social network, relationship attributes are frequently encapsulated in the edge type, and we may want to discover paths formed by the composition of specific relationships (for instance, *subordinate of* and *friend of*).

These types of queries on massive modern datasets may be accelerated using new technologies for tolerating latency to memory, explicitly managing the memory hierarchy, and reducing overheads for fine-grained synchronization within irregular data structures.

4.4. Other, Compelling Applications

4.4.1. Deep Packet Inspection

In networking, many new applications heavily depend on data intensive computation. Examples include real-time traffic statistics, content caching, targeted advertising, and user behavioral analyses. All these applications' computations have one common component – Deep Packet Inspection, or DPI, which was originally introduced into the networking world for network security. DPI uses pattern matching at its core, and this introduces several new challenges for pattern matching.

Control information is frequently expressed in a database that can grow large in size. Complexity of pattern matching rules tends to increase for DPI applications [Yu2006]. Many of these rules are across multiple packets, multiple encapsulated layers of protocols, and multiple flows. Increasingly, rules depend on statistical information and topology.

Next, these computing tasks traverse the database at a speed that is an order of magnitude higher than the traffic data. This is partially because of the large size of the database memory footprint. The other reason is that intelligence is gradually added into high speed devices, which traditionally ran on layer four and below.

Lastly, automation of pattern matching engine generation and a higher level of abstraction are often desired. Viruses change fast not only in terms of signatures but also in terms of behavior. The number of applications and protocols, especially private protocols, increases rapidly [Bonfiglio2007]. To keep up with the pace of the evolution of applications and protocols, both the DPI engine generation and the DPI rules have to be expressed at a higher level of abstraction [Yang2008].

All these challenges necessitate research on both algorithms and hardware acceleration in this area [Le2010, Hua2010a, Hua2010b]. An infrastructure providing a research result repository can constructively help people to leverage work from others. A researcher tracking system built on top of the repository could potentially provide benefit to the industry and make the S2I2 infrastructure sustainable.

4.4.2. Material Sciences

High-throughput instrumentation and the concomitant large-scale data-driven discovery needs are also emerging in other areas of science and engineering. For example, the atom probe tomography (APT) instrument in materials science enables atomic-scale images of materials, and is leading to billion atom datasets. Compute-intensive numerical methods such as molecular dynamics and Fast Multipole Method cater to a diverse set of applications spanning multiple fields – and can benefit from accelerator-driven parallelization.

5. Center Models and Management, Organizational Structure, Governance

Any activity undertaken by the NSF and OCI to support scientific software as a computational infrastructure will have broad importance for the science community. So it is critical that the decisions of what scientific areas, particular application codes, and computational platforms to invest in be made with vigorous, broad community input and advice from senior leadership of the community. Thus, we recommend that the S2I2 programs require that the majority of the investment (say 75%) be prioritized by decisions made by a diverse advisory committee with a well-defined open input process from the broad scientific computing community.

However, accelerators significantly complicate the picture when considering how to select, solidify, and support scientific software. The reason for this is that today accelerators vary widely in hardware structure, software interface, and some like FPGAs present an amorphous structure to software (see earlier report section). As such, it is critical that an S2I2 center adopt a rigorous management approach to selecting which hardware accelerators will be explored and which will be supported in scientific software packages that the center supports. For example, we expect that assessments of number of platforms deployed (installed base) and future projections, expected architectural and programming model continuity of platforms, as well as support difficulty/cost would all factor into center decisions of whether or not to support an accelerator platform. In fact, it is surely the case that the supported platforms will be limited, and the specific supported platforms will change over time.

5.1. Center and Management Model

The workshop participants strongly recommend the establishment of a substantially large group at a single site for the S2I2 center structure. Further, personnel from remote sites can also be included as center members, but splitting the center into many small distributed parts must be avoided to achieve the S2I2 objectives. The core group should include a full-time center director, researchers, graduate students, software developers, and supporting staff. If applicable, the director position can alternatively be held by a (preferably tenured) part-time faculty member with a full-time deputy.

The S2I2 operational model, executed by the center director, must balance research, development, and integration activities to best achieve the overall goal of software creation, maintenance, and sustainability. That is why a strong management model, which is also able to ensure the coordination of sites at different locations, is strongly preferred by the workshop participants.

To meet the requirements of the position, an S2I2 center director must have academic credentials and have demonstrated management experience. Among his/her duties is the facilitation of coordination and interaction with federal agencies, industry, and other national labs and research centers. Also, the director is required to leverage the primary investment received from the NSF. The ability to raise funds from other sources is expected in accordance with NSF guidelines, in order to achieve sustainability beyond the NSF funding time frame.

In order to leverage a common identity both internally and externally, the S2I2 center's core group needs to be co-located at a single site with dedicated office space and other resources. Co-location with other major cyber-infrastructure entities such as a national lab is deemed unnecessary.

An S2I2 must report to an external advisory board. The board members should represent the center's target communities, including industry, SSE's and SSI's, as well as notable scientists.

The center management should identify and engage with relevant SSE's and SSI's. A likely incentive for an SSE/SSI to collaborate with the S2I2 is to acquire new shared funding in joint partnerships for aligned activities.

The center management is expected to focus on a small number of key projects and activities with high impact on the targeted communities. These activities not only include software development or maintenance, but also user education and training. The two latter efforts, aimed at sustainability, may include summer institutes. Detailed recommendations on education and training can be found in Section 9. Moreover, dedicated staff with expertise in software development for accelerators should provide in-depth user support for domain scientists and other users over longer periods of time, up to several months. Such support will facilitate application domain scientists' leveraging promising new accelerator technologies, eventually autonomously. Mechanisms to ensure high scientific impact of in-depth long-term user support of specific users or projects need to be established.

5.2. Governance

The successful execution of the S2I2 and the interaction with other entities should be measured by suitable evaluation metrics. In a similar manner, the performance with respect to the identification and curation of important software artifacts should be monitored. Possible aspects to which the metrics are applied include breakthrough scientific results, publications, citations, and programmer productivity.

We recommend the NSF to encourage specific letters of collaboration from the broadest possible community of scientific software researchers, developers, and domain scientists in support of an S2I2 proposal. However, we suggest the number of letters in the supplementary documents be limited to 10.

6. Center Scope and Goals

6.1. Goals

The primary goal of the center is to investigate, build, distribute, and maintain software that will spur innovation in scientific computing over an extended period of time. The center should stay abreast of current and emerging technologies like multi-core, GPU and FPGA accelerators for supporting and rapidly advancing science, and provide guidance to the scientific user community. A strong software design and development team should build and maintain reliable, usable, and extensible software, organically and from user, developer, and business communities. The center should be a knowledge repository that provides tutorials and training on new software and technologies supported by the center, offers consulting services to help scientists use the available software including long-term engagement, and shares best practices that may even be standardized.

6.2. Software Framework and Repository

A sustainable software infrastructure requires building the “software right”. The center should adopt portable, open, extensible software development frameworks and practices to allow easy intake, sharing and long-term maintenance of code. These promote small businesses and research developers to contribute both open and proprietary software to the center, and build on existing work. Platforms like Eclipse and OpenFabric prevailing in the open source can be reused for this.

A software repository should form the center-piece. Software should be organized in a modular fashion with stakeholders identified for each module to ensure long-term maintenance and use. Center staff with strong software development skills and domain knowledge should screen community contributions to ensure that development practices are met, and the contributions help support the targeted scientific research. Code contribution history and bug tracking features are necessary.

The center should encourage growth of new software while balancing the need to maintain the previous codebase in a manner accessible to scientists and students. Users in specific domains (e.g. genomics, social networks) or of specific technologies (e.g. FPGA, GPU) should be able to download all related software packages to get started with minimal overhead. A framework for visual composition of scientific applications, that allows different domain modules to be pipelined or technology modules to be swapped in/out, is needed. Scientific workflows provide some of these features.

The center should plan for the lifecycle of software. As technology matures and critical mass in a user community is achieved, active development of a software module may transition to one of management, maintenance and even deprecation. Approaches such as packaging hardware accelerators like GPUs, FGPA and multi-core machines along with requisite software platform into appliances can also be useful. This is already happening in next generation sequencing machines. Testing and benchmarking of software is also necessary to ensure verified code with known performance characteristics is present.

Documentation of both software and best practices is essential. Software documentation should assist with both use and further development of the software. The center should identify software,

technology, techniques and algorithms that work best for specific domains and applications. Design patterns and usage guidelines for advanced tools should be available, with the intent of helping users select the right set of software modules for their application and of promoting better development of their applications using advanced technologies. They should provide scientists with sufficient information on the tradeoff between the effort required to incorporate a new software or programming model against the potential gains for their application. Best practices and case studies should be recorded with contributions from center staff, researchers, and industry partners, possibly using a wiki-style system. Viability of new technology for specific scientific applications should be discussed. Knowledge gained and lessons learnt as part of on-boarding new tools should be recorded as they outlive individual techniques. The center should provide an open forum for discussion between the various stakeholders, and help evangelize and encourage community participation and contribution.

6.3. *Scoping the Communities*

Choosing the “right software” to maintain in the center is important. The scope of applications that the center assimilates or builds should be useful in advancing the sciences. The center should target science domains and applications that have impact in the near and long term (2-3 years). The success of the center depends on buy-in from the domain scientists into the software provided. This requires the scientists to be shown the advantages of the new tools and technology. Working closely with a few domain users on software and applications that show immediate results is useful to gain trust, and help start engagement with long-term benefits.

A balanced selection of high risk software applications that can potentially advance sciences radically, as well as software with incremental, sustained improvements should be chosen. Engaging with scientists who are making swift advances in software tools in their domain and helping them build those in the center’s software framework will help push the domain forward. At the same time, working with the broader, underserved community to propagate software tools maintained by the center is necessary. Domains like biology and social networks, which have limited legacy code and many low hanging opportunities, are well suited for a center’s focus.

Both cross-cutting software packages like graph algorithms that can be used across domains, as well as software that integrates vertically for a domain have to be developed. Working with multiple domains can help identify common algorithms and packages in otherwise unrelated fields. Using a uniform software framework and repository will make this transition between communities easier.

The center should provide avenues for computer science investigators to participate to ensure rapid incorporation of advanced software tools. Incentives for such researchers include access to the center’s pool of domain users who can help guide the research direction and increase the adoption and impact of their software.

7. Resources

An NSF center focused on accelerator research will need both hardware and software resources. While the need for hardware resources is obvious, the software resource component for accelerators is larger

than typically used for software application development. Each accelerator, be it a multi-core processor, GPU, or FPGA, needs a special purpose compiler to create the final bit-stream that is loaded on to the device. These compilers are typically supplied by the accelerator manufacturer at a reasonable price. The more expensive development tools used in accelerator development are the specialized tools for finding and exploiting parallelism, pipelining, threading, and/or vectorizing opportunities within the application. These tools can be relatively expensive and quite valuable for programmer productivity and optimal code design.

Starting with multi-core processors there is a long list of compilers that can be used to make binaries for those devices. Currently there are over five different C/C++/Fortran compilers that could be used: PGI, PathScale, Intel, GNU, and vendor supplied compilers like Cray's CCE compiler. In addition, there are companies working on next generation compiler languages and their compilers, like IBM's X10 and Cray's Chapel. Each of these compilers will have their own strengths and weaknesses, and the accelerator researcher will need to find which compiler works best for their application.

Coupled with the multi-core compilers are the libraries needed for optimal math operations like those in scientific libraries. These math libraries can be free, like the Goto library or cost money, like the NAG library. The developer will also need libraries to move data between the cores/accelerators, either PGAS mechanisms like UPC, Co-Array Fortran, Global arrays, or explicit data passing mechanisms like MPI, SHMEM, and Pthreads. Finally there are a host of tools used to debug non-working codes like Totalview, DDT, gdb, and tools to measure application performance and improve optimizations.

GPU devices are coupled with a general purpose processor, so all the development tools previously mentioned are also needed for GPU accelerated code development. An application using GPUs will need to be partitioned into separate CPU and GPU parts, and the GPU portion compiled with a GPU specific compiler. That compiler will come from the GPU manufacturer. The CPU portion is then compiled normally and the resulting bit-streams are joined into one executable.

Since FPGAs are used for making special purpose devices that can be used in consumer products, there are many expensive tools for making optimal designs. Tools like ModelSim, Symplicity, Simulink, MentorGraphs, Impulse-C, and many others, are used for laying out a circuit in the FPGA for optimal performance. These tools tend to be an order of magnitude more expensive than software compilers for multi-core and GPU devices. The FPGA will also need the manufacturer's compiler to compile the final bit-stream that is loaded on the device. More advanced tools will also simulate the CPU and the FPGA code running together for more opportunities to debug the logic.

Finally, in addition to the large suite of software development tools, the researchers will need the accelerator devices themselves. A minimal requirement is to have a single multi-socketed system with multi-core, GPUs and FPGAs attached to the research center. Better yet, one accelerated system will be provided to each researcher. In an ideal scenario, one large multi-node system will be available with each node having multi-cores, GPUs and FPGAs. This way, each researcher will have their own node for accelerator research, and can also experiment with running an application across multiple accelerated nodes. There will also be cases where the accelerator may not be attached to a CPU, but to a separate

special purpose system, like Cray's multi-threaded XMT. In this case a separate system will be needed for multi-threaded research.

8. Collaboration and Education

The workshop participants strongly feel that the fundamental requirements for sustaining a center-scale software effort are to ensure that the general community understands and uses the software developed and integrated by the center, as well as to encourage contributions to such software. Broadly the workshop participants recommend three levels of collaboration and outreach activities: (a) collaborations with other major organizations that are involved in similar efforts, (b) involvement of external contributors and users through visitor programs and (c) education and training of users.

8.1. Domestic and International Collaboration

Cooperation and collaboration with other organizations involved in related and complementary efforts is valuable to the success of any center. This could be a combination of domestic collaborations with institutions funded by other agencies as well as international collaborations. International scientists might often be necessary because no single entity or country can afford a particularly expensive project or investment. Yet, S2I2 projects should not be outsourced to foreign countries. If security concerns regarding international collaborations arise, they should be handled appropriately, for example, by intellectual property or non-disclosure agreements.

8.2. External Visitors

Educating and training external visitors is seen as a major aspect of attaining sustainability. The workshop participants envision a number of different measures, ranging from personal consulting over a longer period of time (on the order of months), to medium-sized courses that take a week, to short courses and tutorials. The rationale of personal consulting is to train domain scientists on how to optimize their code for different accelerators. During their stay at the center, the guests should be expected to learn code optimization on different accelerator technologies, and subsequently share this knowledge with other members of the community, especially at their respective home institute.

The S2I2 center should aim to establish a meaningful mechanism for selecting guests to receive a substantial amount of consulting. As an example, such invitations or scholarships could be handled by a bidding mechanism or any form of peer review mechanism. The workshop attendees feel that it would also be beneficial to request the guests to contribute the application code developed during the visit to the center, though in some cases this might not be possible (such as for export controlled software).

The workshop attendees also feel that a physical center with a core group might attract more people looking for guidance as opposed to small scattered entities. The location of this physical center might also have an impact on the number of people and length of visits the center can attract.

8.3. Education and Training

The S2I2 centers are expected to be places of expertise that can train groups of people in various parallel computing topics, with an emphasis on accelerators. While many universities educate their students in specific technologies for parallel computing, only a few of them teach a general parallel approach to algorithmics and software development. The workshop participants believe that an S2I2 center would be an ideal place to take on this complementary effort. Accordingly, the center staff needs to be trained as instructors, both for large groups and for a one-to-one consulting basis. The workshop participants also feel that in order to attract talented researchers to the center, the teaching/consulting workload might need to be limited appropriately, such as, for example, to 50% of their time.

Moreover, the center should educate other developers and domain scientists by defining and distributing principles and best practices for programming accelerators.

9. Sustainability

There are two aspects to sustainability,

- sustainability of software utilizing accelerators, and
- sustainability of a center supporting that software.

Participants recognize that sustaining the software will require long-term efforts from personnel, and that requires some sustainable group providing those personnel. We first address issues identified around sustainable software and then issues on a sustainable center.

9.1. Ensuring Sustainable Software

9.1.1. Visibility of Existing Software and Solutions

One widely agreed aspect of sustainability is clearly documenting the state of the art. Regardless of any other artifacts produced, the participants agree that encapsulating, transferring, and sustaining the existing knowledge on accelerators is a crucial and unserved need.

Currently, students, researchers, and developers must spend significant effort tracking down existing methods and tools for using accelerators. Accelerators in some form have existed since computing began. The modern accelerators have their roots in 20- to 30-year-old technologies (vector processing, FPGAs). Literature and implementation searches often require contacting the original authors, who may not have access to their old work or even still remain in the same field. This drastically increases start-up cost and raises barriers to using accelerators even when acceleration could open doors to new science results. Scientists should not need to justify funding large quantities of already accomplished computing research to produce new scientific results.

Collecting and maintaining codes, development tools, references, and technical reports would help sustain accelerator users and their codes. Such a collection also draws in *new* users by lowering the

barrier to entry. Similar existing efforts include MGnet¹ for the multi-grid community, the NEOS optimization server², Optimization Online³ and LAPACK with the LAPACK Working Notes⁴.

For a collection to have easily accessible value, the collection needs to be curated and organized by a stakeholder. We will revisit this under sustainability of a center.

Another aspect of visibility is community participation. Many accelerator vendors have large user's groups. For example, NVIDIA hosts a GPU Technology Conference⁵ often cited as a refereed, oral presentation venue. Cray users have formed the Cray User's Group⁶, a refereed conference publication venue including accelerators offered by Cray. Any center needing visibility should participate widely in vendor forums like these as well as traditional academic venues.

9.1.2. Sustainable Benchmarks

Participants all agree that maintaining and evaluating an evolving set of benchmarks and “hero” codes is important to the community. Such benchmarks help developers target appropriate accelerators. Not all software must be sustained. Software targeting inappropriate needs or using inappropriate acceleration technology need not live beyond its initial evaluation, although *knowledge* about the appropriateness of technologies should live through maintained benchmarks. A collection of benchmarks also provides fodder for developing compiler, debugging, and analysis tools.

Beyond prior work, there is a recognized need for information on which accelerators are applicable to which problems. An archive of benchmarks and hero applications supports sustainable software and brings value to both users and vendors. If an accelerator *does not accelerate* the solution to some specific problem, any code targeting that accelerator *should not* be sustained. Identifying which technologies support which applications is a prerequisite to producing sustainable software.

9.1.3. Necessary Qualities of Sustainable Software

Once software both satisfies a need and uses accelerator resources appropriately, the community at large would benefit from some measure of sustainability. There is more to software than its source code, and often the other aspects directly affect the software's long-term prospects.

On one extreme, software can include a model of its resource requirements. Existing measures like computational intensity are excellent examples that can live beyond a specific piece of software or even accelerator platform. Developing and maintaining these performance models is an important service underserved by current efforts. Performance models need little maintenance; their abstraction benefits both users and maintainers. Development of an accurate, predictive performance model is too much to

¹ <http://mgnet.org>

² <http://www-neos.mcs.anl.gov>

³ <http://www.optimization-online.org>

⁴ <http://netlib.org/lapack/lawns>

⁵ http://www.nvidia.com/object/gpu_technology_conference.html

⁶ <http://www.cug.org>

require of many scientists. A center's experts can assist in developing and archiving performance models when that development represents a feasible trade-off between cost and value.

Current typical development of accelerator-based code involves expert programmers (often graduate students) but not documentation experts. The accelerator knowledge encapsulated in the source code, build systems, and execution scripts is not extracted for others to use. Any effort towards sustainability must address such documentation. Expecting graduate students and domain scientists to document all the computational aspects is widely viewed as unrealistic. Trained technical writers would provide immediate value to the community. Such writers can describe the execution mechanisms, use of libraries, and other low-level details. Scientists and students could focus on publishing their new and novel contributions.

There also is wide agreement that any center-maintained software include and *enforce* contribution requirements. Any proposal should address what guidelines will be applied to software. An agreed minimum for sustainable software includes some form of source code documentation and run-time environment requirements. That minimum assumes additional effort to maintain documentation as the code adapts to new techniques, accelerators, and execution environments.

9.1.4. Access to Current and New Accelerators

There was no clear agreement on a method to provide access. Participants generally agreed that having platforms available within the center for performance, correctness, and tool testing is important. Many agree that making systems available to outside collaborators is also important, but the directness and quantity of access is debatable. All agree that access must focus on demonstrating the viability and value of supporting accelerators within applications and libraries.

One option is to be a clearinghouse for information on vendor-hosted demo units with a process similar to the HPC Advisory Council's access process⁷. Another is to support scientist- or developer-in-residence programs where those wanting to accelerate their science codes work closely with developers accustomed to relevant accelerators.

There is universal participant agreement that a focus on sustainability does not require NPACI-level supercomputer systems or access. Every participant felt that large acquisitions of accelerator-based systems are unnecessary and would impede efforts towards broad accelerator support. A focus on smaller-scale development and tuning can leverage *other* efforts when scaling beyond a small system. Obtaining access for a reasonable time period relative to the lifetime of an accelerator is sufficient, and such access and the time periods should be addressed by potential centers.

9.2. ***Supporting Software with a Sustainable Center***

The interactions between center management, application areas, and technology strongly inform any center's potential for sustainability. Here we focus on observations mostly independent of the other sections.

⁷ http://www.hpcadvisorycouncil.com/cluster_center.php

9.2.1. Continuation of a Center

First and foremost, all participants agree that a center must prove its value. That is a precondition for any discussion of the center's sustainability. Any center must prove value not only to the NSF but also to vendors and users. A center should have a concrete plan towards continuing funding beyond the NSF program, establishing also the level of funding appropriate for its changing mission.

Participants agree that a center should have its lifespan broken into phases. There certainly will be an initial start-up phase of collecting hero codes, benchmarks, documentation, and best practices. The center is expected to experience a shift after delivering some major artifacts to the wider community. With major artifacts available, a center likely will include pushing knowledge outward through education and training while still collecting the wider community's accelerator knowledge. As accelerators are adopted into mainstream computing, a center's focus may change in currently unknown ways. The mechanism for addressing these shifts impacts a center's continuance.

Many software infrastructures already exist and have large user communities. Infrastructures spanning from the venerable GNU Emacs environment⁸ to the more recent Eclipse⁹ and NetBeans¹⁰ environments engage significant communities and audiences. Artifacts should leverage not only existing software engineering but also existing communities to establish themselves.

Demonstrated value can lead to vendor funding as in the IUCRC model. Other discussed examples of centers resulting in sustained results include the NSF PACI centers, NSF ERCs, the SRC MARCO centers, and the Department of Energy INCITE program. The Department of Energy SciDAC program is considered an example of trying to sustain software with too little investment in curation of its artifacts or building of community. Individual artifacts found success, but there is little interplay between efforts. The individual successful artifacts justify a program in an established area like traditional high-performance computing. An emerging area like accelerators needs to form a community to weather the boom-bust cycle of any individual technology.

Involvement of international collaborations and funding is an open issue. Participants recognize both the value and the difficulty in international engagement for an NSF-funded organization. Involvement of international users and vendors is a two-way trade. Other nations would gain access to knowledge currently benefiting the United States, while national efforts would gain access to a wider base of technologies and applications. Participants feel that anticipated levels of international involvement should be specified by those who must directly explain international involvement to the nation.

9.2.2. Establishing and Sustaining an Accelerator Community

Participants agree that a community matching science users' needs and accelerator vendors' technologies would provide substantial benefit to both parties. Establishing and sustaining such a

⁸ <http://gnu.org/software/emacs>

⁹ <http://eclipse.org>

¹⁰ <http://netbeans.org>

community helps sustain software and drives infrastructure. Any software infrastructure intended to sustain accelerator software should facilitate community involvement.

Being active in relevant users' groups discussed previously is one mechanism for establishing and growing a community. "Birds of a Feather" groups at major conferences like SC (Supercomputing) are another. Workshops and tutorials co-located at major conferences and later possibly hosted at a center are another well-understood mechanism for nurturing a community. Any center should address its approach towards community involvement along with appropriate timescales.

Drawing in and curating the *output* of NSF SSI and SSE efforts will prove critical in bootstrapping any center. These can form a seed investment in accelerators, and a center should have a strong plan for managing and curating these efforts' outputs.

To draw in vendor contributions and technical support, a center can maintain a list of record-breaking performance and science results similar to the Top500¹¹ but focused on specific benchmarks and hero applications. No accelerator will be good for all uses; that would render it general-purpose. But best-performance lists specifically focused on market areas can attract vendors and encourage repeated participation. A vendor that does not submit new results may be surpassed by another vendor. Continuing this marketing loop has kept supercomputer vendors engaged in the Top500 and interested in its continued publication. Maintaining the balance between the diversity of accelerators and the number and complexity of such metrics needs to be addressed by any center.

9.2.3. Access to Resources and Conditions for Access

The NSF should consider proposers' plans to make technologies available for use and publication of results. Some industry participants stated that they do not use the software artifacts produced, but they do rely on the published results. They are more willing to support a center if they can rely on open publication of new and novel techniques.

10. References

- [ATI-Stream] ATI Stream Technology, www.amd.com/stream (Accessed August 29, 2010)
- [Baker2007] On the Acceleration of Shortest Path Calculations in Transportation Networks, Baker, Z.K. Gokhale, M., in *IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2007.
- [Bell2008] TILE64 - Processor: A 64-Core SoC with Mesh Interconnect, Bell, S., Edwards, B., Amann, J., Conlin, R., Joyce, K., Leung, V., MacKay, J., Reif, M., Liewei Bao, Brown, J., Mattina, M., Chyi-Chang Miao, Ramey, C., Wentzlaff, D., Anderson, W., Berger, E., Fairbanks, N., Khan, D., Montenegro, F., Stickney, J., Zook, J., in *Solid-State Circuits Conference (ISSCC)*, 2008.
- [Bonfiglio2007] Revealing skype traffic: when randomness plays with you, Bonfiglio, D., Mellia, M., Meo, M., Rossi, D., Tofanelli, P. in *Conference on Applications, technologies, architectures, and protocols for computer communications*, 2007, Kyoto, Japan.
- [Convey] Convey Computer Announces Record-Breaking Smith-Waterman Acceleration of 172x, http://www.conveycomputer.com/Resources/Convey_Announces_Record_Breaking_Smith_Waterman_Acceleration.pdf (Accessed August 29, 2010)

¹¹ <http://top500.org>

- [CUDA] NVIDIA CUDA Programming Guide, Version 3.0, http://developer.download.nvidia.com/compute/cuda/3_0/toolkit/docs/NVIDIA_CUDA_ProgrammingGuide.pdf (Accessed August 29, 2010)
- [CUDA-Community] CUDA Community Showcase: GPU computing applications developed on the CUDA architecture by programmers, scientists, and researchers around the world. http://www.nvidia.com/object/cuda_apps_flash_new.html (Accessed August 29, 2010)
- [DARPA-Exascale] ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems, Kogge, P. M. (Editor), Bergman, K., Borkar, S., Carlson, W. W., Dally, W. J., Denneau, M., Franzon, P. D., Keckler, S. W., Klein, D., Lucas, R. F., Scott, S., Snavely, A. E., Sterling, T. L., Williams, R. S., Yelick, K. A., Harrod, W., Campbell, D. P., Hill, K. L., Hiller, J. C., Karp, S., Richards, M. A., Scarpelli, A. J., September 28, 2008.
- [Datta2008] Stencil computation optimization and auto-tuning on state-of-the-art multicore architectures, Datta, K., Murphy, M., Volkov, V., Williams, S., Carter, J., Oliker, L., Patterson, D., Shalf, J., Yelick, K., in *International Conference on High Performance Computing, Networking, Storage and Analysis (Supercomputing)*, 2008.
- [FPGA-Vendors] SRC Computers, Convey Computers, Maxeler Technologies, Cray Inc, Pico Computing, XtremeData Inc, Xilinx Inc, Altera Inc.
- [Fu2009] Accelerating 3D Convolution using Streaming Architectures on FPGAs, Fu, H., Clapp, R. G., Mencer, O., and Pell, O., in *Annual Meeting of Society of Exploration Geophysicists (SEG)*, Houston, 2009.
- [Ganegedara2010] Automation Framework for Large-Scale Regular Expression Matching on FPGA, Ganegedara, T. Yang, Y. E., and Prasanna, V. K. in *International Conference on Field-Programmable Logic and Applications (FPL)*, 2010.
- [Hall2010] EigenCFA: Accelerating flow analysis with GPUs, Prabhu, T., Ramalingam, S., Might, M., and Hall, M., in *ACM Symposium on the Principles of Programming Languages (POPL)*, 2011.
- [HPC-FPGA] High Performance Computing Using FPGAs, Sundararajan, P., *Technical Report WP375 (v1.0)*, September 10, 2010, Xilinx Inc.
- [HT] Programming with Hyper-Threading Technology How to Write Multithreaded Software for Intel® IA-32 Processors, Binstock A. and Gerber, R., 2003, Intel Press.
- [Hua2008] Compact Architecture for High-Throughput Regular Expression Matching on FPGA, Yang, Y. E., Jiang W. and Prasanna, V. K., in *ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, 2008.
- [Hua2010a] High Performance Dictionary-Based String Matching for Deep Packet Inspection, Yang, Y. E., Le H. and Prasanna, V. K., in *IEEE International Conference on Computer Communications (INFOCOM)*, 2010.
- [Hua2010b] Head-Body Partitioned String Matching for Deep Packet Inspection with Scalable and Attack-Resilient Performance, Yang, Y. E., Prasanna, V. K., and Jiang, C., in *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2010.
- [Jeon2010] Parallel Exact Inference on a CPU-GPGPU Heterogeneous System, Jeon, H., Xia, Y. and Prasanna, V. K., in *International Conference on Parallel Processing (ICPP)*, 2010.
- [Le2010] A Memory-Efficient and Modular Approach for String Matching on FPGAs, Le, H. and Prasanna, V. K., in *IEEE Symposium on Field Programmable Custom Computing Machines (FCCM)*, 2010.
- [Miller2010] Graphite: A distributed parallel simulator for multicores, Miller, J.E., Kasture, H., Kurian, G., Gruenwald, C., Beckmann, N., Celio, C., Eastep, J., and Agarwal, A., in *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2010.
- [NSF 10-050] Scientific Software Innovation Institutes (S2I2) – Call for Exploratory Workshop Proposals, José Muñoz, *NSF Dear Colleague Letter*, NSF-10-050, National Science Foundation, Apr 13, 2010
- [NSF 10-551] Software Infrastructure for Sustained Innovation (SI2), *NSF Program Solicitation*, NSF-10-551, National Science Foundation, Nov 7, 2006.
- [Putnam2009] Performance and power of cache-based reconfigurable computing, Putnam, A., Eggers, S., Bennett, D., Dellinger, E., Mason, J., Styles, H., Sundararajan, P., and Wittig, R., in *International Symposium on Computer Architecture*, 2009.

- [Rusu2010] A 65-nm Dual-Core Multithreaded Xeon Processor With 16-MB L3 Cache, Rusu, S., Tam, S., Muljono, H., Ayers, D., Chang, J., Cherkauer, B., Stinson, J., Benoit, J., Varada, R., Leung, J., Limaye, R. D., Vora, S., *IEEE Journal of Solid-State Circuits*, vol.42, no.1, pp.17-25, Jan. 2007.
- [Shafer2010] Enumeration of Bent Boolean Functions by Reconfigurable Computer, Shafer, J. L., Schneider, S.W., Butler, J.T., Stănică, P., in *IEEE Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2010.
- [Shin2010] A 40nm 16-core 128-thread CMT SPARC SoC processor, Shin, J.L., Tam, K., Huang, D., Petrick, B., Pham, H., Changku Hwang, Hongping Li, Smith, A., Johnson, T., Schumacher, F., Greenhill, D., Leon, A.S., and Strong, A., in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2010.
- [Storaasli2007] Exploring Accelerating Science Applications with FPGAs, Storaasli, O. O. and Strenski, D. G., *NCSA/RSSI Proceedings*, Urbana, IL, July 20, 2007.
- [Wendel2010] The implementation of POWER7TM: A highly parallel and scalable multi-core high-end server processor, Wendel, D., Kalla, R., Cargoni, R., Clables, J., Friedrich, J., Frech, R., Kahle, J., Sinharoy, B., Starke, W., Taylor, S., Weitzel, S., Chu, S.G., Islam, S., and Zyuban, V., in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2010.
- [XMT] Cray XMT System Overview, *Technical Report S-2466-14*, December 2009, Cray Inc.
- [Yang2007] GPU accelerated molecular dynamics simulation of thermal conductivities. Yang, J., Wang, Y., and Chen, Y., *Journal of Computational Physics*, vol. 221, no. 2, pp. 799—804, 2007.
- [Yang2008] Compact Architecture for High-Throughput Regular Expression Matching on FPGA, Yang, Y. E., Jiang, W. and Prasanna, V. K., in *ACM/IEEE Symposium on Architecture for networking and communications systems (ANCS)*, 2008.
- [Yi2010] High Throughput and Large Capacity Pipelined Dynamic Search Tree on FPGA, Yang, Y. E. and Prasanna, V. K., in *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*, 2010.
- [Yoo2009] Phoenix rebirth: Scalable MapReduce on a large-scale shared-memory system, Yoo, R.M., Romano, A., Kozyrakis, C. in *IEEE International Symposium on Workload Characterization (IISWC)*, 2009.
- [Yu2006] Fast and Memory-Efficient Regular Expression Matching for Deep Packet Inspection, Yu, F., Chen, Z., Diao, Y., Lakshman, T. V. and Katz, R. H., in *Proceedings of ACM/IEEE Symposium on Architecture for networking and communications systems (ANCS)*, 2006.

Appendix A: Meeting Agenda

Day 1 (Wednesday, October 13)

0730–0815 Coffee/Donuts

0815–0830 Welcome, introductions, workshop overview, objectives, Viktor K. Prasanna, USC and David A. Bader, Georgia Tech

0830–1000 20+10 min each presentation+discussion

Accelerator Technology Area 1: GPUs, Rich Vuduc, Georgia Tech

Accelerator Technology Area 2: FPGAs, Tony Brewer, Convey Computer

Accelerator Technology Area 3: Multi-core, Steve Crago, USC/ISI

1000–1015 Break

1015–1030 S2I2 Initiative, Manish Parashar, Program Manager, OCI, NSF

1030–1200 20+10 min each presentation+discussion

Application Area 1: Bio, Srinivas Aluru, Iowa State /IIT Bombay

Application Area 2: Social Network, Jason Riedy, Georgia Tech

Application Area 3: Security, Jeff Hollingsworth, Univ of Maryland

1200–1300 Working lunch

1300–1400 Center organization and management, metrics for evaluation, David A. Bader, Georgia Tech and Viktor K. Prasanna, USC

1400–1415 Break

1415–1515 Software infrastructure/sustainability specific to accelerators: what interfaces? How do the applications communities benefit?, Pavan Balaji, Argonne Nat'l Lab

1515–1530 Break

1530–1630 Accelerator software development challenges, Volodymyr Kindratenko, UIUC/NCSA

1630–1645 Break

1645–1800 Report preparation discussion, break-out sessions on day 2, assignments, time line

Day 2 (Thursday, October 14)

0730–0815 Coffee/Donuts

0815–0830 Overview of report writing tasks

0830–1000 Breakouts A and B

Breakout A: Center models and management, David A. Bader, GA Tech and Viktor K. Prasanna, USC

Breakout B: Software development, Karthik Gomadam USC

1000–1030 Break

1030–1200 Breakouts C and D

Breakout C: Understanding the community, Yogesh Simmhan, Microsoft Research/USC

Breakout D: Sustainability, Jason Riedy, Georgia Tech

1200–1400 Working lunch: Wrap up, final assignments

1400 Adjourn

Appendix B: List of Participants

Name		Affiliation
Srinivas	Aluru	Iowa State University
Peter	Athanas	Virginia Tech
David A.	Bader	Georgia Institute of Technology
Pavan	Balaji	Argonne National Laboratory
George	Biros	Georgia Institute of Technology
Tony	Brewer	Convey Computer
Richard	Brower	Boston University
Steve	Casselman	DRC Computer Corporation
Andrew	Chien	University of California-San Diego
Steve	Crago	University of Southern California / ISI
Matt	French	University of Southern California / ISI
Maya	Gokhale	Lawrence Livermore National Laboratory
Karthik	Gomadam	University of Southern California
Zhi	Guo	Huawei
Anshul	Gupta	IBM T.J. Watson Research Center
Jeff	Hollingsworth	University of Maryland
Volodymyr	Kindratenko	National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign
Michael	Mahoney	Stanford University
Sukarno	Mertoguno	ONR
Henning	Meyerhenke	Georgia Institute of Technology
Ken-ichi	Nomura	University of Southern California
Manish	Parashar	National Science Foundation
Viktor K.	Prasanna	University of Southern California
Jason	Riedy	Georgia Institute of Technology/CASS-MT
Yogesh	Simmhan	University of Southern California/ Microsoft Research
David	Strenski	Cray Inc.
Prasanna	Sundararajan	Xilinx
Rich	Vuduc	Georgia Institute of Technology
Ross	Walker	San Diego Supercomputer Center
Rand	Waltzman	DARPA