RC25406 (IRE1308-033) August 26, 2013
Other

IBM Research Report

The ExaChallenge Symposium

Rolf Riesen
IBM Research
Smarter Cities Technology Centre
Mulhuddart
Dublin 15, Ireland

Sudip Dosanjh
LBNL/NERSC

Larry Kaplan
Cray, Inc.

Research Division
Almaden - Austin - Beijing - Cambridge - Dublin - Haifa - India - T. J. Watson - Tokyo - Zurich

The ExaChallenge Symposium

Rolf Riesen, IBM Research - Ireland Sudip Dosanjh, LBNL/NERSC
Larry Kaplan, Cray Inc.

October 16–18, 2012

Abstract

The ExaChallenge symposium was held on October 17 and 18, 2012 at the IBM Research laboratory in Dublin, Ireland. The symposium brought together a small group of highly qualified exascale computing experts from institutions in the USA and Europe. A unique symposium format allowed the inclusion of all participants in the discussions of the software and managerial challenges laying ahead. These often lively discussions highlighted areas of concern, disagreement about the correct solution, and open questions that need to be addressed before an exascale system can be put into production use. This report summarizes these discussions and findings.

Contents

List of Figures

List of Tables

1 Executive summary

The first ExaChallenge symposium brought together a small group of highly qualified exascale computing experts to discuss the software and managerial challenges laying ahead before an exascale-size system can be successfully deployed and put into production use. Participants were drawn from academia, vendors, and national research laboratories, in the USA and Europe.

Previous exascale workshops and meetings have identified key areas that need research and development in order to advance the state-of-the-art to the exascale level. Although it is expected that the first exascale systems will appear near the end of this decade, many questions remain to be answered. One of the main goals of the ExaChallenge symposium was to take stock of the progress in our understanding of how to advance three orders of magnitude from the current petascale systems, whether the current research projects are addressing the right questions and are making enough progress, and identify gaps that need to be filled before the exascale vision can be achieved.

In its first incarnation the symposium sought to elicit concerns leading figures in the field have and which approaches they deem promising in the march toward exascale. The symposium was organized as ten sequential sessions addressing topics in systems software from APIs to runtimes for exascale, and topics at the application level on how to deal with the increased complexity. Throughout, the goal was to assess the current state of the art and identify challenges still laying ahead.

The format of the symposium allowed all participants to take an active role in the discussion. All experts present were able to comment and participate in the discussions. Although we had speakers, the sessions were not intended as a podium for individual presentations. To that end, each session had a leader and a wingmate. The role of the session lead was to give a short presentation on the topic to be discussed, and then lead a discussion among all symposium participants. The wingmate's role was to assist the session lead in generating an active discussion. This was done by supplying additional information or playing devil's advocate.

The resulting discussions were often animated and often clearly showed how far opinions differed on whether our current approaches to reach exascale are working or not. The participants expressed approval of this type of symposium because it was highly participatory, generated ideas, and identified areas of disagreement. Although the symposium had ten sessions, not all areas necessary to achieve exascale computing could be covered within the time available and the subject areas represented by the participants. Notable exceptions were hardware architecture and funding for these extreme-scale scientific instruments.

On the question of API standardization, most participants felt it was too early since the approaches on how to program and manage these systems are still in flux. This was despite the acknowledgment that the standardization process takes time and it would be useful for standards to be available when the first systems appear. Runtime systems and programmability (the programmer's burden) were identified again as critical. The role of future runtime systems and OSes for exascale is not yet well defined. They are expected to manage global aspects such as power consumption and allowing computation on data in situ, while giving (legacy) applications the local functionality a full featured OS, like Linux, provides. Although some progress has been made in these areas, much is left to do. Co-design centers and the HPC community at large may be particularly of help in this area, but it requires vendor involvement.

Also acknowledged was the need for more research, innovation, and education. This clashes somewhat with the hoped-for delivery of the first exascale systems by the end of the decade. The lack of time, access to such extreme-scale systems, and experience with them means research and learning will have to occur alongside deployment of early systems. At the same time, the next generation of computer and computational scientists needs to be trained to work with these extreme-scale systems.

Research and education are also crucial in systems and application design for future systems. There was no consensus on whether a single type of system could both address the commercial data center

market and still be suitable to solve demanding scientific problems. It may be that approaching the physics problems to be solved from a different angle may go a long way in producing applications that work better on these envisioned systems. However, this requires further research and better and more targeted education of future code developers.

Application developers currently have to deal with several, incompatible ways of using accelerators. Portability is limited, but in order to get the highest performance, applications have to make use of these technologies. On the horizon are application changes that provide hints to the system on how faults should be handled. Applications probably also need to help conserve power consumption. Data movement and dealing with millions of threads are additional challenges that need to be tackled now by evolving applications. Programming languages and models that hide some of the complexity of a modern supercomputer, yet allow the expression of data locality, are needed but not really in sight yet.

Research in power, fault tolerance, parallelism, and data movement inside a deep memory hierarchy needs to be done. Identifying and funding the research that can ignite a disruptive change is difficult, yet may be necessary to make the significant advances that are needed.

Simulation and co-design are key contributors to success but face large hurdles. Simulating these extreme-scale systems grows exponentially more complex. Different experiments need more accuracy in some parts of the simulated system, but can live with less accuracy in other parts. Providing a modular simulator that lets the experimenter shift the accuracy focus is made even more difficult by vendor IP issues for components that need to be simulated in high resolution to get valid power consumption information.

Although co-design centers have been established and have started to produce mini-apps that help in machine procurement and act as conduits to try out new algorithmic and systems software ideas, a lot more needs to be done. The co-design centers need to increase their interactions with application, tool, and system software developers, and establish technology paths to vendors. They also have to make sure they are addressing exascale, not just the next generation of machines.

Fault tolerance remains a hot topic with a lot of uncertainty about what to expect from future extrem-scale systems, where to best apply fault tolerance in the software stack, and how much work applications will have to do to reach a scalable solution. Even if the number of faults and their severity in future systems will be fewer and less than what some people currently expect, it is still beneficial to reduce the overhead of fault tolerance mechanisms. Even small savings improve utilization of these systems and can save millions of Dollars.

Figure 1: Group picture of the 2012 symposium participants.

2 Introduction

The first ExaChallenge symposium was held at IBM's research laboratory in Dublin, Ireland from October 16 to 18, 2012. It was organized by *Rolf Riesen*, IBM Research; *Sudip Dosanjh*, Sandia National Laboratories (now LBNL/NERSC); and *Larry Kaplan*, Cray Inc. Figure 1 shows most of the participants of the 2012 symposium. Table 1 lists all participants and their institutions.

2.1 Goals

The topic of exascale systems and computing has been discussed in many forums and workshops over the last few years. Outstanding examples include the International Exascale Software Project (IESP) series of workshops [13] and the US DOE commissioned reports on technological challenges [6] and software challenges [2].

The prediction is still that the first exascale systems used as scientific instruments will appear before the end of the decade. Much progress has been made since the first reports appeared outlining the challenges ahead. The goal of the ExaChallenge symposium was to take a snapshot of the current state and assess whether our research agendas are still appropriate or need to be revised. A secondary goal was to explore compromises that may need to be made to reach exascale.

For example, it is important that application designers and system developers interact with each other. At exascale, some assumptions about fault tolerance, scalability, and runtime support will change and impact the role an application has to play when interacting with system services. Another area where

Table 1: ExaChallenge 2012 symposium participants

Name	Institution
Gabriel Antoniu	INRIA
Ron Brightwell	Sandia National Laboratories (SNL)
Sudip Dosanjh	LBNL/NERSC
Turlough Downes	Dublin City University (DCU)
Christian Engelmann	Oak Ridge National Laboratory (ORNL)
Kurt Ferreira	Sandia National Laboratories (SNL)
Vladimir Getov	University of Westminster
Hermann Härtig	TU Dresden
Simon Hammond	Sandia National Laboratories (SNL)
Larry Kaplan	Cray Inc.
Kostas Katrinis	IBM Research - Ireland
Ludek Kucera	Charles University
Alexey Lastovetsky	University College Dublin (USC)
Pierre Lemarinier	IBM Research - Ireland
Barney Maccabe	Oak Ridge National Laboratory (ORNL)
Jeffrey Nichols	Oak Ridge National Laboratory (ORNL)
Bogdan Nicolae	IBM Research - Ireland
Dimitrios Nikolopoulos	Queen's University of Belfast
Brian Quinn	Intel
Mustafa Rafique	IBM Research - Ireland
Rolf Riesen	IBM Research - Ireland
Arun Rodrigues	Sandia National Laboratories (SNL)
Duncan Roweth	Cray Inc.
Thomas Schulthess	Swiss National Supercomputing Center (CSCS)
Thomas Sterling	Indiana University - CREST
Aidan Thompson	Sandia National Laboratories (SNL)
Henry Tufo	University of Colorado at Boulder
Sudhakar Yalamanchili	Georgia Institute of Technology

Table 2: Program for Tuesday, October 16, 2012

Start	End	Duration	Event	Location
18:00	19:00	1:00	Pre-dinner drinks	Dunboyne Castle
19:00	21:00	2:00	Dinner	Hotel

compromises may become necessary is in the area of performance and system scalability. Although 10^{18} operations per second is the goal, not all aspects of the system will improve with the same factor. Technologies not originally intended for supercomputing, from the product lines aimed at commercial data centers, may need to be utilized in order to make production of exascale systems viable.

There are many areas that need to be studied, analyzed, and possibly reevaluated in order to reach exascale. This symposium was an initial stab at the mountain of work that lays ahead.

2.2 Session format

The main purpose of the symposium was to generate and exchange ideas. Since almost all participants are experts in the field, a session format that facilitates exchange of information, rather than a one-way flow, seemed most appropriate. Panels at computer science conferences allow the panelist to express their opinions and provide information. Sometimes there is discussion among the panelists, but almost never is the audience involved much.

For this symposium we intended to give all participants the opportunity to be active in the discussions and contribute. Each session was dedicated to a specific discussion topic. A **session lead** had the task of introducing the topic and then spur and motivate a discussion among all participants. A session lead may have to take on the role of a teacher, devil's advocate, or interviewer.

Session leads were told they could use the first ten to fifteen minutes of each hour-long session to start. All session leads chose to give a brief presentation at the beginning of their session. That was not required, however. Motivational speakers, if they wanted, could have begun a session without visual aids.

Because this session format is somewhat uncommon, the organizing committee felt that a backup would be valuable, should the discussion come to an early halt. For each session we selected a **wingmate**. That person's task was to assist the session lead in promoting an active discussion among all participants. In that role the wingmate may play devil's advocate to the session lead, pose additional questions to the session lead or the participants, or support the session lead by providing additional information or answers to questions and challenges. In short, the wingmate's task was to assist the session lead in assuring that the discussion does not run dry and make it interesting for all participants.

The idea was that session leads and their wingmates communicate before the symposium and coordinate a strategy to keep their session going.

2.3 Final program

Tables 2, 3, and 4 list the final program of the symposium. *Lisa Amini*, distinguished IBM engineer and director of the Dublin research lab, gave a brief welcome to the symposium participants. This was followed by each participant briefly introducing themselves.

The sessions and breaks were kept on time, although often the discussions could have easily gone on for longer. In a future meeting, longer sessions may be appropriate to allow for more in-depth discussions.

Table 3: Program for Wednesday, October 17, 2012

Start	End	Duration	Event	Location
09:00	09:10	0:10	Welcome *Lisa Amini*	Exposition Space
09:10	09:30	0:20	Introductions	Exposition Space
09:30	10:30	1:00	Session 1 **Common community APIs** Session lead: *Larry Kaplan* Wingmate: *Dimitrios S. Nikolopoulos*	Exposition Space
10:30	10:45	0:15	Break	Yeats room
10:45	11:55	1:10	Session 2 **HPC runtime opportunities and challenges** Session lead: *Thomas Sterling* Wingmate: *Hermann Härtig*	Exposition Space
12:00	12:55	0:55	Lunch	Cafeteria, Building 2
13:00	14:00	1:00	Session 3 **The programmer's burden** Session lead: *Ron Brightwell* Wingmate: *Turlough Downes*	Exposition Space
14:00	15:00	1:00	Session 4 **Research challenges** Session lead: *Barney Maccabe* Wingmate: *Vladimir Getov*	Exposition Space
15:00	15:15	0:15	Break	Yeats room
15:15	16:15	1:00	Session 5 **Exascale simulations** Session lead: *Sudhakar Yalamanchili* Wingmate: *Arun Rodrigues*	Exposition Space
16:15	17:15	1:00	Session 6 **Co-design** Session lead: *Sudip Dosanjh* Wingmates: *Aidan Thompson and Simon Hammond*	Exposition Space
19:30	20:30	1:00	Pre-dinner drinks	The Church Bar
20:30	22:00	1:30	Dinner	and Restaurant

Table 4: Program for Thursday, October 18, 2012

Start	End	Duration	Event	Location
09:30	10:30	1:00	Session 7 **Expanding the scope of traditional HPC systems** Session lead: *Duncan Roweth* Wingmate: *Rolf Riesen*	Exposition Space
10:30	10:45	0:15	Break	Yeats room
10:45	11:55	1:10	Session 8 **Application perspective** Session lead: *Thomas Schulthess* Wingmate: *Henry Tufo*	Exposition Space
12:00	12:55	0:55	Lunch	Cafeteria, Building 2
13:00	14:00	1:00	Session 9 **Fault tolerance** Session lead: *Christian Engelmann* Wingmate: *Larry Kaplan*	Exposition Space
14:00	15:00	1:00	Session 10 **Next steps** Session lead: *Jeffrey Nichols* Wingmate: *Thomas Sterling*	Exposition Space
15:00	15:15	0:15	Break	Yeats room
15:15	16:15	1:00	Wrap-up *All*	Exposition Space

3 Sessions

There were ten sessions over the course of two days, each held according to the format described in Section 2.2. The following sub-sections each have the same structure: A heading listing the session lead and wingmate, the pre-symposium description of that session, a summary of the presentation given at the beginning of the session, excerpts from the discussion following it, and a post-symposium summary of the session.

The first time a participant's name appears in this document, and when it appears in listings, the full name is given. After that, only first names are used to identify speakers. The exceptions are *Thomas Sterling* and *Thomas Schulthess*, unless it is clear from the context which Thomas is meant.

3.1 Common community APIs

Leads

Session lead *Larry Kaplan*, Cray Inc.
Wingmate: *Dimitrios Nikolopoulos*, Queen's University of Belfast.

Description

> A variety of vendor specific hardware and software is expected to be developed for exascale and HPC. This variety may pose a challenge to both application and other software designers and limit portability. In which areas could common APIs help to bridge the portability gap while providing access to new features? What is a good way to create the necessary APIs and standardize them with vendor and user participation? How soon should this be done? What standardization challenges exist?

Presentation

Larry Kaplan starts his presentation by asking why it is important to have common APIs for extreme-scale systems. He points out that there will be more than one exascale system and, therefore, a need for portability. *Larry* stresses that APIs for portability are needed at the application level but also in lower layers of the software stack. While existing programming environments and languages are defined, that is not necessarily true for new languages, new runtime environments, or new OSes proposed for these emerging systems.

It is very likely that OS and runtime interfaces will change in the future to adapt to the needs of exascale systems. The hope is that scalable interfaces for tools, process and job management, fault tolerance, power management, and I/O will evolve. In order for users to take advantage of these new options and the research community to provide interoperable and alternative implementations, common APIs have to be in place. This would facilitate vendors and the community to work together at the leading edge.

Many of the challenges predicted for exascale systems; e.g. power management and fault resilience, will require multiple levels in the software stack to work in cooperation. *Larry* lists as one example the APIs needed for resilience. Well defined interfaces are needed at the MPI level to let applications know when things go wrong and let them instruct the system how a given fault should be handled. This in turn requires interaction between the MPI library and the runtime system, which in turn relies on system services to react appropriately. Of course, the OS needs to provide APIs for detection and reporting of faults, as well as other services needed to manage processes.

Not all the pieces of these complex software modules will be written by the same people, and multiple implementations may exist. Since they need to interact with each other, some standardization is required. The question is when these API standardization efforts should take place. Some argue that if that process starts too soon, a sub-optimal solution may be chosen before the best solution is known. However, standardization takes time, and even more time is needed after that to build products based on these standards. For things such as MPI and Fortran, which are clearly defined and have started looking at some of these issues, we have a little bit more time, while other efforts for less clearly defined interfaces need to start now.

Discussion

Dimitrios Nikolopoulos asks how we can cope with new technology, how to plan ahead, and whether we can influence emerging hardware early on. *Larry* suggests that depending on technology, lower level solutions that are not visible at the user level may be appropriate.

Ron Brightwell feels that an API should not be a way to shift complexity away from the application programmer. The responsibility for a scalable end product should be shared. As examples he lists system programmers who do not necessarily have control over the runtime above, and network interfaces built for MPI only, without regard to other parts of the system, such as the runtime, which also needs to exchange information and cannot use MPI to do it.

Barney Maccabe asks how standardization of common APIs can be started. He mentions Portals [9] as an example of one group's effort to create an API for low-level message passing. For wide spread adoption, buy-in from more vendors and laboratories is needed. How to get that? *Ron* adds that it is difficult to get people to abandon old APIs and mentions POSIX as an example. *Jeffrey Nichols* uses OpenACC [12] as an example of how hard this can be. OpenACC is meant to replace Nvidia's CUDA [27], which is not entrenched yet, but OpenACC is already struggling to find a foothold without vendors pushing it more aggressively.

Alexey Lastovetsky says it is obvious that APIs are necessary and that we need efforts like the MPI standardization, aimed at scheduling, monitoring, and power management. *Larry* asks whether that is research and *Alexey* answers that such a process utilizes research. One facet of it is to ask people what their needs are and learn from experience. *Alexey* feels that it is too early to standardize.

This prompts *Sudhakar Yalamanchili* to ask whether vendors are exposing enough information about their low-level hardware. *Thomas Sterling* chimes in saying that runtime systems will influence higher level APIs. This is one aspect the co-design centers are supposed to explore: How to get from the application level down to the lower levels? *Simon Hammond* suggests to interact with, and talk to, these centers. Their goal is to optimize hardware and software down to a very low level. It would seem that the co-design centers should play an important role in any API standardization efforts, but, at the moment, do not.

This confuses *Aidan Thompson*. He speculates that APIs are not discussed within the co-design centers at the moment because "a lot of things are third on the priority list." The centers have to consider scalability, performance, power, resilience, and more, *Aidan* adds.

"A lot of things are third on the priority list." (*Aidan Thompson*)

Duncan Roweth points out that we are at the leading edge with these systems. For the moment, platform-specific solutions may be the way to go. First learn how to use these systems before APIs get carved into stone. *Ron* concurs and says that we can standardize after we have learned how to make these things work. This is akin on how MPI came along. There were many different message passing systems, usually machine specific, before the community had learned what is needed and what is important. The need for cross-platform portability then led to MPI. *Duncan* seems to agree when he says that vendors need to have an interest; i.e., incentive, before standardization can happen.

Alexey feels that both approaches could be done at the same time: Start new paradigms and begin porting legacy codes. *Sudip Dosanjh* warns that standardization may not be sufficient for performance portability. This prompts *Vladimir Getov* to mention MPI and HPF: some efforts succeed, while others do not. Although the participants at the symposium did not delve further into this, it may be worthwhile to look at previous standardization efforts and what can be learned from them [18].

Barney sees a bigger challenge: Even if a well-defined, working API is created, it may still not succeed if the hardware and software layers it is built upon are a disaster. *Larry* emphasizes that the issue is to identify which pieces matter and to get people interested and involved. *Thomas Sterling* says that priorities need to be set: APIs for programmability and performance portability are further down on the list. *Barney* throws in that Linux is not a suitable API for power management. *Thomas* counters that we need to build a proof of concept and then throw it away. Do that twice. He wants an API that isolates the runtime so that the runtime can change. He stresses that the potential needs to be demonstrated, before standardization is possible.

"Linux is not a suitable API for power management." (*Barney Maccabe*)

Both *Thomas* and *Barney* agree that there are all kinds of standards, many of them not interoperable, and trying out new designs is fine. The question is how to drive adoption. *Thomas* cautions that we could be wrong and should wait to push for adoption before we have a qualified model. How good does this model or design have to be? Does it need to be proved? Existing approaches and APIs should not constrain us.

"Everything you disagree with is Ron's fault!" (*Thomas Sterling*)

During the conclusion of this session, *Dimitrios* asks whether APIs are really a problem in the end. *Barney* answers that it is a huge investment for applications to move from one platform to another. Therefore, applications move cautiously. He mentions the acceptance rate of GPU accelerators as an example, and states that the first exascale "application" (Linpack) will use MPI.

Ron says that vendors have APIs, even though they may be machine specific, and that they should be exposed. This would allow others to make use of them and integrate them in higher level APIs.

Summary

Although *Larry* stresses the need to begin API standardization now, many people in the room feel it is too early; that more experience with these systems is needed before successful APIs can be created. Furthermore, several of the participants believe that a more organic approach is okay: Let the need grow stronger and then use the experience gained at that time to guide a standardization effort.

3.2 HPC runtime opportunities and challenges

Leads

Session lead: *Thomas Sterling*, Indiana University.
Wingmate: *Hermann Härtig*, TU Dresden.

Description

> This session will discuss the need for exposing and exploiting information about system execution state on a continuing basis and applying it to task scheduling and resource management as well as to discover new parallelism on the fly. The objective of runtime system software for HPC is to make dramatic improvements in efficiency and scalability. But it imposes additional overheads that can also be a source of performance degradation. This session will consider the balance between these contending influences.

Presentation

Thomas begins his presentation by pointing out that many different classes of architectures; e.g. Tianhe-1A, KEI, and the BG/Q, already exist and that many more, such as Intel's Many Integrated Core architecture (MIC) [38], are to come. We are in flux. These hardware architecture changes will force software stack changes. Runtime systems, so far mostly eschewed by HPC for performance reasons, will be game changers. *Thomas* foresees runtime systems that are ephemeral, dedicated and existing only within an application. He is not talking about traditional runtime systems which are a persistent part of the OS and dedicated to the hardware system. These new systems will not deliver 100% of the available hardware performance, but will enable a move from a static to a dynamic operational regime. This enables a system to adapt as it is running and behave more like a guided missile with continuous course correction rather than a fired projectile with a fixed trajectory.

Performance in the future will depend on many factors, including efficiency, an application's parallelism, the availability and reliability of a system, and effects such as starvation, latency, overhead, and waiting for contention resolution. *Thomas* asserts that only a dynamic system can possibly cope with all of these factors. While this sounds complicated and burdensome, there are also many opportunities to address efficiency, scalability, programmability, performance portability power/energy, and reliability.

In the future it may be necessary to focus on memory bandwidth rather than floating point performance when evaluating resource utilization. It may also be necessary to move the work to the data, instead of the current model where data has to flow to the processor. Addressing scalability is also important: There needs to be enough work to be done by a thread. How to discover parallelism, and what granularity of parallelism is right?

Although *Thomas* believes that a dynamic runtime system can address these issues, he is aware of the challenges that lie ahead. In particular, such sophisticated runtime systems will add overhead and scheduling may bound the effective granularity and therefore limit concurrency and scalability. Acknowledging the previous sessions, *Thomas* also lists OS interfaces as one of the challenges. It will be necessary to lift some responsibilities from the OS up into the runtime and impose new demands upon the OS.

Many of these challenges are being addressed by the X-Stack [28] program and in particular the current XPRESS [19] program which encompasses a variety of initiatives to enable exascale performance of future DOE computing systems. *Thomas* stresses that a new execution model is needed to achieve exascale. An incremental approach will not do and a bigger jump is needed. We are in a crisis already because more and more codes do not scale.

Thomas initiates the discussion by asking four questions:

1. Will a runtime system deliver what we need?
2. How does an HPC runtime system change, positively and negatively, the programming models?
3. How does such a runtime system interact with future OSes?
4. How should we proceed to achieve a viable runtime system software component for exascale?

Discussion

Hermann Härtig asks how an OS would need to be structured to comply with the vision *Thomas* has just presented. *Hermann* asked whether some of the challenges; e.g., resource management had not been already solved in real-time systems and OSes like MOSIX [3, 4]. *Ron* agrees with *Hermann* that there are lessons to be learned from embedded and real-time systems. He says the difference is in the goals; there is no system-wide view in an exascale system (MOSIX provides a single-system image of a system).

Jeff says the US exascale effort is different from the rest of the world. There is a 20 MW constraint on such systems, although vendors say they may need 40 MW. This may require that CPUs and memories be turned off at times to lower power consumption. *Jeff* asks whether the runtimes described by *Thomas* can help with that. *Thomas* replies that the information a runtime derives from an application could be used to control power. *Jeff* thinks this would be a good co-design example: Set 20 MW as a constraint and work with application and hardware people to meet it.

Dimitrios wonders whether letting the runtime system manage power consumption is enough, while *Vladimir* states that power management is already being worked on by vendors. Solutions will come from mobile computing and other low-power devices. *Thomas* says that we are off by an order of magnitude. He says that we need to turn off power to the communication subsystem when we are not moving data. This prompts *Barney* to say that a runtime can also get into the way. How can it help an application to control power. *Thomas* says that the runtime would act as a conduit.

Barney reiterates his question: How does a runtime system help [with power]? Does the runtime observe an application and control its energy consumption, or does the runtime provide an API that an application can use to self-control? *Ron* interjects that mobile phones are a perfect example: The support [to control power] must be provided and exposed by the hardware first, then the OS and applications can adapt. Building on that, *Alexey*, referring to the OpenX software architecture diagram *Thomas* showed, suggests that the runtime should expose communication at all levels. The application should be allowed to manage the communication resource.

Taking a broader view, *Sudhakar* says that management of resources has historically been in time and space. To reach a sub-20 MW exascale system, more than a local power API will be needed. Examples are moving computation to the data and scheduling at various levels. The runtime needs to become “omnipresent” with a multidimensional cost model and at different time scales. *Larry* jokes that the runtime is kind of like the government: “We are here to help you,” and then more seriously asks whether such an all-encompassing runtime leaves room for an OS. *Thomas* states that we have not had a runtime in HPC yet and *Ron* says that protection and isolation are the tasks of an OS.

Larry says we should have an OS inside accelerators and *Barney* asks what for, *Larry* lists cross-mapped memory as an example that is very difficult to debug. At that point *Sudhakar* asks whether the accelerator model will persist. What if they were incorporated as first class models? They need to be on the memory bus. He asks *Larry* what the role of the OS and the runtime would be in that case.

Because *Thomas* mentioned that each application would have its own ephemeral runtime, *Christian Engelmann* asks whether MPI and OpenMP would need their own runtime systems. *Thomas* replies that everything is in tight interplay: The runtime, the programming model, etc. The question is what the next steps to take are.

For this to work, *Jeff* says that the co-design centers need to be vendor agnostic, otherwise we cannot succeed. We need more centers and bigger platforms to work on. *Sudip* says that although the IAA [20, 21] started co-design in 2008, we have not yet fully figured out how all these things can be integrated. A big question is what the right level of abstraction is. It is possible that intellectual property (IP) is a problem. The HPC community needs to speak to vendors with one voice. That will be a challenge but is necessary for large companies like AMD, Nvidia, and Intel. *Larry* asks whether Cray Inc. could act as a conduit, but *Jeff* thinks that the community needs to talk to the vendors directly.

Summary

Thomas Sterling's vision for HPC runtime systems of the future presents a complex structure touching all parts of a high-end system from programming models to system software to hardware. Such runtime systems may help control power consumption and regulate other scalability aspects. Successfully designing and building such a system is an enormous task that requires coordination and the adoption of new paradigms. Co-design centers and the HPC community as a group may play an important role in making this endeavor a success for all.

3.3 The programmer's burden

Leads

Session lead *Ron Brightwell*, Sandia National Laboratories.
Wingmate: *Turlough Downes*, Dublin City University.

Description

> What high-level changes are going to be required for applications to reach exascale? How important will communication avoidance be? Will programmer awareness of power indexpower!programmer awareness and reliability be required? To what level of detail? Will Bulk Synchronous Programming (BSP) survive? Must it?

Presentation

Ron's first two slides humorously proclaim that application developers are evil. He describes a scene where he offers a new OS that improves scalability and performance. The application developers then complain that they do not want to change their makefiles, use a cross compiler, make hardware-specific optimizations and request that the new OS [or way of doing things] has to improve performance on all machines and everything has to be portable. In act two, the application developers come back excitedly asking Ron for his help with a new thing. They explain that it is a custom piece of hardware, requires a cross compiler, that they have to change their makefiles, and the new code is not longer portable.

> "Application developers are evil." (*Ron Brightwell*)

Ron's point is that, unless we are willing to ignore application developers, they should drive the requirements for the OS and runtime. In light of that, he uses the rest of his presentation to ask specific questions that need to be answered before an exascale system can successfully run the applications it was built for.

Discussion

Ron's first question is what high-level changes will be necessary for applications to reach exascale. *Larry* interjects "No BSP!" because with that many parallel threads we need asynchrony. *Sudhakar* counters that BSP [37] provides a structure to manage millions of threads. If not BSP, then what? *Ron* restates that we cannot handle global synchronization at that scale and asks at what level synchronization should be done then.

Alexey suggests to concentrate on communication for exascale. It seems people are not too worried about inter-node communication. *Jeff* says we will have about 100,000 nodes, with about 1,000 to 10,000 threads each. He asks whether that is still the anticipation for the first exascale systems, and states that the number of nodes in that case will stay about the same as we have in current high-end systems.

Simon adds that the node model will be hierarchical: groups of threads working together, and super groups that communicate off-node. *Aidan* suggests to grow the lowest level of such a hierarchy from petascale to exascale, and change the problems to work on exascale; i.e., perform more complicated calculations. *Vladimir* says that is two to three orders of magnitude and asks whether we need to support legacy applications. He believes that exascale creates a niche within HPC and wonders how many people can possibly work on that.

"No BSP!" (*Larry Kaplan*)

Ron's next prepared question is how important communication avoidance will be. *Aidan* thinks it is very important. To achieve it, lower layers in the parallel algorithms need to be adapted. *Ron* says that, in addition to performance optimization, other people want lower message rates to save power and that it could help with resilience. *Arun Rodrigues* says that it is currently not possible to reduce power consumption of the network due to the BSP model and that systems need to be configured for maximum needs.

Turlough Downes mentions that many physics applications are self synchronizing and communication avoidance may not be that easy. *Arun* suggests to move toward a data flow model and that it might be possible to do underneath MPI, by transmitting partial buffers. *Barney* warns that dropping message passing semantics and moving to a data flow model is a major change. *Arun* explains that it might be possible to do some of it at lower layers and that we could also change the programming model.

If transmitting partial buffers, then how would we deal with non-blocking sends, *Larry* asks. *Arun* answers that it can be done on the receive side, but *Barney* points out that the buffers are needed for retransmission during error recovery and, therefore, we need to change the programming model.

Ron's next prepared question is whether programmers need to become aware of power and reliability in future systems. *Simon* thinks applications do not really care about power. Programmers may be willing to declare "robust" variables to indicate which regions of memory may need better protection from faults. That is because application people understand the concept of errors, but have no notion of how they could save power.

Turlough wants to know more specifically what *Ron* meant with his question and what usage model it applies to. Are we talking about applications using the entire system, or lots of smaller applications running on a large system? He believes we cannot have a general-purpose exascale system. People using it will need to know that they are on an exascale system. In the future, people will be used to petascale systems and will write exascale-aware applications.

Ron mentions mobile applications as an example of power aware programs. They minimize access to the GPS in order to save power. *Sudip* says that manufacturers force applications developers' hands. *Aidan* says that HPC application developers foremost want correct answers and good performance. *Arun* suggests to provide an incentive for power-savvy jobs, for example by giving them a higher priority in the job queue. *Ron* is pessimistic. He says the ASC compute facilities do not charge for performance problems.

Going back to applications declaring “robust” variables, *Thomas Sterling* reminds us that faults are not restricted to a variable’s content; its address is just as important and may get corrupted too. This prompts *Ron* to wonder about the vulnerability of the OS and runtime. *Hermann* says checkpoints need to become more efficient, but *Simon* thinks we need something better than checkpoint/restart.

Ron moves the discussion along by asking his prepared question of why we have the expectation to develop on a laptop and then run at exascale. He skips the questions whether BSP will survive and whether we really want to maintain the expectation of performance, since we already talked about BSP and are beginning to run out of time for this session.

Ron doubts that developing on a laptop makes sense. *Sudip* counters that people have to deal with parallelism on today’s personal computers (PC) that use multicore processors. *Ron* says that is not the same. The exascale issues of scaling, power, and reliability are not present.

Barney chimes in that it is a mixed blessing. More students are now exposed to parallel programming environments, but they also have the expectation of a full-featured OS and runtime system which are not present in the same way on large-scale systems. It is easier to scale down than up. The reason is that we have not defined a limitation model that would explain why something does not scale up. *Sudip* says that people want to do science not computer science (CS).

Jeff states that parallel programming on individual systems is driven by business, such as gaming, and education. He doubts that there are masses of programmers who work on laptops and then expect their codes to scale to exascale machines.

Sudip mentions this is akin to climate application developers who had to move from vector machines to massively parallel processors (MPP). According to *Sudip*, some of them are still bitter. *Henry Tufo* explains that the reason is that there was more science in the vector codes, and that some of that has not made the transitions to MPPs, even though the MPP codes are more scalable.

Ron has a long list of questions remaining in his presentation, but we are running out of time. The last one briefly discussed is whether programmer productivity is really an issue. *Turlough* states that it is “publish or die.” Therefore, codes have to be developed quickly. High productivity parallel languages and development systems, for example X10 [11] are needed.

Summary

The needs of application developers are continuously changing. While they wish for common and performance preserving APIs and standards, they also need to make use of the latest technological advances. Currently this means using accelerators and be willing to dive deep into new technologies and have different versions of an application for the various systems in existence. It also means dealing with millions of threads in extreme-scale systems. Synchronization will be a major issue. Fault tolerance and data movement will be very important as well and at least some of it will have to be done with application guidance.

Because application developers are, from the point of systems developers’ perspective, end users, the application developers should drive the requirements for lower level systems software such as the OS and runtime. In this session, again, the topic of educating the next generation – in this case application developers – is discussed.

3.4 Research challenges

Leads

Session lead: *Barney Maccabe*, Oak Ridge National Laboratory.
Wingmate: *Vladimir Getov*, University of Westminster.

Description

> How can research help address challenges expected to arise with the advent of exascale systems? Predicted issues, such as fault tolerance, power dissipation, usability, programmability, and scalability are hot topics in research laboratories. Are there issues not being addressed? Is progress in these areas advancing fast enough?

Presentation

In his presentation, *Barney* proclaims we need a revolution to address the daunting exascale challenges ahead of us. Not only that, but we are due for one since the last one – "Attack of the Killer Micros" [10] – was in the late 1980s. *Barney* lists four areas that need a revolution: Power, parallelism, memory hierarchy, and resilience. For power, *Barney* says it needs to be managed at all levels from the machine room to the OS, to the runtime, to the application. The amount of available parallelism needs to be increased by increasing application asynchrony. Deep memory hierarchies will require explicit management. Data movement across that hierarchy dominates cost and consideration of it needs to become more important than counting (floating point) operations. For resilience, we need to be aware that in an exascale machine, some part will always be in a failed or degraded state.

In June 1993, four years after Eugene Brooks' warning about the imminent attack of the killer micros, the first Top500 list [25] shows that the revolution was over.

> "We all know the answer; some of us may even be right." (*Barney Maccabe*)

Barney stirs up the discussion by showing a slide summarizing some predictions from the 1994 petaflops computing meeting in Pasadena [24, 35]. Some of those early prophecies were that a petascale system would need ten million processors, would fail every few minutes, be enormous in its physical size, and cost about 25 billion Dollars. Even though the perceived challenges were huge, a petaflops was achieved in 2008, and according to *Barney*, without an intervening revolution. Contradicting his earlier statement, maybe no revolution is needed to reach the next level of computing although today's predictions for exascale systems mirror those made in 1994.

An area where revolutions have been prescribed in order to improve power output and fuel efficiency, is the internal combustion engine. It is still around and performing at levels never predicted in the 1970s. *Barney* compares this to the x86 instruction set and the permanence of Fortran. If a revolution is needed to reach exascale, it will be difficult to ignite and overcome inertia. *Barney's* recipe is to get it started at the bottom. People in power want to protect the status quo. To create a revolution, the value of an alternative needs to be demonstrated and a "hungry community" needs to be incited with a few well placed bets. An investment in tools can soften the impact of an impending change.

Finishing his presentation, *Barney* charges the participants to address three questions: How can research help address the expected exascale challenges? Are the predicted issues addressed in current research efforts? And, is progress in these areas advancing fast enough?

Discussion

Thomas Sterling forcefully protests *Barney's* assertion that we had no revolution. *Thomas* lists the move from strong scaling to weak scaling as one example and accuses *Barney* of corrupting our intellectual thinking by glossing over such important changes. *Barney* responds that that was only a change in the definition of success. *Thomas* counters that rather than ending in 1994, that was the year the revolution started. We moved to MPI/C and Beowulf clusters began to appear.

"This corrupts our intellectual thinking." (*Thomas Sterling*)

Barney insists that it had been over in 1994. It may not have permeated the whole world then, but it was nevertheless over. He asks the following questions: If we are in a revolution, when will it end? Why are we not standardizing architectures? Why are we not redefining success?

"Once those in power join the revolution, the revolution is over." (*Barney Maccabe*)

So, what is the goal? To create a revolution? *Barney* asserts that we are not likely to start a revolution. Other people with a need, will. But, they are not. *Simon* points out that there are people in need, but GPU are not a solution. *Barney* says they are stuck; they need an innovation; a disruptive technology. *Jeff* wonders why they cannot use accelerators. *Simon* responds that accelerators are not good for sparse, multi-physics codes. *Jeff* counters that dozens of codes scale, which makes *Barney* wonder whether these are codes that never made the transition off vector machines. This brings *Barney* back to suggesting that the move from vector machines to MPPs happened because there was a "hungry" community of people in need of something other than the available vector machines and that a bet is in order; i.e., something like the killer micros bet that led a few adventurous laboratories try massive parallelism, to work around the scaling issues at the time.

The lively discussion continues when *Barney* starts going down the three questions he presented to the participants at the end of his talk. If research were to "place bets" to incubate a revolution, where should we start? We cannot chase everything. *Barney* suggests a "structured" approach to the revolution. *Arun* suggests to throw seeds in an organized fashion. Some will take; but most wont. *Barney* says that we have a technology change [accelerators], but nothing disruptive.

Vladimir thinks that the problem may be a too US-centric view. China, Brazil, and Europe have different applications and may have different needs. The current drive toward exascale is missing a lot of people and applications. He points at the Exascale Chat [34] Host Simon held at the International Supercomputing Conference (ISC) earlier this year. That panel was very international and had a very different perspective.

Barney mentions how Nancy Lynch counted messages in distributed systems and was able to gain new insights that way. He brainstorms energy consumption. The effect is indirect: data movement requires energy. Maybe we should count memory operations, since that is what uses power.

Sudhakar says that HPC moves with the markets. HPC innovations will be market driven. *Barney* says we are still in a hall of mirrors. We are just riding along instead of inventing HPC specific processors. There was some HPC innovation for interconnects, but not much. *Sudhakar* steers the discussion to asymmetric processors and multicore processors with different cores and asks how they will change architecture. Both will increase transistor count, but the question is where to place the bets. But *Barney* warns that we cannot cry wolf too many times. It better be a real revolution, otherwise we should not call it that.

Alexey lists a couple of instances that caused revolutions. We used to have highly specialized computing centers that used a couple of programming languages and were operated by nerds. Then the PC came along. Revolutions happen when systems become commodity. Another example is the telephone companies laying a lot of optical fiber cables. This has enabled innovations such as the Grid'5000 [17, 8] distributed experiment infrastructure. Revolutions happen when a new technology becomes widely available and a lot of people start thinking about how to use it in new ways.

Turlough points out that specialized hardware is very difficult to get. We are much more likely to have a revolution if it involves commodity hardware. *Henry* points out that this is what Nvidia has just done: it triggered a trend in HPC with commodity hardware. Nvidia identified a usage model and capitalized on that. Why can we not capitalize on this trend? *Simon* states that we cannot use a single vendor, but *Jeff* interrupts that we have three: Intel's MIC, Nvidia, and AMD. The problem is that these devices are "still sitting outside"; i.e., these accelerators are not on the memory bus. *Jeff* also laments that it may be too late for a revolution.

"It's too late, we don't have time to have a revolution anymore!" (*Jeffrey Nichols*)

Henry is more optimistic. In the past we used commodity components and modified them for our needs. This may be the first time we can dictate what goes into the next processor. *Arun* says we have done that before, but *Barney* states that others; e.g., the mobile phone makers, have more sway than we do. *Henry* tells us to take on a more active role, which brings *Barney* back to the question which investments we can make that will pass the test of time. The next technology refresh will wipe out investments in current software and will make it hard to use what is out there.

Summary

Four areas in particular will need to undergo radical changes: Power, parallelism, memory hierarchy, and resilience. All layers of the software and hardware stack will need to be involved. Applications will have to help manage deep memory hierarchies and control data movement.

There is inertia in changing how application manage resources. Investments in a few key technologies that show clear performance and efficiency gains, may trigger the needed changes. A disruptive change is needed to bridge the gap to exascale. The difficult question is what research needs to be funded that can ignite such a change.

3.5 Exascale simulations

Leads

Session lead: *Sudhakar Yalamanchili*, Georgia Institute of Technology.
Wingmate: *Arun Rodrigues*, Sandia National Laboratories.

Description

Discuss the need and challenges of simulating exascale systems.

Presentation

Sudhakar begins his presentation by stating that the problem of simulating an exascale system is too large for a monolithic solution. The problem is that full-system complexity is growing exponentially, while simulation capacity is only growing linearly. Mature models and simulators exist, but they are usually monolithic, custom, and not built for composition or re-use.

Four factors contribute to the immensity of the problem. Scale is the obvious one with the number of nodes, cores, and threads being too numerous to simulate with great fidelity. Multiple audiences have different requirements, from purchasers to application writers to systems engineers analyzing network behavior and processor utilization. The size of exascale systems increases complexity. Multi-physics applications, OS effects, and new languages are examples. Finally, there are constraints that have to be dealt with: performance, cost, power, reliability, cooling, usability, risk, and size.

Further complicating things is that different people use terms like models, simulation, and emulation differently. Drawing different options on a graph with evaluation quality on the x axis, and simulation scope/parallelism on the y axis, *Sudhakar* shows that there is a spectrum of solutions to consider which vary with their suitability to produce desired information. In general, the trade off between fidelity and scale is the driving factor.

Because of all this, *Sudhakar* points out that a community effort is needed. A single monolithic solution cannot solve the problem. The efforts of different groups need to be brought together to compose new tools and work on multiple levels of abstraction. It is also necessary that different groups of researchers, such as application developers, architects, and system software experts provide input and help working toward a solution. This is where a common vocabulary becomes important. Otherwise the different groups will not understand each other.

Another issue *Sudhakar* mentions is IP. Some simulations require detailed product information that vendors are reluctant to make public. Is there a way to create modules, maybe by the vendors themselves, that create valid output while protecting IP? An example of where such collaboration is needed, is the energy, power, and resilience stack. In order to determine where energy is dissipated, the entire software stack, including the hardware, has to be evaluated. Some of the information necessary to do that can only come from vendors.

The challenge is to understand multiple interacting physical phenomena with the goal to create a modeling environment that lets us understand architectural impact and evolve a co-design methodology to optimize systems. *Sudhakar* has two example slides: one showing the interaction between reliability and thermal fields and the other showing the interaction between cooling and performance.

At the end of his talk, *Sudhakar* summarizes the challenges and opportunities he has presented. Two that stand out are model accuracy and validation, and the difficulty of engineering a simulator. His concluding slide has specific questions, based on his presentation so far, for the symposium participants. The ensuing discussion centers on three of these: Does co-design require special attention? What do application developers and system software researchers need? How can we work with industry?

Discussion

Jeff starts the discussion with the question whether the co-design centers use simulators. *Simon* says yes, but there are not enough, accurate tools. Open simulators without proprietary IP embedded have less accurate timing. *Sudhakar* warns that this will have impact on the accuracy of power simulations and wonders what the impact on the OS might be.

Barney is not sure we need to simulate the OS. He wants applications or runtime systems to get direct access to the communication hardware. *Sudhakar* asks what kind of tools the simulation community could supply to help OS research. *Barney* says to do things like cache injection [23, 22]. *Sudhakar* points out that the way some simulators are built makes it difficult to provide fine-grained access to the scratch pad and simultaneously abstract the rest of the runtime.

Kurt says that systems software does not use simulation very much. Local issues can be done on a node. There is no need yet for simulation. However, *Vladimir* mentions that Blue Gene used simulation extensively; e.g., Mambo [7]. *Arun* distinguishes between design tools and bring-up tools. Mambo is powerful, but difficult to use and modify. Validation is different.

Dimitrios wonders whether sampling would allow us to run full applications and whether the resulting accuracy would give us more confidence.

Christian points out that there is no simulator for dynamic runtime environments at scale. *Sudhakar* asks whether application skeletons, plus a runtime, plus hardware simulation could do that. *Arun* asks *Thomas Sterling* about using simulators for runtime systems. *Thomas* responds by saying that Adolfy Hoisie has high-level models and states that we need to know event counts, not time variance across billions of threads. We can extrapolate with a small number of key parameters. Use of queuing models is also an option. He jokingly adds that they are sufficiently insincere, so there is no real disappointment, but they can be useful. *Kostas* remarks that feeding a queuing model requires a distribution function. *Thomas* says that we do not need to find a closed form solution.

Sudhakar asks about interaction with industry. Would it be possible to have models that cannot be released. The issue is proprietary information that vendors may not want to release. Having a sealed module that could be inserted into an open simulation tool, might be an option. He lists memory controllers as an example of devices that have a lot of "secret sauce" and are therefore difficult to simulate. However, a model of a memory controller may not be that hard to create and may still be useful. *Arun* asks what kind of models would industry accept or believe in, instead of saying that they are not accurate enough?

Summary

Simulating exascale systems is difficult because the number of elements to be simulated grows exponentially, while simulation capability grows linearly. Different users of simulators have different requirements that cannot be all satisfied by a monolithic simulator. Modularity is required to tune the accuracy for specific parts of the simulation to the needs of a given experiment. Allowing vendor IP to be incorporated into a generally available simulator is necessary to achieve accurate power readings. However, there has to be a way to protect vendors' interests, or the necessary proprietary information or modules will not become available.

3.6 Co-design

Leads

Session lead: *Sudip Dosanjh*, LBNL/NERSC.
Wingmates: *Aidan Thompson* and *Simon Hammond*, Sandia National Laboratories.

Description

> What has been learned so far? Is it working? What are the implication for system software? How far down the software stack should/can co-design go?

Presentation

Simon begins the two part presentation with an overview of the co-design efforts at Sandia National Laboratories. He says that the interaction of system software, specialized hardware, algorithms, and dealing with different vendors makes programing and designing HPC systems a very complex endeavor. However, that situation is not new: embedded systems have had similar problems and dealt with it using a technique called co-design [33].

The USA DOE is funding four co-design centers with the goal to influence next-generation applications and architectures [29]. There are also efforts under way to build a simulation infrastructure, and several X-Stack projects [30] will help shape the future software landscape. In addition, vendors are involved through matching funds FastForward projects and procurements.

Initial efforts at Sandia National Laboratories concentrate on creating a set of so called mini-apps [5] that help evaluate hardware architectures and are suitable for running inside a simulator. The number of mini-apps is increasing so they cover a larger spectrum of application types. At the moment, code vectorization is a big issue.

Aidan presents the second part, concentrating on molecular dynamics (MD). With the increased parallelism available in exascale systems, it is important, and not easy, to extract more parallelism from molecular dynamics applications running on a wide variety of systems. A lot of recent MD demonstrations of extreme-scale systems have achieved parallelism by scaling up the atom count with the node count. This implies constructing MD simulations with billions of atoms, with millions of atoms per node, interacting according to very simplified models. However, the usefulness of MD is limited by accuracy, not atom-count. Simulating more atoms may not advance science. The best MD simulations typically consist of thousands to several millions of atoms interacting according to detailed models. Hybrid parallel approaches are required to achieve much better strong scaling, to the point of having only a few very expensive atoms per node.

“Simulating more atoms may not advance science.” (*Aidan Thompson*)

While co-design is not new, the hope is that the current focus on it will help influence hardware development and further integration across the software stack to make getting to exascale easier. *Aidan* stresses that getting there will be hard and that reaching exascale for legacy codes and algorithms will be even harder. He then poses four questions to all participants:

1. What do you need from the co-design process?
2. Are you designing or are you co-designing?
3. What can you give us to help?
4. What is the most important place for us to start?

Discussion

Thomas Sterling initiates the discussion by stating that we need a better understanding of the mini-apps in order to rewrite them. Mini-apps are not benchmarks and are meant to evolve over time to adapt to new architectures and runtime environments. *Thomas* also requests that information about points of contact at each of the co-design centers be made public. This is important for other efforts, such as runtime system design, to learn from the centers and, in turn, to influence them.

Jeff is concerned about the metric for success of the co-design centers. He suggests that success could be measured by the number of architectural changes based on requirements identified by the co-design centers. But *Jeff* states that there is currently no information flow from the co-design centers to HPC companies.

Alexey wants to know how co-design is supposed to work. He asks whether algorithms need to change. *Simon* answers that computer science people make the changes and assess performance. Application (algorithm) specialists then comment on these changes and may throw them out. The goal is to find the best combination of program workload and architecture.

Sudip goes into more detail. He says that mini-apps are a mode of communication. They should not be simply posted on a web site to be consumed. Feedback is encouraged and needed. Instead of specifying a bandwidth or flops goals, the mini-apps provide the requirements [26][1]. They exemplify the problem to be solved. Feedback on how that can be done is part of the co-design process.

[1]The mini-apps are part of the draft of the technical requirements for the Trinity (LANL) and NERSC-8 platforms.

A big concern is the performance of vector operations. Algorithms and code needs to be optimized for modern and future architectures. To date, *Sudip* says, a lot of NDAs have been signed, but it is a slow process to affect product changes. In the end it may not be DOE that forces change but the market, through a company like Nvidia trying to adjust to new market demands. *Alexey* interjects that Nvidia is indeed interested in designing a streaming architecture for analytics and big data.

Sudip says that the DOE and the HPC community will not change the main course of companies like Nvidia, but there is the possibility to have some impact. One example is the DOE FastForward program. It provides an opportunity for vendors to add their own versions of mini-apps to the mix. In addition, DOE has big data applications as well.

Thomas Sterling wonders how much of this effort and money is for exascale versus a ten petaflops system. *Simon* says that for 2018, the expectation is that vendors will provide a programming model that includes MPI. This prompts *Thomas* to ask whether that is official policy, to which *Sudip* replies it is not. *Larry* asks what is stated in the RFI.

Vladimir says that petaflops systems will evolve into exascale systems. The border between them is somewhat political and subjective. How do we define an exascale code? *Sudip* answers that the six mini-apps have been vetted to be representative exascale applications. The definition of performance in the RFI was the Linpack benchmark but also an improvement of 100 to 1,000 for production codes. Although the 1,000 number will probably not be achievable in the first round.

Sudip comments that this is better than it has been in the past. This is the first time that DOE is committed to applications. *Simon* adds that MPI needs to be provided and working well for legacy applications. However, a future machine may also support other programming models in addition. *Thomas* is shocked by this development. *Vladimir* suggests to co-design applications such that they will work with a new programming model.

Summary

The co-design centers and related efforts promise a more application and result oriented approach to future system design and acquisitions. However, concern exists that these efforts are not moving fast enough, target systems and problems are less than exascale, and may not have the necessary impact with vendors.

3.7 Expanding the scope of traditional HPC systems

Leads

Session lead: *Duncan Roweth*, Cray Inc.
Wingmate: *Rolf Riesen*, IBM Research.

Description

Traditional large-scale scientific applications are not enough to drive the market. New areas, such as analytics, may be served by exascale systems or smaller systems using exascale technologies. In which market areas can exascale-capable machines play a role? What compromises have to be made to enable this broader-use spectrum? What technologies, not specifically designed for supercomputing, can be leveraged to reach an exaflops sooner, cheaper? Vice versa, how can HPC technologies and methods help the larger data center/cloud space?

Presentation

Duncan begins the presentation by reminding us that there will be a small number of exascale systems and a "reasonable" number of smaller HPC systems, but the overall market size is small. Which brings us to the core question for this session: can the same technology be used in a wider market[2] and is it applicable? Traditional HPC tries very hard to minimize data movement because of its various costs; e.g., time and power. However, data movement dominates many application areas. One example of concern is that a NIC with a broad injection bandwidth range may be too expensive for the common market.

The CPUs used in HPC systems are the same ones available for other uses. The bytes per flop to memory bandwidth ratio and its implication is reasonably well understood in the HPC community, but varies widely. Global system bandwidth impact is less well understood but is important because it has a large impact on cost. It is also a parameter that varies widely from system to system.

The cost function for a high global bandwidth network contains some factors that can be chosen, and others that are market or physics driven. Using optical interconnects is currently expensive. This results in a reduction in the injection-to-global-bandwidth ratio, or a reduction in injection bandwidth. In other words, we need to wait for another step change in the cost of optical networks before HPC can become cost-effective for more general uses.

System characteristics, such as injection bandwidth per node, number of fully connected nodes (the group size), the global bandwidth to injection ratio, and the storage hierarchy, determine what type of application a system is suitable for. *Duncan* shows a scale with weighted average number of peers starting at 1 on the left and going beyond 5,000 on the right. Finite difference, finite element, and most ASC codes (ACES) are located on the left-hand side. Spectral weather and climate codes reside in the middle of that line, and graph and sorting applications are to the far right. New analytics applications also fall onto the right side with their all-to-all communication patterns and connectivity among peers. *Thomas Sterling's* asks how sensitive these numbers are and how they change when capacity changes. *Duncan* says that as applications move from weak scaling to strong scaling, they use less memory per node and that forces them to move to the right on the scale because they have to interact with more peers now.

An example application is neuroscience simulation with the goal to enable targeted drug development and bringing down cost. There are many different brain diseases with different biological causes but very similar symptoms. This makes is expensive and difficult to select patients for clinical trials and identify specific drug targets. The European human brain project [15] aims to address this problem by using compute power to make pharmacological research profitable again.

Some of the work to simulate pieces of the brain is beginning now on existing systems. As the complexity increases, memory requirements and operation count will go up. It is expected that a cellular simulation of a human brain will require an exascale machine with 100 petabytes of memory.

As a second example, *Duncan* mentions difficult MapReduce problems. Although MapReduce operations are a mainstay of today's data centers and large clusters, some MapReduce workloads need a more capable network during their reduce phase[36]. Other research has shown that high-end HPC systems are very capable of running such high-demand MapReduce workloads [32].

Duncan then initiates the discussion with a statement and two questions.

1. Our requirements are broader than those met by traditional HPC systems. Often, sparse problems do not fit well into a steep memory hierarchy.

2. What can the HPC community gain from work to expand the scope of the systems that we use?

[2]Or vice versa, can more wide-spread technology be used for supercomputing?

3. How broad a set of applications do we need to run well on an exascale system?

Discussion

Larry starts the discussion with a question about *Duncan's* slide that shows applications along a line of weighted average number of peers. He wonders whether applications on the far right, those communicating with a lot of different peers, are important enough to push HPC there. *Thomas Schulthess* says this view of applications is too static. We need to look at different ways of solving a given problem. Regarding data, he states that it makes no sense to allow access to all data at the nanosecond time scale.

Duncan says it takes many years to "move" a code, but *Thomas* asserts it can be done on the order of one year. He further clarifies that it is the lifetime of the model that matters, not the lifetime of any application implementing that model. But *Henry* interjects that codes live forever. *Thomas* says that is the fault of CS: We have fifty-year-old computational models that are too abstract.

Henry says that in physics there are theoreticians and experimentalists. In CS we do things differently. *Thomas* replies that we need to change that: writing code is not that expensive. *Larry* thinks it is a cultural problem. *Aidan* contradicts him. It is not cultural; moving codes to the left of *Duncan's* scale is a hard problem. He adds that validation is also very hard.

Thomas says that in physics experimentalists get trained to build and evaluate machines (scientific tools). By contrast, CS students are not trained like that; not even math is required! The codes they produce are a mess and cannot be maintained. That is why these codes are not amiable to change. In support, *Henry* comments that professors do not get paid to build codes. They need to publish papers.

This prompts *Barney* to state that we are educating our students in the wrong manner. *Rolf* thinks Java is to blame because it allows programs to be written without understanding the underlying machine. *Alexey* says it is too easy to write games and phone applications. We should take students form physics and mathematics as [CS] Ph.D. students.

"You are educating your students wrong." (*Barney Maccabe*)

Sudhakar brings up the raw cost of writing a million-line application. *Thomas* says that laboratory equipment does not last thirty years and our applications should not either. The models underlying these applications do. *Hermann* mentions that it is impossible to get a paper accepted at an OS conference without clean experiments and a thorough evaluation. *Alexey* gives as an example that there is no course that teaches memory hierarchy and how to manage it. *Thomas* says the fundamental problem is that CS (based on CE) is in the area of discrete mathematics, while computational science deals with continuous mathematics.

Vladimir tries to get the discussion back to the topic of this session. He asks what *Duncan* meant when he said that global bandwidth is less well understood. He says that the "fatness" of nodes determines the number of them needed to solve a particular problem. *Thomas* says that global bandwidth is a problem because the natural topology of a data set does not match the network topology of the underlying machine. *Duncan* adds that mapping (and faults) are the problem. *Thomas* reaffirms that we need a way to lay out data and processes in a machine so that the physics problem to be solved matches that topology. *Duncan* says that Cray Inc. has done some work along those lines with Sandia National Laboratories. We need a runtime that allows us to express and request this mapping.

Summary

There is no consensus on whether a single type of system could both address the commercial data center market, yet still be suitable to solve demanding scientific problems. It is clear that some problems; e.g., number of communicating peers and globally available bandwidth are difficult and need to be addressed.

However, it may be that approaching the physics problems to be solved from a different angle may go a long way in producing applications that work better on these envisioned systems. Education of computer science and computational science students is seen as important toward that goal.

3.8 Application perspective

Leads

Session lead: *Thomas Schulthess*, Swiss National Supercomputing Center (CSCS).
Wingmate: *Henry Tufo*, University of Colorado at Boulder.

Description

> Are hybrid systems here to stay? If so, how will they be programmed? Physical constraints of computer hardware are forcing a rethinking of our programming models. Which models are finding acceptance among scientific programmers and how can they contribute to the design of future exascale supercomputing systems?

Presentation

Thomas tells the audience that some materials applications at Oak Ridge National Laboratory have shown performance at well above a petaflops. These codes and problems are expected to reach exascale. But there are others that have not even reached a 100 teraflops in performance. The key is what *Thomas* calls the *arithmetic intensity of computation.* It is the ratio of the number of floating point operations performed over the amount of data transferred.

He groups applications into algorithmic motifs from low intensity sparse linear algebra, matrix-vector, vector-vector codes to high arithmetic intensity dense matrix-matrix codes. *Thomas* says that claiming supercomputers are general purpose machines and are also HPC where performance matters, is a contradiction.

As an example of the problem, *Thomas* talks about COSMO [31] a weather prediction code that is in production use in Switzerland. An older version of it concentrated 80% of its performance in a small section of the code. The physics part is compute bound, while the dynamics are memory bound. He shows a code example from the physics section that performs 136 flops using 3 memory accesses, and another from the dynamics sections that only does 5 flops for every 3 memory accesses.

In order to improve the performance of this important application, it had to be changed to employ bandwidth saving strategies such as computation on the fly and increasing its data locality. But an adaption of the hardware was also necessary to provide more memory bandwidth; i.e., adding GPU hardware. To take advantage of this hardware and to hide the complexity of the optimizations, the core of COSMO was rewritten in C++ using templates to allow mapping it to CPUs or GPUs. While running on the GPUs brought a large speedup, transferring the data back and forth to main memory between the physics and dynamics phase negated all performance gains.

The solution was to use scratchpad memory inside the GPU and leave the data there. *Thomas Sterling* asks whether this approach generalizes. *Thomas Schulthess* answers that twelve applications have used this approach so far and then expands on the things we need today and certainly for exascale. Increasing data locality is a key factor. We need a language and a mathematical model to express locality. Scalability has to be addressed from the ground up.

We also need a programming model and environment for distributed memory machines with complex nodes. Modern nodes can no longer be described using the von Neumann model. In the 90s using distributed memory was difficult. MPI solved that problem. Now we need something like that for the architectures we have today. *Thomas'* last slide shows the progression in performance over the last twenty five years using as examples the yearly Gordon Bell Prize winners and the programming models they used. An extrapolation to exascale predicts that we will reach exaflops performance in 2018. *Thomas* warns that if the HPC community does not do anything, the programming model will include CUDA.

> "We will have one exaflops in a real application in 2018. You will end up with CUDA. Please, do something!" (*Thomas Schulthess*)

Discussion

Thomas' plea to provide something better than CUDA starts the discussion. *Henry* asks whether *Thomas* really thinks that we will get rid of MPI by 2018. *Thomas* counters that Chapel is not it. He expands on his ideas. For a funding model he suggests to give the money to the weather service, but make them hire our [HPC] people to optimize the codes. A longer exchange between *Thomas* and *Henry* lets *Thomas* predict that with the proper optimizations an exaflops could be used for climate modeling over a 10 km^2 area over 100 years, and a 2 m^2 resolution. Then *Larry* and *Thomas* discuss programming languages with the agreement that the language or a library needs to map the data to memory.

Vladimir asks whether this climate code can run at exascale. *Thomas* answers probably not, and that it may need more refactoring. The goal is to run a global simulation at 1 km^2 resolution for 100 years. A petaflops machine has enough message passing performance for that, but not enough memory. Once ocean cooling and such are enabled in the simulation, an exascale machine is required.

Henry makes a distinction between weather and climate codes. Some of the [funding] things that work now for weather codes are not done in climate codes. Weather is local and states are willing and able to support it, but climate is global. *Thomas* says it was not always like that for weather and the approaches that have been successful can be applied to climate.

Summary

Some applications are very suitable for scaling to an exaflops machine, while others face much larger hurdles. Programming languages and models that hide some of the complexity of a modern supercomputer, yet allow the expression of data locality, are needed but not really in sight yet. Bringing less scalable codes to exascale will require careful analysis and refactoring of these codes. All of this depends on funding. While there is funding available for problems like weather where execution speed and accuracy have an immediate impact, funding for longer term research, such as climate simulation, is more difficult to arrange.

3.9 Fault tolerance

Leads

Session lead: *Christian Engelmann*, Oak Ridge National Laboratory.
Wingmate: *Larry Kaplan*, Cray Inc.

Description

> Permanent and transient faults may occur continuously in an exascale system due to decreased component reliability and increased component counts. What fault types and

frequencies should be expected? Can evolutionary fault tolerance approaches provide resilience at exascale, or are more revolutionary concepts needed? Which layer (OS, runtime, and/or application) is responsible for assuring resilience?

Presentation

Christian starts with a list of exascale resilience workshops that have been held and informs us that there will be a bird of a feather session at Supercomputing this year. It is clear that fault resilience of large scale systems is an important topic and many people are studying it. He then focuses on the US DOE's need in this area. Because the DOE uses mission-critical applications that require a high level of accuracy, have very long running times, and will utilize millions of threads, reliability is extremely important. At the same time, there is a trend toward less reliable Commercial Off The Shelf (COTS) components and a need for many more of them, since frequency scaling is giving way to core and node scaling.

In subsequent slides, *Christian* elaborates on the problem. It is expected that the number of soft errors in processor and memory devices will grow as their density increases. In a system with 1,000,000 devices manufactured at 11 nm, *Christian* calculates a Mean Time To Failure (MTTF) of 0.2 h, if no additional protective measures are taken. Maybe even more worrisome is that these same calculations also predict about two undetected errors each hour.

For an exascale system by 2020, 7 to 10 nm process technology will be used at near threshold voltages in order to meet energy usage requirements. Some estimates put an exascale system using such components at a five times higher risk for errors than the current Titan system at Oak Ridge National Laboratory. At this point *Ron* interjects that *Christian's* estimates are lower bounds, since they do not take file systems and components such as NVRAM, into account. There is also a discussion whether node Mean Time Between Failures (MTBF) will stay the same. Some vendors claim that it will improve, but it seems more likely that it will stay the same or even decrease. However, there is little room for improvement. Modern system no longer have disks or fans which in the past were the main culprits for faults.

For solutions, application-level checkpoint/restart to a parallel file system is the current standard. There are many advanced solutions to improve resilience, but apart from system-level and incremental/differential checkpoint/restart, none of them are used in production. Although local checkpointing techniques can be used in the short term, and more advanced techniques can be deployed in the future, a better understanding of the problem is necessary. This includes classification of errors, error rates, fault root causes, and error propagation. We need that information for current and future systems. *Larry* states that Cray Inc. is precluded from releasing failure data. *Kurt Ferreira* thinks there is data that is not being looked at. This prompts *Barney* to state that data [availability] is a red herring. He wants us to state requirements. *Arun* says we need to know what the root causes of failures are, whether they are in the network, ECC, etc. There are also cost trade-offs to consider that span power, resilience, performance, and deployment. Standard test suites and metrics are also needed to compare proposed solutions.

Christian's final slide holds these points for discussion:

- What fault types and frequencies should be expected?
- Can evolutionary fault tolerance approaches provide resilience, or are more revolutionary concepts needed?
- Which layer (hardware, OS, runtime, and/or application) is responsible for assuring resilience?
- What are the performance, resilience, power, and deployment cost trade-offs at exascale?

- Do we need standards for HPC resilience terms, metrics, methods, and APIs?
- Does the local OS need to be resilient (in addition to the global OS)?

Discussion

The discussion starts while *Christian* is still presenting. *Ron* continues by asking what Cray Inc. has done to improve reliability. Reliability was not part of the constraints when Red Storm was designed and delivered. *Ron* thinks it is possible to learn from these lessons for a next system, rather than gathering data from bad decisions. He mentions cheap motherboards as an example.

"Data is a red herring. What data do you need?" (*Barney Maccabe*)

Ron suggests to do the models first, in order to drive data. *Thomas Schulthess* turns the discussion to applications. He says that stochastic sampling applications have lived on flaky nodes for years, and that the OS and the communication layer can deal with flaky nodes, but MPI cannot. *Barney* contests that Monte Carlo may produce bad results if the failures are not random. *Thomas* thinks he can deal with that. He says it is necessary today because the machines are shared and we cannot be sure whether the state is truly random or not. The pseudo random number generators are always the worst problem. We cannot prove that they are "random enough." *Barney* remains unconvinced that *Thomas'* solution is enough, even if MPI was fixed.

The discussion turns back to fault tolerance in general with *Christian* stating that we need to talk about resilience more than performance. *Larry* agrees and adds power to the list of things to be considered. *Christian* makes the point by saying that wrong results of failed runs have no performance. *Kurt* states that we need standards; the current terminology in use is mind numbing.

Christian agrees that standards and MPI-level fault recovery are good, but warns that if recovery takes too long, it is useless. How should we measure quality of a fault resilience method? He wonders whether the local OS even needs to be resilient because the global OS already has to tolerate node failures. *Barney* wonders whether we even need bit-level correctness. *Larry* thinks it depends on the failure rate.

Kurt states that failures are not random and takes that as an indication that a large number of faults occur in the OS. *Larry* sees no reason to distinguish OS faults from node failures. *Ron* warns that things cannot just fail left and right; recovery cost matters. *Larry* adds that energy to solution is important too.

Ron clarifies that the MPI standard says nothing about processes and that is why it is difficult to add fault tolerance to MPI. *Larry* asks whether the MPI Forum will specify requirements for HPC. That may not work on clusters without a supervising system. *Ron* guesses that a runtime standard would help.

Summary

It seems there is a wide spectrum of ideas about the level at which fault resilience will be most effectively addressed, how to do it, and how much of it will actually be necessary. There seems to be consensus that the community needs to agree on metrics and methods to evaluate the reliability and performance under stress of whole systems. Depending on the actual impact of failures and the different ways of dealing with them, a range of options may provide the optimal solution for different types of systems and applications.

Even if the faults in future systems will not be as severe as some researchers currently anticipate, efforts to make fault resilience more efficient are not wasted. Writing fewer and smaller checkpoint files, for example, clearly lowers the administrative burden of a system and makes more compute cycles and data bandwidth available for applications. Taking a 10% performance hit due to faults may be acceptable, but being able to lower that to 5% or 1% still constitutes million Dollar savings.

3.10 Next steps

Leads

Session lead: *Jeffrey Nichols*, Oak Ridge National Laboratory.
Wingmate: *Thomas Sterling*, Indiana University.

Description

> We are on the road to exascale and have a slightly better idea of what these systems will look like than we did five years ago. Is ongoing research on track to make these systems scalable, less power hungry, usable for the intended application domains, and more resilient to faults? Where do we stand and are any course corrections necessary? How should we address the remaining challenges?

Presentation

The final session of the symposium is lead by *Jeff* who wants to make sure we are not missing anything. He asks whether we have a common vision and common goals, whether we are leveraging our investments and collaborations, and reminds us that educating the next generation is important. He also tells us that the nature of research has changed. A millennium ago it was experimental: natural phenomena were described and quantified using experiments. Only 500 years ago did we begin to formally describe our theories and express them using mathematical notations. In the last 50 years, computers helped us simulate complex phenomena, and today data itself has become a phenomena to be explored. It will take exascale systems to manipulate, process, and analyze the deluge of data. Without those kinds of instruments, we will not be able to understand the data puring in. Data-intensive science and its challenges have been studied in [1].

Thomas Schulthess interjects that data is not a fourth paradigm of science [16], it underlies the three other paradigms.

Future compute systems must support all four scientific discovery paradigms: experiment, theory, simulation, and data. Some scientific instruments today, such as the spallation neutron source at Oak Ridge National Laboratory have already generated a petabyte of data. Although a system like Titan has 700 TB of memory, processing all that data is difficult because it is distributed among 20,000 nodes. But *Jeff* warns that it is not enough to just prepare for (big) data storage and transport. There also has to be enough compute power available to process it. He then lists several examples of scientific applications that generate huge amounts of data but also need the corresponding computational power to carry out their simulations.

> "Data is not a fourth paradigm; it underlies the other three." (*Thomas Schulthess*)

Electrical power is also a problem. In November 2011, running at 2.3 petaflops, Jaguar, the precursor to Titan, used 7.0 MW and that does not include cooling. That is equivalent to 7,000 homes in a small city. Adding more traditional CPUs to reach 20 petaflops would have required 60 MW; enough for 60,000 homes. Instead, Titan employs GPU accelerators which allow for the ten-fold increase in peak performance by using only 8.2 MW.

To initiate the discussion, *Jeff* shows a slide refining his questions at the beginning of his talk. He asks: Is it feasible to achieve an exa-op/s by 20xx or 2020, using no more than 20 MW? Another GPU jump like the one observed upgrading Jaguar to Titan is not in sight. That means we will not be able to afford to power and cool an exascale system. Also, the system itself will be too expensive. His back of the envelope calculation shows that 200 - 400 cabinets will be needed, at a cost of $1M each. That means an exascale system will cost $400M, which is too much.

Jeff asks again whether we have common goals in hardware, software, co-design, applications, and system acquisitions? Does our time line and funding profile match up? Are we leveraging funding in the US by DOE, DOD, NSF, and industry, and internationally? He thinks we need another $200M of R&D funding each year for industry and the laboratories. And, we also need another $200M funding for applications.

Discussion

On the question whether we are leveraging our funding, *Jeff* uses IESP [13] as an example. He says a company like Cray Inc. will not let a laboratory or a university develop the next OS kernel because it is on the critical path. That means we have to leverage things that are going to happen anyway.

Education and the skill set of the next generation of students comes up several times during this session. *Jeff* says that the skill set of the people in this room would be very difficult to replace. This year Oak Ridge National Laboratory lost more people than they hired. This is the first time in over ten years.

Thomas Sterling reminds us that IBM was founded because of a big data problem [14]. It took nine years to complete a census that occurs every ten years. Automation was needed to deal with the growth of data. *Thomas* then went on saying that computers use data, but humans use knowledge. We need to convert data into knowledge. Visualization is an intermediate form, but computers do not understand visuals.

Thomas Sterling also informs us that the Chinese have three exascale programs, and they are merging all three into a single Instruction Set Architecture (ISA). If others follow that example, and that choice of ISA is wrong, we will be set back a decade or two.

This brings us back to the need for a solid HPC education program. *Thomas Schulthess* stresses again that data underlies all of science. It is not enough to just analyze data. *Barney* agrees. He says that the periodic table was not useful until we understood it. *Jeff* contends that there are multiple ways of looking at data science. It is not about data management, it is about information.

Thomas Schulthess says data alone is useless. We need knowledge. In order to get knowledge, we need mathematical models to extract knowledge from the data. Data alone is not it; we need to have a theory for data. *Jeff* thinks we can start looking at the data before we have a theory for it. *Barney* says we need to be able to make predictions.

Summary

It is clear that we have to deal with the data deluge and need to build affordable systems that can manage all that data. However, that alone is not enough. We have to find ways to create knowledge and understanding from the information contained in the data. In order to do that we need the education system to produce researchers who can understand the systems needed for this task but also understand the science needed to turn that data into knowledge.

4 Wrap-up

At the very end of the symposium we had a brief discussion about the symposium itself and whether it should be extended into a series.

One lesson learned was to start planning and send out invitations much earlier. The lead time for an international event like that was too short. Many people who would have made valuable additions were not able to attend on such short notice.

The discussion format for the sessions was a success and was appreciated by the attendees. It allowed for genuine discussions to ensue, and participants were much more likely to pay attention to the speakers and discussions, rather than getting absorbed in their laptops.

Sudhakar suggested that, maybe, longer sessions would allow for more in-depth coverage of a given topic. Both *Hermann Härtig* and *Sudhakar Yalamanchili* offered to host a second symposium. Several participants expressed an interest in holding it in Europe again.

Glossary

ACES Alliance for Computing at Extreme Scale, a joint partnership between Sandia National Laboratories and Los Alamos National Laboratory for the computing needs of the NNSA. 25

API Application Programming interface. The functions, semantic agreements, and conventions programmers use to access a library or system. 3, 8, 10–12, 14, 17, 29

ASC Advanced Simulation and Computing, a campaign by the NNSA to shift emphasis from test-based confidence to (computer) simulation-based confidence of nuclear weapons assessment and certification requirements. 16, 25

BSP Bulk synchronous parallel [37]. 15–17

CE Computer engineering. 26

co-design A development process where scientific problem requirements influence architecture, design, and technology of future systems. 3, 4, 11, 14, 15, 21–24, 31

COTS Commercial Off The Shelf. Production components that are readily available, as opposed to custom-made parts found in some supercomputers. 28

CS Computer Science. 17, 26

CSCS Swiss National Supercomputing Center. 6, 26

CUDA A parallel computing platform and programming model invented by NVIDIA for its graphics processing units (GPUs). 11, 27, 28

DOD Department of Defense. 31

DOE Department of Energy. 5, 13, 22–24, 28, 31

ECC Error Correcting Codes commonly used in memory devices to correct hardware faults. 29

exa Prefix for 10^{18}; e.g., a exaflops. 24, 27, 28, 31

flops Floating-point operation per second. 18, 23, 24, 27, 28, 31

GPU Graphics Processing Unit, used as compute accelerators in HPC systems. 12, 19, 27, 31

HPC High Performance Computing. 3, 10, 13–16, 19, 22–25, 27, 29, 30, 32

IAA Institute for advanced architectures and algorithms, a partnership between SNL and ORNL to address the architectural challenges in hardware and software of future systems. 14

IESP International Exascale Software Project, a consortium to address software challenges on the way to exascale. 5, 32

INRIA Institut National de Recherche en Informatique et en Automatique, a French national research institution. 6

IP Intellectual Property. 15, 21, 22

ISA Instruction Set Architecture. 32

ISC International Supercomputing Conference, held each Summer in Germany; every other Top500 list is announced here. 19

LANL Los Alamos National Laboratory, in New Mexico, USA. 23

LBNL Lawrence Berkeley National Laboratory (Berkeley Lab). 5, 6, 22

Linpack A benchmark performing numerical linear algebra; used to rank the computers on the Top500 list. 12, 24

MD Molecular Dynamics, a computer simulation of the physical movements of atoms and molecules. 22

MIC Intel's Many Integrated Core architecture, pronounced Mike. 13, 19

MPI Message Passing Interface, a standard API to transmit data in complex manners within a system. 10–12, 14, 16, 18, 24, 27, 28, 30

MPP Massively Parallel Processor. 17, 19

MTBF Mean Time Between Failures. 29

MTTF Mean Time To (next) Failure. 29

MW Mega Watt = 10^6 Watts. 14, 31

NDA Non-Disclosure Agreement. 23

NERSC National Energy Research Scientific Computing center. 5, 6, 22, 23

NIC Network Interconnect Controller. 24

NNSA National Nuclear Security Administration, an agency of the US DOE. 33

NSF National Science Foundation. 31

NVRAM Non-Volatile Random Access Memory. 29

OpenACC A programming standard for parallel computing on heterogeneous CPU/GPU systems. 11

OpenMP Open Multi-Processing, an API that supports multi-platform shared memory multiprocessing programming. 14

OS Operating System. 3, 10, 13–15, 17, 18, 20, 21, 26, 28–30, 32

PC Personal Computer. 17, 19

peta Prefix for 10^{15}; e.g., a petabyte. 18, 24, 25, 27, 28, 31

POSIX Portable Operating System Interface, a family of standards for OS, library, and shell compatibility among (mostly) Unix systems. 11

RFI Request for Information, often precedes a Request for Proposal (RFP). 24

SNL Sandia National Laboratories. 6

strong scaling The ability to run a fixed problem size on ever larger parallel systems. 18, 23, 25

TB Tera byte = 10^{12} bytes. 31

tera Prefix for 10^{12}; e.g., a teraflops. 27

weak scaling The ability to grow a problem size to make use of larger parallel systems. 18, 25

References

[1] J. Ahrens, B. Hendrickson, G. Long, S. Miller, R. Ross, and D. Williams. Data-intensive science in the US DOE: Case studies and future challenges. *Computing in Science Engineering*, 13(6):14–24, 2011.

[2] Saman Amarasinghe and et al. Exascale software study: Software challenges in extreme scale systems. http://users.ece.gatech.edu/mrichard/ExascaleComputingStudyReports/ECSS%20report%20101909.pdf, September 2009.

[3] Amnon Barak and Oren La'adan. The MOSIX multicomputer operating system for high performance cluster computing. *Future Gener. Comput. Syst.*, 13(4-5):361–372, 1998.

[4] Amnon Barak and Amnon Shiloh. MOSIX cluster operating system. http://www.mosix.org/, December 2012.

[5] Richard F. Barrett, Michael A. Heroux, Paul T. Lin, Courtenay T. Vaughan, and Alan B. Williams. Poster: mini-applications: vehicles for co-design. In *Proceedings of the 2011 companion on High Performance Computing Networking, Storage and Analysis Companion*, SC '11 Companion, New York, NY, USA, 2011. ACM.

[6] Keren Bergman, Shekhar Borkar, Dan Campbell, William Carlson, William Dally, Monty Denneau, Paul Franzon, William Harrod, Kerry Hill, Jon Hiller, Sherman Karp, Stephen Keckler, Dean Klein, Peter Kogge, Robert Lucas, Mark Richards, Al Scarpelli, Steven Scott, Allan Snavely, Thomas Sterling, R. Stanley Williams, and Katherine Yelick. Exascale computing study: Technology challenges in achieving exascale systems. http://www.science.energy.gov/ascr/Research/CS/DARPAexascale-hardware(2008).pdf, September 2008.

[7] Patrick Bohrer, James Peterson, Mootaz Elnozahy, Ram Rajamony, Ahmed Gheith, Ron Rockhold, Charles Lefurgy, Hazim Shafi, Tarun Nakra, Rick Simpson, Evan Speight, Kartik Sudeep, Eric Van Hensbergen, and Lixin Zhang. Mambo: a full system simulator for the PowerPC architecture. *SIGMETRICS Perform. Eval. Rev.*, 31(4):8–12, March 2004.

[8] Raphaël Bolze, Franck Cappello, Eddy Caron, Michel Daydé, Frédéric Desprez, Emmanuel Jeannot, Yvon Jégou, Stephane Lanteri, Julien Leduc, Noredine Melab, Guillaume Mornet, Raymond Namyst, Pascale Primet, Benjamin Quetier, Olivier Richard, El-Ghazali Talbi, and Iréa Touche. Grid'5000: A large scale and highly reconfigurable experimental grid testbed. *Int. J. High Perform. Comput. Appl.*, 20(4):481–494, November 2006.

[9] Ron Brightwell, Arthur B. Maccabe, and Rolf Riesen. Design, implementation, and performance of MPI on Portals 3.0. *The International Journal of High Performance Computing Applications*, 17(1):7–20, 2003.

[10] Eugene Brooks. The attack of the killer micros. Teraflop Computing Panel, Supercomputing, 1989.

[11] Philippe Charles, Christian Grothoff, Vijay Saraswat, Christopher Donawa, Allan Kielstra, Kemal Ebcioglu, Christoph von Praun, and Vivek Sarkar. X10: an object-oriented approach to non-uniform cluster computing. *SIGPLAN Not.*, 40(10):519–538, October 2005.

[12] Cray, CAPS, Nvidia, and PGI. OpenACC: Directives for accelerators. http://www.openacc-standard.org/, December 2012.

[13] Jack Dongarra, Pete Beckman, Terry Moore, Patrick Aerts, Giovanni Aloisio, Jean-Claude Andre, David Barkai, Jean-Yves Berthou, Taisuke Boku, Bertrand Braunschweig, Franck Cappello, Barbara Chapman, Xuebin Chi, Alok Choudhary, Sudip Dosanjh, Thom Dunning, Sandro Fiore, Al Geist, Bill Gropp, Robert Harrison, Mark Hereld, Michael Heroux, Adolfy Hoisie, Koh Hotta, Zhong Jin, Yutaka Ishikawa, Fred Johnson, Sanjay Kale, Richard Kenway, David Keyes, Bill Kramer, Jesus Labarta, Alain Lichnewsky, Thomas Lippert, Bob Lucas, Barney Maccabe, Satoshi Matsuoka, Paul Messina, Peter Michielse, Bernd Mohr, Matthias S. Mueller, Wolfgang E. Nagel, Hiroshi Nakashima, Michael E Papka, Dan Reed, Mitsuhisa Sato, Ed Seidel, John Shalf, David Skinner, Marc Snir, Thomas Sterling, Rick Stevens, Fred Streitz, Bob Sugar, Shinji Sumimoto, William Tang, John Taylor, Rajeev Thakur, Anne Trefethen, Mateo Valero, Aad Van Der Steen, Jeffrey Vetter, Peg Williams, Robert Wisniewski, and Kathy Yelick. The international exascale software project roadmap. *Int. J. High Perform. Comput. Appl.*, 25(1):3–60, February 2011.

[14] George A. Fierheller. *Do Not Fold, Spindle Or Mutilate: The 'Hole' Story of Punched Cards*. Stewart Publishing & Printing, 2006.

[15] HBP. The human brain project. http://www.humanbrainproject.eu, February 2013.

[16] Tony Hey, Stewart Tansley, and Kristin Tolle, editors. *The Fourth Paradigm: Data-Intesive Scientific Discovery*. Microsoft Research, 2009.

[17] INRIA. Grid 5000. http://www.grid5000.fr, December 2012.

[18] Ken Kennedy, Charles Koelbel, and Hans Zima. The rise and fall of High Performance Fortran: an historical object lesson. In *Proceedings of the third ACM SIGPLAN conference on History of programming languages*, HOPL III, pages 7–1 – 7–22, New York, NY, USA, 2007. ACM.

[19] Sandia National Laboratories. eXascale PRogramming Environment and System Software (XPRESS). http://xstack.sandia.gov/xpress/index.html, December 2012.

[20] Sandia National Laboratories. Institute for advanced architectures and algorithms (IAA). http://iaa.sandia.gov, December 2012.

[21] Oak Ridge National Laboratory. Institute for advanced architectures and algorithms (IAA). http://www.csm.ornl.gov/iaa/, December 2012.

[22] Edgar A. León, Rolf Riesen, Kurt B. Ferreira, and Arthur B. Maccabe. Cache injection for parallel applications. In *Proceedings of the 20th international symposium on High Performance Distributed Computing (HPDC)*, HPDC '11, pages 15–26, New York, NY, USA, 2011. ACM.

[23] Edgar A. León, Rolf Riesen, Arthur B. Maccabe, and Patrick G. Bridges. Instruction-level simulation of a cluster at scale. In *SC'09: High Performance Networking and Computing: Proceedings of the 2009 ACM/IEEE SC09 Conference: November 14–20, 2009, Portland, Oregon, USA*. ACM Press and IEEE Computer Society Press, November 2009.

[24] Paul Messina, Thomas Sterling, Jarrett S. Cohen, and Paul H. Smith. Leading HPC figures consider petaflops computing. http://www.crpc.rice.edu/newsArchive/1463.html, December 2012.

[25] Hans Meuer, Erich Strohmaier, Jack Dongarra, and Horst Simon. The top500 list: Twenty years of insight into HPC performance. http://www.top500.org, December 2012.

[26] NERSC. Draft NERSC-8 / Trinity benchmarks. http://www.nersc.gov/systems/trinity-nersc-8-rfp/draft-nersc-8-trinity-benchmarks/, February 2013.

[27] Nvidia. CUDA parallel computing platform. http://www.nvidia.com/object/cuda_home_new.html, December 2012.

[28] United States of America Department of Energy. X-stack software. http://www.xstack.org/, December 2012.

[29] United States of America Department of Energy. Scientific discovery through advanced computing (SciDAC): Co-design. http://science.energy.gov/ascr/research/scidac/co-design/, February 2013.

[30] United States of America Department of Energy. X-stack software. http://www.xstack.org/, February 2013.

[31] Federal Office of Meteorology and Climatology MeteoSwiss. The numerical weather prediction model COSMO. http://www.meteosuisse.admin.ch/web/en/weather/models/cosmo.html, February 2013.

[32] Steven J. Plimpton and Karen D. Devine. Mapreduce in MPI for large-scale graph algorithms. *Parallel Comput.*, 37(9):610–632, September 2011.

[33] John Shalf, Dan Quinlan, and Curtis Janssen. Rethinking hardware-software codesign for exascale systems. *Computer*, 44(11):22–30, November 2011.

[34] Horst Simon. Exascale chat. http://www.isc-events.com/isc12_ap/eventdetails.php?t=event&o=265&a=select&ra=speakerdetails, June 2012.

[35] Thomas Sterling, Paul Messina, and Paul H. Smith. *Enabling technologies for petaflops computing*. MIT Press, Cambridge, MA, USA, 1995.

[36] Tiankai Tu, Charles A. Rendleman, David W. Borhani, Ron O. Dror, Justin Gullingsrud, Morten Ø. Jensen, John L. Klepeis, Paul Maragakis, Patrick Miller, Kate A. Stafford, and David E. Shaw. A scalable parallel framework for analyzing terascale molecular dynamics simulation trajectories. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, SC '08. IEEE Press, 2008.

[37] Leslie G. Valiant. A bridging model for parallel computation. *Commun. ACM*, 33(8):103–111, August 1990.

[38] Wikipedia. Intel MIC. http://en.wikipedia.org/wiki/Intel_MIC, December 2012.

Index