7th IESP Meeting Working Group Report on:

Revolutionary Approaches: Punctuated Equilibrium or Continuous Evolution?

Thomas L. Sterling, Indiana University
Bronis R. de Supinski, Lawrence Livermore National Laboratory
version-5

1. Introduction

Projection of the possible path forward for HPC towards exascale computing is guided by two histories. One expresses the success of incremental techniques of improvement over the last two decades in advancing delivered computing performance. The second exposes as many as half a dozen paradigm shifts over the extended period of 60 to 70 years when progress in underlying device technologies catalyzed dramatic changes in the way computing was organized and conducted and computing systems were designed and operated. At the 7th meeting of the International Exascale Software Project (IESP) conducted in Cologne Germany, a working group considered the potential needs and approaches of a revolutionary strategy for exascale software. This brief note documents its initial findings.

The evolutionary path successfully supported an era, *Pax MPI*, of stability with dependable improvements through incremental changes. The latter revolutionary path over the extended history also provided marked events of dramatic improvement but through disruptive changes that demanded redefinition of structures and methods. These two histories provide contradictory lessons and present the HPC community with the challenge of determining which path applies to the realization of sustained exaflops performance by the end of this decade.

A sequence of incremental enhancements provides a clear path to preserve existing investments in large-scale applications so it is far preferable to the more risky disruptive strategy. However, if an evolutionary trajectory cannot reach the desired end state then we must pursue a revolutionary although uncertain path even in the presence of incurred costs to refactoring systems, applications, and methods.

The international HPC community dedicated to the realization of exascale computing by the end of this decade is divided on the issue of the approach and strategy of achieving this goal. Four dominant issues prevail: the socialization and controversy of change; the exascale challenges that may demand revolution; the candidate system areas in which revolution is possible; and the mitigation of the disruptive impact of such revolutionary changes. We require a détente between those demanding cautionary progress and those insisting on change in spite of fracturing the means of scalability beyond current practices.

More than gap filling may be required but only through responsible methods that include continuity with existing legacy codes, models, and skills. In a key finding, the IESP working group concluded that in many cases the same end goal could be reached through both punctuated equilibrium (i.e., revolutionary change) or continuous evolution. We may require large leaps to determine the correct end state but we may also be able to reach that end state through more gradual methods once we know the right direction.

Unfortunately the community has become polarized to a degree that inhibits the formulation of a shared constructive middle ground. Fear of the extremes has precluded the development of a rational all encompassing strategy to address the daunting challenges while maintaining the continuity of productive capability from passed software investment in applications and environments. This note begins the dialog to bridge the current chasm and to derive the needed techniques.

2. Opportunities for Possible Revolutionary Approaches

The IESP "Revolutionary Approaches" working group focused on those areas that may most require more than incremental extensions of current and recent practices. We identified these areas through the subjective criterion of the level of concern felt by working group members in reaching exascale by 2020. We discussed each issue and we report the six issues for which no strong disagreement was raised. Thus, we provide a conservative view on controversial ideas, which is perhaps an oxymoron.

2.1 Execution Models

The disruptive events in the history of supercomputing that mark the phase changes of HPC in each case resulted in a new paradigm of computing as reflected in the *execution model*. An execution model is a set of guiding principles that govern the roles and relationships of the integral component layers of a computing. The current predominant scalable execution model is Communicating Sequential Processes (CSP). New execution models capture advances, and the associated challenges, in underlying enabling technologies. The execution model impacts co-design across programming models and architectures as well as the supporting system software. As we prepare for exascale systems, we may require a new execution model that better reflects challenges posed by extreme concurrency, multicore sockets and GPUs, all of which may require greater support for asynchrony than CSP provides.

2.2 Changing the Way We Think

More broadly, an area that may require revolutionary changes is how we think about the merger of systems and programming. Conventionally the user has the illusion of controlling the machine. Future systems may require us to relinquish this simplistic assumption. Extreme concurrency will require us to exploit runtime information that allows non-determinacy and variability of the execution path that produces dependable, but not bitwise reproducible, answers. Thus a change of culture and attitude in which we do not control every cycle but rather influence the execution to find a path to the right answer appears necessary to deliver substantially greater performance than we achieve on current systems.

2.3 Incorporating Intelligent Methods

As systems become more complex and heterogeneous with highly varying latencies and overheads and potentially unpredictable contention, we may have to select among alternatives at multiple levels on-the-fly through intelligent controls. Intelligent controls could measure and detect conditions on a continual basis throughout a computation and can support introspective operation. They could employ goal-driven objective functions that determine how to select among alternative approaches or paths. Such methods must operate in real time but cannot impose significant overhead or any potential benefits will be dissipated. Objective functions are reflected at multiple levels from simple autonomic low-level responses to high-level complex decisions that reflect difficult choices. For future HPC, intelligent controls must reflect spatial awareness, as well as temporal, from a data perspective. Further we may require hardware support for operational knowledge and prediction, potentially through an Information Backplane that provides a protocol between system layers for mutual dynamic introspective behavior.

2.4 Operating Systems

Most existing supercomputers employ some variant of the Unix operating system, with Linux dominating the Top-500 list. While continually evolving, the basic architecture of this critical software was established over 30 years ago. Because of the inadequacies of conventional Unix, several variations, including lightweight kernels and user runtime systems for improved efficiencies, have been pursued. During much of the last two decades on clusters and MPPs systems have really been collectives of many operating systems running side by side instead of a single system image with some umbrella scheduling package.

The challenge of managing a billion cores and the associated scale of their memory and interconnection network is unprecedented. Exascale systems will pose additional challenges such as fault tolerance through graceful degradation, dynamic resource management, and energy control. Also, system and core architectures are changing significantly, with multicore and heterogeneous computing emerging. Future parallel programming languages may require new classes of service from the OS as programming models change to meet scaling and efficiency requirements. We have not yet resolved the OS role in addressing these additional complexities.

OS changes may be proactive with new concepts in system control or they may be reactive in support of dramatic changes in architecture, programming paradigms, and runtime systems that rely on the OS. In either case, the OS may require revolutionary change to meet its fundamental role in HPC systems.

2.5 Programming Models

Future programming models are the source of the greatest dissonance in the plans for exascale systems, perhaps because the right solution (i.e., end state) is so unclear. Parallel programming models change in response to advances in system structure to represent parallelism, data structures, and control flow in response to technology progress. These changes may arise from a new execution model or simply reflect an alternative means of crafting parallel programs. In either case, a programming model provides a necessary level of abstraction that couples the demands of an application with the resource capabilities of the physical system.

Exascale programming models must devise a set of semantics that yield sustained billion-way concurrency in heterogeneous environments. These semantics include the forms of parallelism and the means of sequencing and enabling operations such as synchronization constructs. They also entail the interrelationship of control flow with data structures and their distribution. While we still must determine how to provide the required support, the new semantics will often require algorithmic changes since many existing algorithms cannot sustain billion-way concurrency.

Many factors besides concurrency are important to exascale programming models. A particularly critical concern is the migration path for legacy codes from existing programming models. Further, those existing programming models must derive some benefit from exascale systems because the existing investment in large-scale applications, as well as the time required to produce them, is so large. For similar reasons, new programming models must interoperate effectively with old ones. For these reasons, concurrently pursuing revolutionary and evolutionary paths to the final exascale programming model may provide the greatest benefit.

2.6 Correctness and Debugging

One objective widely thought to require a revolutionary approach is the critical challenges of achieving correctness through methods of verification and debugging. Exascale performance does not impose this need: existing strategies provide inadequate insight into the root cause of programming errors. However, billion-way asynchronous parallelism may make this lack of insight intolerable. We may even have to replace current concepts of correctness and absolute exactness with boundedness and quasi reproducibility. Tools for verification and validation, error detection and debugging, and confidence in answers at varying scales require new models, quite possibly revolutionary, to solve this problem.

2.7 Persistent Storage

We face an entire array of application challenges that are largely data-intensive and bound by the capabilities of the storage system. We have relied on the basic model of file systems that was derived in the 1960s with some semantics from the 1940s (e.g., rewind). Even with important strides such as MPI-IO, HPF5, and RAID, we still treat mass storage as an entirely separate system from in-memory computation.

This view fails to balance the computational and storage capabilities of our systems. The growing imbalance between them as well as reduce main memory capacity relate to computational capabilities will exacerbate existing problems with how applications use storage capabilities. Further NVRAM technology may lead to another layer in the storage hierarchy. Thus, vertical movement of data may increase, leading to techniques reminiscent of out-of-core algorithms. We may require far more lightweight and agile transitions between the storage technologies and new programming methods to unify them in a single name space to achieve dynamic reactive data movement. We may need these revolutionary strategies to realize the full potential of exascale computing for the widest array of applications.

3. Related Factors and Issues

We discussed several issues related to responsibly realizing potential revolutions in computing systems and techniques in order to achieve exascale computing capability. This section summarizes some of the most prominent ones.

3.1 Managing Revolution

If future supercomputer designs do not deliver increased application performance, they will not be deployed and the field will stall, perhaps irrevocably as commercial aviation did in the 1960s, after when flight times have not changed significantly. We may require a perhaps oxymoronic “incremental revolution” by which the workload continues to run but for which significant performance improvements require concomitant investment in code refactoring.

Supporting gradual steps will permit planning by which organizations, agencies, and nations can project their respective paths across HPC generations. Before we impose change, we must identify the destination, possibly through preparatory research and proof-of-concept development. At launch time of the revolutionary system concepts, the process must begin with something credible. Further, we expect many current practices will convey to future methodologies.

Nonetheless, for something new to be born, something has to die. HPC users will adopt new practices over time and adapt to them in order to achieve orders of magnitude benefits in science, technology, commerce, and security. We require a culture change but one that can be managed and transitioned rather than disruptive through bridging methods and education. Rewriting of widely used libraries by groups of experts could provide many early benefits to the broad community. High level programming models, even declarative or domain specific programming interfaces may also mitigate the challenge of transition. All of these approaches and others constitute means of managing the possibly essential HPC revolution.

3.2 Intelligence in Systems

Except in some special cases, future systems will rely, perhaps heavily, on runtime functionality realized through new runtime system software in cooperation with the

OS and architecture driven by compiler and user programming interfaces. The incorporation of intelligent controls will provide the opportunity to introduce higher order operational policies. Many methods are already being pursued to self-adapt codes to underlying core architectures, for work scheduling, and for dynamic load balancing. Other areas such as fault management and energy optimization are potential candidates. A new multilevel intelligence that manages these disparate goals may provide a quasi self-aware environment through introspective means that can optimize potentially conflicting requirements to achieve overall best operation.

3.3 Reliance on More than Compiler Technology

Historically compilers have dictated system usage. Decades of remarkable progress in advanced compiler technology has bred a culture of strong reliance on this system layer. However, reliance on compilers has also limited progress in efficiency and scalability. Problems like inter-procedural analysis and automatic parallelization are still topics of research even after decades of admittedly good work. Often limited to static information, compilers are necessarily conservative.

The common culture to rely on compilers hinders future directions to deliver superior capabilities. Compilers cannot alone provide the necessary understanding to proscribe how large-scale systems must operate. With the emergence of new runtime systems applied to the field of HPC, an entirely new class of opportunities is becoming available. Although compilers will continue to play a crucial role, we must also trust runtime systems and even some advances in hardware support to guide the computation of exascale systems and their applications.

4. Conclusions

We, the international community planning for exascale, are uncertain at this time of the software architecture and the software components of which it is comprised required to support of exascale systems by the end of this decade. Some elements may reflect revolutionary methods in order to support part or all of the workload. We have summarized the initial conclusions of a self-selected representative group, the working group on “Revolutionary Approaches” of the 7th IESP meeting.

IESP is considering four important issues in order to establish its long-term strategy. First, we must socialize the topic to overcome the current polarization with respect to possible strategies. Second, we must assess which functional areas will most likely demand some revolutionary techniques. Third, we must identify the revolutionary approaches that can address those requirements. Fourth, we must mitigate the potentially disruptive effects of those revolutionary approaches. This report can serve as initial attempt to overcome the polarization by identifying consensus functional areas for significant change and discussing potential mitigation approaches. Importantly, we still require much more research to identify the appropriate end states before the international HPC community can confidently pursue either revolutionary or evolutionary strategies to reach those states.

Contributors:

- Pete Beckman, Argonne National Laboratory
- Ron Brightwell, Sandia National Laboratories
- Barbara Chapman, University of Houston
- Jack Dongarra, University of Tennessee, Knoxville
- Anshu Dubey, University of C
- Al Geist, Oak Ridge National Laboratory
- Andrew Jones, NAG
- Alice Koniges, Lawrence Berkeley National Laboratory
- Jesus Labarta, Barcelona Supercomputing Centre
- Bob Lucas, USC Information Sciences Institute
- Paul Messina, Argonne National Laboratory
- Hiroshi Nakamura, University of Tokyo
- Hiroshi Nakashimi, Kyoto University
- Thomas Sterling, Indiana University (Co-Chair)
- Shinji Sumimoto, Fujitsu Inc.
- Bronis de Supinski, Lawrence Livermore National Laboratory (Co-Chair)
- Kenjiro Taura, University of Tokyo
- Rajeev Thaker, Argonne National Laboratory
- Anne Trefethen, Oxford University
- Vladimir Voevodin, Moscow State University