EDITORS: Kathryn Mohror, mohror1@llnl.gov
John M. Shalf, jshalf@lbl.gov

DEPARTMENT: LEADERSHIP COMPUTING

Exascale Was Not Inevitable; Neither Is What Comes Next

Erik W. Draeger, *Lawrence Livermore National Laboratory, Livermore, CA, 94550, USA*
Andrew Siegel, *Argonne National Laboratory, and University of Chicago, Chicago, IL, 60637, USA*

The long, steady advance of supercomputing capabilities historically makes it tempting to assume that similar future improvements are inevitable. In fact, the successful path to exascale systems was far from certain, requiring a shift to accelerator-based designs, an associated end-to-end rethinking of traditional computational approaches, and a fundamental change in how scientists collaborate and interact. In this article, we share lessons from the U.S. Department of Energy Exascale Computing Project along with perspectives on how to design and implement scientific applications in the exascale and post-exascale eras.

Most of us are familiar with Moore's Law: the famous 1965 prediction turned catchphrase for the long, steady, and remarkable trend of performance improvement in commodity computing hardware over the past 60 years.

MOTIVATION

The realization of Moore's prediction represents a mind-boggling technological achievement, resulting in modern computers that can calculate an astonishing ten-million to one-hundred-million times faster than their counterparts of just 50 years ago, and with an associated overall impact on society that would be impossible to overstate. Commensurate with these rapid technological advances is a lesser known, parallel trend that has an equally profound impact on many areas of science and engineering—the more specialized, exotic machines known as supercomputers, which form the foundation of the field known as high-performance computing (HPC).

From a computer architecture standpoint, HPC focuses on designing specialized systems much bigger and faster than their commodity counterparts. The performance of these systems is traditionally measured using floating-point operations per second (flop/s). While it is debatable whether flop/s remains a useful measure of a supercomputer's true capability, they continue to serve as the primary temporal marker for HPC progress. The fastest supercomputers typically perform 10,000–100,000 times more flop/s than conventional systems, giving a roughly 20-year head start compared to reliance on commodity hardware. In isolation, the value of such specialized systems is marginal. But for critical open questions in science and technology underpinned by massive computational models, a two-decade head start is often the difference between progress and stagnation. The past 50+ years have shown that supercomputers are an increasingly critical tool for research across a broadening range of disciplines, spanning topics from fundamental scientific inquiry to the optimized design of complex engineered systems.

The history of supercomputer design is a field of research in its own right, with a plethora of remarkable achievements, the ebb and flow of numerous technological trends, and the arrival and disappearance of many key contributors, both large and small. A watershed transition in supercomputer design occurred in the early 1990s with the trend away from custom "vector" architectures to customized networks ("clusters") of less specialized, often commodity hardware. This era spanned close to 25 years, from the early 1990s until roughly 2015, and coincided with progress that brought us from gigaflop/s to teraflop/s. From the perspective of the HPC practitioner, the cluster period represented

Digital Object Identifier 10.1109/MCSE.2023.3311375
Date of current version 25 October 2023.

a long era of relative stability. The abstract machine model varied little from generation to generation. Leveraging improved performance typically meant "scaling" distributed algorithms to run efficiently across increasingly larger clusters, while at the same time benefiting from higher performance of the individual processors themselves. Technical challenges were always present but never seemed insuperable. Software improvements tended to be incremental refinements largely consistent with existing approaches. This ease of transition enabled a sharper focus on core elements of HPC—scientific investigation and the discovery of new, better algorithms that has resulted in a parallel "algorithmic Moore's Law."[1]

By 2015, at the end of the cluster era, HPC had long been established as a critical pillar of scientific discovery, engineering design, and national security. The HPC community had come to expect and rely on a supercomputing version of Moore's Law, enabling continuously increasing performance while also unleashing creative new ways to make use of rapidly increasing compute resources. At the same time, the future of HPC had suddenly became very unclear. The HPC community uses flop goalposts as symbolic beacons in units of thousand-fold increases in performance—gigaflop/s (10^9), teraflop/s (10^{12}) petaflop/s (10^{15}). The petaflop/s barrier was achieved, but the scale-up approach to designing larger systems hit fundamental roadblocks that marked a clear end of the cluster era. Among these roadblocks, none seemed more insurmountable than the power associated with running an HPC resource—both the power needed to operate it and the heat that needed to be dissipated to maintain its operation. The reality was stark—the path from petascale to exascale required a 1,000x performance increase at a roughly 2x power cost. Furthermore, the level of concurrency required to realize exascale performance would likely reach tens of billions, presenting an unprecedented challenge both for existing algorithms as well as system resiliency. None of these barriers had obvious solutions. Was it even possible to build a usable exascale machine? If so, how would it be programmed? What would that mean for decades of investment in existing legacy codes? And what are the implications for the longer term future? To overcome these obstacles and realize a productive exascale ecosystem required a bold and novel approach that took the community out of their comfort zone into new and unfamiliar ways of collaborating.

THE EXASCALE COMPUTING PROJECT

In recognition of the critical importance of exascale and to meet its massive technical challenges, Congress and the Department of Energy (DOE) provided the HPC community with a unique opportunity. In 2016, The Exascale Computing Project (ECP) was launched as a seven-year, collaborative community effort of unprecedented scale aimed at realizing a productive exascale ecosystem by 2022. Approximately 1,000 researchers participated across 15 DOE research labs and 57 universities. ECP was organized around three principal focus areas—application development, software technology, and hardware integration, intended to represent all aspects of the computational ecosystem required to enable productive science on exascale platforms. This included both a diverse set of roughly two dozen leading-edge scientific applications developed alongside a broad portfolio of enabling computational technologies (ECTs). Close coordination within and between focus areas was driven using shared-fate milestones and integration-based deliverables.

The size and scope of ECP was unprecedented. While DOE has long made substantial investments in scientific applications and their ECT dependencies, it has never organized a sustained, coordinated effort across so many institutions and disciplinary boundaries toward a common set of project goals. For application teams, ECP represented a unique opportunity to conduct a comprehensive overhaul of established capabilities in a highly collaborative environment with access to domain experts, HPC vendors, and the leadership class facilities.

RETHINKING APPLICATION DEVELOPMENT FOR EXASCALE AND BEYOND

It has now been nearly seven years since the inception of ECP, and ECP application and software technology teams across a wide range of scientific disciplines are closing in on successful completion of their project goals.[2] The coordinated effort between domain scientists, software developers, and hardware experts was critical to this successful outcome[3] and exposed realities about fundamental aspects of scientific computing projects that may benefit the community in charting a path forward. We share a number of key observations next.

Don't Port Application Codes

The transition to exascale required a move from CPU to hybrid, accelerator-based (GPU) architectures. A common misconception is that this constituted a "code porting" exercise, e.g., focusing on in-place modifications like loop interchange and data layout to enable more efficient execution on exascale hardware.

While such optimizations have their role in ECP projects, porting alone is insufficient. During such a technological revolution a far more holistic approach to algorithm redesign is necessary. The application as a whole must be thoughtfully and thoroughly reexamined in light of both existing and future hardware realities. Changes to the speed of computation relative to memory bandwidth, the availability of specialized compute hardware, and diversity in the size and layout of memory hierarchies all necessitate a ground-up reevaluation of the physical models employed, the discretization approach, the underlying equations, and how best to solve them within a given accuracy.

An early example of this type of algorithmic reassessment came from a coupled computational fluid dynamics and Monte Carlo neutronics application designed to simulate proposed small modular nuclear reactor designs. One of the constituent neutronics codes, Shift,[a] was originally designed around a traditional "history-based" algorithm, where neutrons are distributed across processors that each track their local particles simultaneously, one by one from birth to death (absorption or escape). This algorithm is highly efficient on CPUs but performs poorly on GPU architectures, which utilize a single-instruction, multiple-threads (SIMT) execution model. SIMT parallelism depends on each thread following the same execution path through the code, a requirement seemingly tailor-made to conflict with a random Monte Carlo sampling of particle interactions. To mitigate branch divergence and leverage SIMT parallelism, researchers had to literally turn the algorithm inside out, implementing a radically different, "event-based" algorithm built on ideas originally developed to leverage vector-based supercomputers in the 1970s. In this approach, parallel workloads are sorted and assembled based on common code paths, ensuring SIMT-friendly execution by construction. This required a substantial refactoring of Shift but resulted in significantly improved performance on the same hardware.[4]

Other ECP projects adapted their approach in even more fundamental ways. The popular quantum chemistry package GAMESS[b] switched to a fragment-based approach to reduce complexity to O(n) for high accuracy electronic structure calculations; the EQSIM seismic modeling project[c] added an adaptively refined curvilinear mesh to represent surface topography coupled to a more efficient Cartesian mesh below the surface; the climate modeling project added a two-dimensional localized cloud-resolving subgrid model versus exploring brute force higher-resolution three-dimensional calculations. In these and many related cases, the availability of exascale forced researchers to go far beyond porting, and instead to think more fundamentally about which paths would most take advantage of added performance while enhancing scientific insight.

[a]Part of the SCALE Suite: https://www.ornl.gov/scale
[b]https://www.msg.chem.iastate.edu/gamess/
[c]https://github.com/geodynamics/sw4

Don't Throw It Over The Fence

Historically, HPC practitioners have placed a high value on self-reliance. Self-reliance implies control, and new HPC systems are often unstable, slow to support full language specifications, and do not provide early access to optimized versions of even common third-party libraries. Minimizing external dependencies and maintaining substantial control of the application has allowed developers to directly manage performance at all levels and to more quickly adapt to new hardware.

OTHER ECP PROJECTS ADAPTED THEIR APPROACH IN EVEN MORE FUNDAMENTAL WAYS.

Despite these benefits, it has become increasingly clear that this model is no longer practical for most application teams. Eschewing third party ECTs now equates to foregoing the performance benefits of expert specialization on increasingly complex systems. Furthermore, heterogeneous hardware increases the challenges associated with "over-the-fence" development, where ECT developers build and refine software independently before releasing it ("throwing it over the fence") to application teams. Even many common ECT use cases—linear solvers, time integration, spatial discretization, geometry composition—must be specially adapted to the equations being solved and the target application domain. Optimal software implementations often require careful data management and persistence between kernels. For programming models, this means development and optimization using realistically complex workloads to avoid inadvertent bottlenecks.

From the perspective of the ECT provider, the most successful tools in ECP follow a cycle of Development → Deployment → Integration → Refinement. Cases where optimal abstractions are identified a priori, in the absence of focused integration with application teams (an "over-the-fence" model), are far more the exception than the rule. Examples abound in ECP of the critical importance of direct producer/consumer engagement. Image reconstruction of light-source data required fast nonuniform fast Fourier transforms of

custom size to achieve real-time throughput goals on exascale systems, a use case unlikely to be prioritized by the vendor; a neutron particle-tracking application required meshing support, but the available optimized particle-mesh libraries failed to support the use case of a particle fully exiting the domain, requiring rethinking of basic abstractions to accommodate this more general use case; an astrophysics application required specialized dynamically refined oct-tree mesh representation, but the leading adaptive meshing framework initially lacked optimized support for this specialized choice; optimal power-flow calculations for power grids require optimized sparse indefinite solvers for GPUs, something that vendors did not initially prioritize, resulting in surprisingly poor performance on their challenge problem.

A common theme of all these examples is the need for direct engagement with applications by ECT teams to uncover key performance bottlenecks or capability gaps. By incentivizing such collaborations and breaking down the development barriers between providers and end users, these and many similar problems were quickly fixed, supporting application goals, and broadening the impact of the ECT on new potential users.

STAYING TOO LONG IN THE REGIME OF BENCHMARKS IS TEMPTING, BUT A MISTAKE: COMPLEXITY MUST BE MET HEAD ON.

Don't Stop At Benchmarks

There has long been and will continue to be value in simple, well-defined software workloads that can be used for direct comparisons between compilers, programming models and compute hardware. Increasingly, HPC vendors are eager for so-called proxy applications,[5] simplified representations of application codes with minimal complexity, to prototype or demonstrate the anticipated performance of new machines. While these tools have their place, application teams are cautioned against relying too heavily on such simplified abstractions. By construction, benchmarks and proxies strip away many of the challenging aspects of full scientific application workloads, e.g., load imbalance, complex inputs and data decomposition, deep call stacks, and kernel diversity.

ECP application teams were tasked with demonstrating a significant performance or capability improvement on an exascale challenge problem, defined in consultation with external subject matter experts to represent a meaningful advance in their field. Although many teams utilized standalone kernel benchmarks and proxy apps as a starting point for performance analysis and debugging, full application runs consistently uncovered issues that were not exposed by the simplified workloads. Staying too long in the regime of benchmarks is tempting, but a mistake: complexity must be met head on.

WHAT NEXT?

While the extraordinary achievements of ECP are a clear demonstration of the power and impact of exascale computing, the reality is that most HPC projects thrive on "mid-range" HPC resources. We foresee a near-term future where the tools and technologies that achieved exascale begin to fill the broader HPC ecosystem. Currently, new procurements in industry, academia, and research labs are increasingly favoring small to mid-sized clusters of accelerated nodes versus larger clusters of conventional CPUs. While efficient use of these systems is challenging, a side effect of ECP has been to pave the way for the waves of applications that are sure to follow. We fear that the gap between early adopters and those who haven't made the transition is enormous and growing rapidly. Application teams who don't want to be left behind may look to ECP for examples of how to successfully adapt to the new HPC ecosystem and follow its model of end-to-end redesign in close collaboration with the developers of critical software dependencies when possible.

Perhaps the biggest question now is what comes after exascale? What will HPC hardware look like? What scale will ultimately be achievable within a given power budget? We cannot answer these questions with certainty, but it seems likely that the complex interdependency of hardware and software will continue to grow and with it the need to rethink and readapt applications and their enabling computational technologies. Large-scale, coordinated efforts similar to ECP will likely be needed to drive significant capability advancements going forward.

ACKNOWLEDGMENTS

This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

REFERENCES

1. President's Information Technology Advisory Committee, "Computational science: Ensuring America's competitiveness," *Nat. Coordination Office Inf. Technol. Res. Develop.*, Washington, DC, USA, Tech. Rep., Jun. 2005. [Online]. Available: http://www.nitrd.gov/pitac/reports/20050609_computational/computational.pdf
2. T. M. Evans et al., "A survey of software implementations used by application codes in the exascale computing project," *Int. J. High Perform. Comput. Appl.*, vol. 36, no. 1, pp. 5–12, 2022, doi: 10.1177/10943420211028940.
3. L. McInnes et al., "How community software ecosystems can unlock the potential of exascale computing," *Nature Comput. Sci.*, vol. 1, no. 2, pp. 92–94, Feb. 2021, doi: 10.1038/s43588-021-00033-y.
4. S. P. Hamilton and T. M. Evans, "Continuous-energy monte Carlo neutron transport on GPUs in the shift code," *Ann. Nucl. Energy*, vol. 128, pp. 236–247, Jun. 2019, doi: 10.1016/j.anucene.2019.01.012.
5. R. F. Barrett et al., "On the role of co-design in high performance computing," in *Transition of HPC Towards Exascale Computing*, E. H. D'Hollander et al., Eds. Amsterdam, The Netherlands: IOS Press, 2013, pp. 141–155.

ERIK W. DRAEGER is the deputy director of application development for the Exascale Computing Project. He is also the director of the High Performance Computing Innovation Center and RADIUSS project at Lawrence Livermore National Laboratory, Livermore, CA, 94550, USA, as well as the Scientific Computing group leader at the Center for Applied Scientific Computing. Contact him at draeger1@llnl.gov.

ANDREW SIEGEL is the director of application development for the Exascale Computing Project. He is a senior scientist at Argonne National Laboratory, and an adjunct professor at the University of Chicago, Chicago, IL, 60637, USA. Contact him at siegela@uchicago.edu.