How community software ecosystems can unlock the potential of exascale computing

Emerging exascale architectures and systems will provide a sizable increase in raw computing power for science. To ensure the full potential of these new and diverse architectures, as well as the longevity and sustainability of science applications, we need to embrace software ecosystems as first-class citizens.

Lois Curfman McInnes, Michael A. Heroux, Erik W. Draeger, Andrew Siegel, Susan Coghlan and Katie Antypas

After decades of relatively constant and straightforward performance growth in computing hardware, high-performance computing (HPC) applications are now facing disruptive changes in hardware architectures driven by the end of Dennard scaling and the slowing of Moore's law. These new computing architectures on the path toward exascale — defined as the capability to perform 10^{18} operations per second — have the potential to unleash enormous gains in computational capability for science (even at the desktop level) but come with substantially increased programming complexity[1,2]. Heterogeneous memory spaces, massive parallel concurrency requirements, and reduced bandwidth to compute ratios, along with changing storage paradigms and a proliferation of hardware accelerators, make developing and optimizing applications for upcoming systems a challenge. Here, we argue that the casual confederations of applications and software used on HPC systems in the past are not enough to ensure efficient utilization of upcoming hardware. Instead, each software project should intentionally consider the relationships among products that it develops and uses. Moreover, communities should take on software ecosystem perspectives — that is, consider establishing collections of interdependent products whose development teams have incentives to collaborate to provide aggregate value, where the whole is greater than the sum of its parts. Community software ecosystem perspectives are essential to fully harness the complementary advances needed in applied mathematics, computer science and domain sciences as we work toward the exascale era and a sustainable path for next-generation computational science.

Collaborating across disciplines via software

For application developers, the idea of delegating functionality to external libraries is not new. For decades, standard mathematical operations such as dense linear algebra and fast Fourier transforms have routinely been outsourced to highly optimized, platform-specific implementations. However, introducing external dependencies has never been a step for application teams to take lightly. Indeed, conventional wisdom for codes targeting portability and performance on cutting-edge platforms has been to minimize their dependence on software capabilities not directly supported by vendors, to reduce the risk of roadblocks that cannot be quickly fixed.

With the rapid pace of change and increasing diversity in hardware, this conservative approach now carries risks of its own because achieving good performance on new architectures requires increased development effort for refactoring and tuning. The transition to hosted accelerated architectures, specifically nodes with multicore CPUs and multiple GPUs, adds a challenging dimension of complexity. The need to develop and encode new, highly concurrent algorithms, coordinate data motion, and achieve parallel kernel execution on different heterogeneous architectures has given rise to new programming models, including abstraction layers (for example, Kokkos, RAJA and UMPIRE)[3], designed to enable performance portability — the ability to obtain good performance from the same source code on a variety of target platforms.

Foundational software infrastructure for common computational domains, from adaptive mesh refinement libraries to shared components for multiphysics coupling, provides opportunities for both modernization of existing applications as well as new development, especially as we target new heterogeneous architectures. The decisions facing application teams now are not only which external software products to use but also how to best engage with development of these products. Application teams who chose to become active participants in community ecosystems — coordinating to fully understand intended use cases, define robust APIs, provide input to testing and optimization, and identify opportunities for shared development — are better positioned to achieve their science goals than those who are simply passive consumers of software products.

New science drivers in HPC

Traditional simulation and modeling continue to drive exponentially increasing demand for HPC due to requirements for higher resolution, increased fidelity, and multiphysics/multiscale coupling; at the same time, new HPC workloads are also emerging. The explosion of data from sensors, detectors, accelerators, microscopes, telescopes and sequencers is overwhelming local computing capabilities, as well as scientists' ability to move, manage, store and analyze the data. New communities and large-scale collaborations developing around these experimental instruments require novel modes of interacting with HPC systems. For example, application teams incorporating large-scale data often have complex software dependencies requiring specific versions and software instances, which must be validated and tested before use. The teams typically have complex workflow requirements, with a scientific pipeline potentially starting and ending outside a computational facility, meaning workflow and scheduler software become eminently important. Furthermore, the increasing use of artificial intelligence (AI) in both large-scale simulations and experimental data analysis is driving changes in the traditional HPC software stack. For example, AI software products developed in industry are being ported and optimized for HPC systems, while new research is under way on AI software to serve the unique needs of science.

Software as a first-class citizen

Reusable software products, which encapsulate domain-specific expertise that is leverageable across multiple applications, represent key opportunities for collaboration. Researchers in applied mathematics and computer science have a long tradition of developing cutting-edge software libraries and tools that underpin advances in computational science[4–6]. Often, however, the resulting software is a byproduct of research on new algorithms or domain-specific science, rather than direct investment in the software itself[7]. Given the disruptive changes in computing architectures, the increasing system heterogeneity, and the increasing number of communities coming to HPC with different expectations and requirements for software, it is important to design new paradigms for developing, testing, managing and deploying software ecosystems on the road to exascale. Furthermore, issues related to dependency management bring to the forefront challenges in documentation, distribution, coordination and reproducibility. Software quality assurance is an increasingly urgent topic, as open-source software is emerging as a central resource for technical computing in government, academia and industry. Moreover, there is general awareness that the broader computational science community faces similar urgent software challenges, even if it does not have the mandate to prepare for exascale computing platforms. Thus, the time is ripe for the computational science community to fully embrace software ecosystem perspectives.

The Exascale Computing Project

Teams from the Exascale Computing Project (ECP)[8], funded by the US Department of Energy (DOE), are working toward scientific advances on forthcoming exascale platforms. Efforts target a diverse suite of applications in chemistry, materials, energy, Earth and space science, data analytics, optimization, AI, and national security[9,10]. In turn, these applications build on software components, including programming models and runtimes, mathematical libraries, data and visualization packages, and development tools[3] that comprise the Extreme-scale Scientific Software Stack (E4S). E4S represents a portfolio-driven effort to collect, test and deliver the latest in reusable open-source HPC software products, as driven by the common needs of applications. As new exascale-ready components are developed, they are integrated and tested to ensure correctness and version compatibility, and are delivered to application teams via from-source builds, containers and cloud environments. E4S establishes product quality expectations and provides a portal as a starting point for access to all product documentation. As we go forward, E4S will also play a central role in software quality assurance, needed to help ensure the integrity of computational results.

Early experiences with E4S indicate some success in helping to overcome software collaboration challenges across distributed aggregate teams. A key lesson learned is the need for close collaboration between teams developing applications and reusable software technologies, as well as the need for crosscutting strategies to increase developer productivity and software sustainability, thereby mitigating technical risks by building a firmer foundation for reproducible, sustainable science[11]. E4S is an open architecture, welcoming contributions from the broader HPC community. Recognizing that each community's needs are unique, the E4S approach being used by the ECP (as well as the software ecosystem strategies of other communities) could be used as inspiration for a community to establish an ecosystem that addresses its unique goals and requirements.

Leveraging cognitive and social sciences

Large-scale computational science is fundamentally team-oriented. Establishing team environments that foster creativity, innovation, individual satisfaction and team productivity is essential for progress, but presently computational science teams tend to treat individual and team challenges in an ad hoc manner. We can benefit from applying the cognitive and social sciences to better understand and improve how teams develop and use software to conduct research[12]. We are finding that engaging trained cognitive and social scientists as integral team members can lead to more effective outcomes, especially in the context of distributed aggregate teams (also known as 'teams of teams')[13], who are working across disciplines in pursuit of next-generation science. Moreover, improving incentives, credit[14] and metrics for work on high-impact software (including publication and peer review), along with rewarding career paths (such as the research software engineering movement[15]) is increasingly important.

In recent years, international community members have established a range of grassroots organizations and projects to address these growing technical and social challenges in research software[16,17]. In their respective spheres of influence, these groups — including the Software Sustainability Institute (SSI), the US Research Software Sustainability Institute (URSSI), and the Better Scientific Software site — nurture communities, promote the growth of software ecosystems, and provide information about effective approaches for creating, sustaining and collaborating via scientific research software.

Conclusion and outlook

To fully exploit emerging exascale computational resources for next-generation computational science, we must explicitly acknowledge the critical role of software as the foundation of sustained collaboration and scientific progress. By embracing software ecosystem perspectives and tackling the intertwined technical, cognitive and social challenges related to high-quality, sustainable scientific software, we can fully integrate research advances across applied mathematics, computer science and domain sciences for next-generation computational science. While the difficulties of extreme-scale computing and large, multidisciplinary research projects intensify software challenges, these issues are relevant across all computing scales and project sizes, given universal increases in complexity and change in scientific computing, along with the need to ensure the trustworthiness of computational results.

We encourage exploration of the resources being created by various organizations that support software ecosystem health by promoting collaboration on community policies and best practices for scientific software. All of us as members of the computational science community can play important roles in addressing these technical, cognitive and social challenges in scientific software by catalyzing change in our own projects, institutions and communities. This work is essential so that we can collectively advance toward predictive computational science and reap its benefits for science and society. ❐

Lois Curfman McInnes[1]✉, Michael A. Heroux[2], Erik W. Draeger[3], Andrew Siegel[1], Susan Coghlan[4] and Katie Antypas[5]

[1]*Mathematics and Computer Science Division, Argonne National Laboratory, Lemont, IL, USA.* [2]*Center for Computing Research, Sandia National Laboratories, Albuquerque, NM, USA.* [3]*Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA, USA.* [4]*Argonne Leadership Computing Facility, Argonne National Laboratory, Lemont, IL, USA.* [5]*National Energy Research Scientific Computing Center, Lawrence Berkeley National Laboratory, Berkeley, CA, USA.*
✉*e-mail: curfman@anl.gov*

Published online: 22 February 2021
https://doi.org/10.1038/s43588-021-00033-y

References

1. Dongarra, J. et al. *Int. J. High Perform. Comput. Appl.* **25**, 3–60 (2011).
2. Hack, J. et al. *Crosscut Report: Exascale Requirements Review* (OSTI, 2017).
3. Heroux, M. et al. *ECP Software Technologies Capability Assessment Report 2.5* (ECP, 2020).
4. Rüde, U., Willcox, K., McInnes, L. C. & De Sterck, H. *SIAM Rev.* **60**, 707–754 (2018).
5. Hendrickson, B. et al. *ASCR@40: Highlights and Impacts of ASCR's Programs* (OSTI, 2020).
6. Keyes, D. E. et al. *Int. J. High Perform. Comput. Appl.* **27**, 4–83 (2013).
7. Keyes, D. et al. *Report of the National Science Foundation Advisory Committee on CyberInfrastructure* (Task Force on Software for Science and Engineering, 2011).
8. Kothe, D., Lee, S. & Qualters, I. *Comput. Sci. Eng.* **21**, 17–29 (2019).
9. Siegel, A. et al. *Early Application Results on Pre-exascale Architecture with Analysis of Performance Challenges and Projections, ECP Milestone Report PM-AD-1080* (ECP, 2020).
10. Alexander, F. et al. *Phil. Trans. R. Soc. A* **378**, 20190056 (2019).
11. Heroux, M. et al. *Advancing Scientific Productivity through Better Scientific Software: Developer Productivity and Software Sustainability Report, ECP-U-RPT-2020-0001* (OSTI, 2020).
12. Heroux, M. *Better Scientific Software* https://bssw.io/blog_posts/research-software-science-a-scientific-approach-to-understanding-and-improving-how-we-develop-and-use-software-for-research (2019).
13. Raybourn, E., Moulton, J. D. & Hungerford, A. In *HCI in Business, Government and Organizations. Information Systems and Analytics* (eds Nah, F. H. & Siau, K.) 408–421 (Springer, 2019).
14. Casari, A. et al. *Nat. Comput. Sci.* **1**, 2 (2021).
15. Hettrick, S. *Software Sustainability Institute* https://www.software.ac.uk/blog/2016-08-17-not-so-brief-history-research-software-engineers-0 (2016).
16. Katz, D. S. et al. *Comput. Sci. Eng.* **21**, 8–24 (2019).
17. McInnes, L. C., Katz, D. S. & Lathrop, S. *SIAM News* https://sinews.siam.org/Details-Page/computational-research-software-challenges-and-community-organizations-working-for-culture-change (2019).

Acknowledgements

We thank all those whose work is represented in this article for their commitment to realizing next-generation computational science. This work was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the US Department of Energy Office of Science and the National Nuclear Security Administration. This work was performed under the auspices of the US Department of Energy by Argonne National Laboratory under contract number DE-AC02-06CH11357 (L.C.M., A.S., S.C.), by Lawrence Berkeley National Laboratory under contract DE-AC02-05CH11231 (K.A.), by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344 (E.W.D.), and by Sandia National Laboratories under contract DEAC04-94AL85000 (M.A.H.). This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the US Department of Energy or the US government.

Author contributions

The authors serve as the leads of ECP focus areas: Software Technology, M.A.H. (director) and L.C.M. (deputy director); Application Development, A.S. (director) and E.W.D. (deputy director); Hardware and Integration, K.A. (director) and S.C. (deputy director). Their contributions to the ideas and writing of this article reflect their work in these roles.

Competing interests

The authors declare no competing interests.