Reinventing the Ecosystem to Improve HPC Software

Chunhua Liao[1]

Lawrence Livermore National Laboratory

Challenge

High Performance Computing is becoming increasingly difficult for users to use due to the ever-growing complexity of heterogeneous hardware and software components. Users often have to spend tremendous time on each step of HPC workflows, including learning how to use a system, adapting their codes to novel hardware features, debugging the execution, conducting performance analysis, applying various optimizations, and so on. This process typically needs to repeat multiple times before users can sufficiently utilize the supercomputers to obtain satisfactory results.

On the other hand, there are numerous software packages being developed to help users in various steps of their workflows to make the process more productive. However, the very existence of such software packages (including tools for performance, modernization and correctness) is not easily known to users. Many promising software packages have not received timely support to be robust and sustainable. This trend has to be reversed or only few users would enjoy the HPC systems.

Opportunity

There is an opportunity to reinvent the HPC ecosystem to have HPC of the users, by the users , and for the users. A well-designed ecosystem would speedup both hardware advances and software development based on timely user feedback. In particular, the HPC community should benefit from research and development of the following elements in the ecosystem:

1. Holistic metrics of success: The mainstream success metrics of HPC are machine-oriented, focusing on machine performance and efficiency. Human factors are not officially considered when ranking the best supercomputers as of today. A set of new holistic success metrics is needed to consider the total time, cost, and quality of results when evaluating HPC systems. The total time should include not just machine cycles, but also human cycles spent on training, thinking steps, number of keystrokes, mouse clicks or even cursor travel distances. With the right metrics, we can drive the HPC software development in the right directions.
2. A centralized marketplace. Currently, HPC Software packages are scattered all over the place, released through ad-hoc channels and formats. It is challenging for users to find the right software tools for their problems. The best practice of providing and finding software is through a common software marketplace. Prominent examples include Apple‘s app store and Google Play store. The HPC community needs to set up a similar marketplace for HPC software packages so providers can register software information while users can easily

[1] Contact: liao6@llnl.gov Prepared under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. LLNL-ABS-829260

search for what they want. Such a marketplace can also accept user reviews to give feedback to providers while facilitating communications among users.

3. A user-directed rewarding system. The reality of software development is that financial support is essential. Traditional funding systems use a top-down approach with many benefits. User-directed, bottom-up funding systems may provide additional support for software developers who are overlooked by higher management in typical funding cycles. The user-directed rewarding system can be implemented as peer-to-peer bonuses. Early experiments could use virtual currency in the community. People could earn virtual currency if they develop good software tools for the community. Or they could send earned virtual currency to other providers if they are happy with the software provided. This rewarding system can be made part of the software marketplace.
4. Standard benchmarks, competitions and state-of-the-art dashboards. If you can't measure it, you can't improve it. Performance-oriented benchmarks have been instrumental in HPC to drive the advance of supercomputers. We need similar investment in correctness and productivity benchmarks that can help reduce time-to-solution. Such benchmarks can be used to generate a dashboard comparing the state-of-the-art tools for a given problem, fostering fair competition and facilitating tool selections.
5. Software tools as composable services. Users often find that no single software tool is perfect in every aspect. Assembling multiple tools is a natural solution. Traditionally, tool researchers and developers are incentivized to focus on their own individual tools, without too much desire to work with other tools. To foster software tool development as composable microservices, we should invest in better interoperability of software tools, by promoting common APIs and result exchange formats for each type of software tools.

Why Now

“The best time to plant a tree was 20 years ago. The second best time is now.” We expect that most people would agree with the ideas of holistic success metrics, a common software marketplace, and complementary user-directed rewarding systems. There are also growing interests in benchmark development focusing on other than performance in the community. DataRaceBench[1] is an example benchmark focusing on correctness of data race detection tools, with a dashboard showing apple-to-apple comparison of existing tools to facilitate tool selections. A recent study[2] also shows that composing multiple data race detection tools can deliver better results, once common APIs and data exchange formats are defined.

References

1. Chunhua Liao, Pei-Hung Lin, Joshua Asplund, Markus Schordan and Ian Karlin, DataRaceBench: A Benchmark Suite for Systematic Evaluation of Data Race Detection Tools, The International Conference for High Performance Computing, Networking, Storage and Analysis, Denver, CO, Nov. 12-17, 2017
2. Yaying Shi, Anjia Wang, Yonghong Yan, Chunhua Liao, RDS: A Cloud-Based Metaservice for Detecting Data Races in Parallel Programs, 14th IEEE/ACM International Conference on Utility and Cloud Computing, University of Leicester, Leicester, UK, December 6-9, 2021