2024-2025 CRA Quadrennial Paper

Setting a Course for Post-Moore Software Performance

William Gropp (University of Illinois Urbana-Champaign), Randal Burns (Johns Hopkins University), Brian LaMacchia (Farcaster Consulting Group), Charles E. Leiserson (MIT), and Michela Taufer (University of Tennessee, Knoxville)

With Moore's Law having ended, the U.S. must pivot from relying on hardware improvements to investing heavily in software performance engineering (SPE) through research, education, and workforce development to maintain its technological edge, especially since few software engineers currently possess these critical skills.

For decades, the United States has enjoyed scientific, economic, and strategic advantages because of our ability to consistently engineer better and faster computer hardware. Not anymore. The era of cheap, exponential hardware advances driven by semiconductor miniaturization and Moore's Law has ended. To maintain our historical advantages, the U.S. now faces the challenge of designing applications that use available hardware more efficiently. Future progress in AI, biology, physics, economics, finance, chemistry, geoscience, and other fields all depends on fast computing, as does our national defense.

Researchers have identified three opportunity areas for significantly enhancing performance and reducing energy consumption, thus enabling continued growth in performance in the post-Moore era:

- **Algorithms**: The design of efficient, step-by-step procedures used to solve large-scale problems or perform modular tasks in complex workflows.
- **Hardware architecture**: The organization of computer components, such as processors, accelerators, and memory hierarchies.
- **Software**: Programs that run on hardware architectures to implement an application or system.

Opportunities for increased application and system performance exist in all three areas. Algorithm design, however, is limited to the relatively few individuals with specialized expertise and exceptional math skills. Hardware architecture design is similarly restricted to those with specialized expertise and access to state-of-the-art fabrication technology. In contrast, software is ubiquitous, and approximately 1.7 million individuals in the United States are employed as software developers. Software performance engineering (SPE) — optimizing code to run faster or use fewer resources — offers the most accessible opportunity for AI and other applications to improve performance and minimize energy consumption.

Unlike the faster hardware provided by Moore's Law, which sped up all applications, software performance improvements affect only the applications that run the specific code being optimized. Although shared libraries and platforms can be performance-engineered to run faster, achieving high-performance applications typically requires custom engineering of the applications themselves. For example, AI models can be trained significantly faster by employing techniques such as using simpler math calculations, distributing the work across multiple chips, and leveraging special software tools optimized for specific models. These techniques can improve model training speeds by a factor of 10x, and multiple optimizations can achieve gains of up to 150x. However, effectively implementing these techniques requires deep knowledge of computer hardware and the AI application used.

Maintaining the scientific, economic, and strategic advantages the United States enjoyed during the era of Moore's Law will require a large-scale effort across all U.S.-produced software. Unfortunately, few software engineers are proficient in SPE. Software-productivity toolchains for SPE remain underdeveloped and few universities teach even the foundational principles of SPE, let alone provide comprehensive curricula.

Addressing this challenge requires significant investments in three key areas:

1. **Prioritizing SPE research:** Productivity tools must be developed for SPE to ease the burden on average programmers and to relieve experts from painstaking and tedious work. Programming languages must evolve to enable the rapid development of efficient software. Computer system vendors must develop instrumentation frameworks that enable programmers and users to gain insight into the performance and energy consumption of their programs.

2. **Educating students in SPE:** Universities must be encouraged to develop undergraduate and graduate SPE courses and to devise comprehensive curricula. While research universities should lead this effort, they must collaborate with teaching-focused institutions and community colleges to share their curricular materials and ensure that SPE expertise does not remain a niche or esoteric skill.

3. **Upskilling the existing workforce:** The current workforce must acquire and practice SPE skills. Industry standards should be developed to report the performance and energy consumption of applications and systems, particularly those produced by the government's supply chain. Additionally, regulations or economic incentives should be implemented to promote software efficiency.

The types of research, education, and workforce development for software performance that we recommend below complement the national investment in the CHIPS and Science Act ("CHIPS", Pub.L. 117-167). CHIPS authorizes investments in industries and their workforces to position the U.S. competitively and secure the semiconductor supply chain. It also invests in future research into post-Moore's law architectures. Since the modern semiconductors produced in the U.S. under CHIPS will have leading-edge complexity, they will inherently possess a high degree of parallelism, making them difficult to program effectively with current software technology. A complementary effort focused on SPE will ensure that the CHIPS investment is not undermined by producing ostensibly high-performing hardware that cannot be effectively programmed to produce efficient, resilient, and secure applications.

Recommendations

Addressing the challenges above will require significant national investment in research, education, and industrial training. We propose the following recommendations:

1. Promote Research into SPE

SPE encompasses many techniques across computer science, including algorithms, AI, parallel computing, concurrency, caching, networking, databases, file systems, and GPUs. However, performance-engineered code is often complicated, which can conflict with other desirable software properties, such as correctness, security, modularity, maintainability, debuggability, portability, modifiability, robustness, and reliability. Research is needed to understand how programmers can structure fast and energy-efficient code while preserving these properties. Additionally, research is required to simplify and instrument systems so that their performance properties can be easily understood by programmers and end users.

2. Invest in Software-Productivity Tools for Performance

Software productivity tools for performance are outdated. Compilers, profilers, race detectors, scalability analyzers, memory-footprint analyzers were largely designed decades ago, during the Moore era, when software performance was less critical. Research is now needed to develop a new generation of performance tools that provide feedback to programmers about the resources their code consumes. Performance counters should be incorporated into chip architectures so tools can provide developers fine-grained performance feedback. AI tools, including large-language models, should be leveraged to accelerate the writing of high-performance code and optimize legacy code.

3. Pursue Research into Efficient Software for Heterogeneous Systems

Modern computing hardware is becoming increasingly complex as diverse components, such as multi-core CPUs, GPUs, and specialized accelerators, are integrated into traditional architectures to speed up computation. Current development tools, such as programming languages and compilers, are not optimized to structure or execute code efficiently for these systems. Investment in tools to simplify and optimize software development on heterogeneous computing architectures will improve and accelerate software development, making it easier for non-experts to program diverse computing systems. From multi-core CPUs to specialized GPUs, software solutions must be flexible and resilient, supporting the variety of architectures that will define the post-Moore era. As nonclassical architectures, including quantum and neuromorphic systems, are developed, research into writing software for these systems should include performance as a key metric.

4. Educate and Train the IT Workforce

Research universities should be encouraged to develop and offer SPE curricula within their full-time degree programs. Since these programs have a strong track record of placing students into engineering jobs that leverage their skills, widespread adoption across research universities will rapidly enhance the workforce. Additionally, these programs will create curricula, materials, and expertise that can be shared with other institutions of higher education, particularly teaching-focused colleges, community colleges, and professional education providers. The development of online courses, badges, and certificate offerings in the field of SPE should also be supported.

Conclusion

Science today relies on the availability of fast computing, as does the economic health of our nation. Moreover, in the face of growing global competition, the United States must act decisively to maintain its technological edge in the post-Moore era. To sustain a competitive advantage, we need a concerted national effort to invest in SPE research and to educate a workforce of programmers with SPE expertise. Without immediate investment, we risk jeopardizing the benefits of an economically strong and free society for ourselves and future generations.

This material is based upon work supported by the U.S. National Science Foundation (NSF) under Grant No. 2300842. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF.