Research Opportunities in Operating Systems for High-Performance Scientific Computing

Contributors: Pete Beckman (ANL), Ron Brightwell (SNL), Rudi Eigenmann (Univ of Delaware), Christian Engelmann (ORNL), Roberto Gioiosa (PNNL), Kamil Iskra (ANL), Shantenu Jha (BNL), Jack Lange (Univ of Pittsburgh), Tapasya Patki (LLNL), Kevin Pedretti (SNL)

DOE Point of Contact: Hal Finkel <hal.finkel@science.doe.gov>
May 18, 2021

As high-performance-computing (HPC) systems continue to evolve, with increasingly diverse and heterogeneous hardware, increasingly-complex requirements for security and multi-tenancy, and increasingly-demanding requirements for resiliency and monitoring, research in operating systems must continue to seed innovation to meet future needs. In the following, we survey current trends in system software and HPC systems for scientific computing, associated research challenges and open questions, infrastructure requirements for operating-systems research, communities who should be involved in that research, and the anticipated benefits of success.

The primary aspects of the modern computing ecosystem key to understanding future research opportunities in operating systems are:

- **Full-stack co-design for extreme heterogeneity and scalability**
- **Adaptive management and partitioning of resources**
- **Smart supercomputer systems and facilities**

Current Trends

Full-stack co-design for extreme heterogeneity and scalability

Increasing heterogeneity, increasing system scales, and increasing migration (e.g., due to increasing virtualization and VM-to-resource mapping) are requiring ever more advanced compilers, runtime libraries, and operating systems. Moreover, software must increasingly be optimized and specialized for different kinds of hardware. As the execution environment diversifies, including accelerators, network interfaces, and other components with their own programming environments, all aspects of the system are subject to specialization for particular components. For example, while task scheduling is generally managed directly by runtime

libraries at a low level, GPUs feature hardware thread/task scheduling at a higher level, necessitating special support within task-scheduling system abstractions.

Increasing specialization at the hardware level is driving system heterogeneity, and with specialization within the software required for optimal support of this heterogeneous hardware, systems research and development increasingly requires co-design of both the hardware and the software. With the advent of high-quality open-source hardware designs and fast cycle-accurate simulation tools, it is now possible to rapidly co-design pre-silicon hardware together with full software stacks. This opens up opportunities to co-design OS functionality and hardware designs in hours vs. years previously, with the ability to iterate hardware designs on the fly. For example, this capability was recently used to port KVM to the RISC-V architecture prior to final ratification of the RISC-V virtualization extensions, providing critical validation and feedback. There is great potential to co-design OS support for common interfaces to heterogeneous accelerators, virtual memory schemes, tightly coupled accelerators, and other novel hardware capabilities. As rapid, full-stack co-design continues to mature, it may become feasible to rapidly develop and economically deploy domain-optimized silicon targeted at DOE computing workloads, delivering world-class efficiency and performance.

Adaptive management and partitioning of resources

As an increasing number of extreme-scale applications are composed of multiple components, workflows that must often be “rewired” dynamically, the price of static decisions about workload placement and execution, and their management increases with scale and heterogeneity. Moreover, the computational capabilities of HPC compute nodes keep increasing, yet fully leveraging those capabilities using individual applications gets ever harder. Running a *diverse mix* of workloads on a node would improve the overall resource utilization, however this is often avoided, especially the multi-user scenarios, because this tends to introduce performance variability which makes it more difficult to reason about the performance of individual applications. Implementing multi-component workloads and workflows that break free from the current, static HPC-resources resource model is currently painful. Furthermore, as extreme-scale multi-component applications increasingly have the coupling between them vary over the application life-cycle, this imposes a further requirement of adaptive and dynamic resource management.

A lack of standards currently hampers the construction of composable, extensible scientific workflows: currently nearly every multi-component application and workflow assumes a different model of resource partitioning, selection, and availability, as well as different interfaces, resource state models, and resource / process management. This makes applications and workflows brittle and the barrier to portability and extensibility significant. Identifying generalized abstractions, interfaces and their implementations to minimize, if not reduce these barriers is imperative in order to reach scales and sophistication (over resource types, functionality needed etc.). These requirements are common across scales (“edge to exascale”), application types and time-scales.

The set of relevant resources continues to increase, now including CPU cores, shared caches, main memory, NIC, and heterogeneous hardware accelerators. This broadening collection of resources makes both adaptive management and partitioning even more difficult. Virtualization techniques are a popular existing solution in cloud environments, and offer partial solutions to some of these challenges, but they have traditionally been avoided in HPC due to overheads. HPC-specific multi-kernels can offer lower overheads at the cost of additional implementation complexity. Adoption of containers is increasing, also offering partial solutions to some management and partitioning challenges, but bringing new challenges around composability and efficient workflow integration.

One particularly-important resource to manage is memory, and not only is the memory topology landscape is already pretty complex (shared caches, NUMA, host vs accelerator memory), it is only expected to deepen in extremely-heterogeneous architectures (NV, PIM, disaggregated memory). Extreme-scale applications are already facing limited memory capacity and bandwidth, yet there are no widely accepted, standardized interfaces for exploiting deep memory hierarchies. If anything, the trend seems to be to attempt to hide the complexity, e.g., by presenting additional hierarchy levels as a transparent cache or through “unified” memory with transparent page migration behind the scenes. While such approaches often reduce the effort required for initial porting by application writers (who would rather not have to deal with additional hardware complexity), they can be harmful to overall performance due to suboptimal use of the capabilities made available by the hardware. There is an inherent conflict between how much of the memory hierarchy to hide from users (so as not to overly complicate the programming) vs how many novel capabilities to expose (to take advantage of the possible performance improvements).

Smart supercomputer systems and facilities

Operational intelligence (OI) optimizes the efficiency and effectiveness of systems and facilities using an observe-orient-decide-act loop for adaptation. The loop consists of operational data aggregation, operational data analytics, decision making that considers trade-offs, and operational configuration actions. Human-in-the-loop OI at different granularities is the current state of practice. 24/7 staff monitors and analyzes live system and facility data using dashboards and takes actions typically only in emergencies. Other operations staff monitors and analyzes daily, weekly and monthly system and facility data for maintenance and tuning. For example, adaptation to emerging reliability threats is mostly an entirely manual and sometimes ad hoc task and root cause analyses often take months [2]. Corrective actions are often highly limited, as the resilience “toolbox”, i.e., the number and types of corrective actions, employed in today’s computing systems and facilities is highly limited. It is unknown how this set of actions, and the data needed to inform them, must evolve into the future.

The DOE’s recent Computational Facilities Research Workshop report [1] identified smart systems and facilities as a broad challenge area with enabling automation and eliminating human-in-the-loop requirements as a cross-cutting theme. Smart supercomputer systems and facilities employ machine-in-the-loop OI for autonomous decision-making. Human-in-the-loop

needs are eliminated by an automated online control that is capable of aggregating operational data, understanding behavioral patterns and trade-offs, and taking appropriate operational actions in real time. It may be assisted by a “black box” AI trained offline with archived data and/or with synthetic data created by a digital twin. It may also rely on causal models, reinforcement learning or advanced statistical methods. Multiple independent control loops may address different aspects. For example, DOE’s flagship supercomputers may use such smart system capability for: (1) scheduling adversarial (network congestion) workloads, (2) dealing with anomalies (slow/failing equipment, job, service, network or storage), (3) tuning compiler/runtime parameters of applications, and (4) improving energy efficiency. It may also be used in the entire supercomputing facility, such as for improving the operation of network, storage, cooling and other systems.

Research Challenges and Open Questions

Research in a number of these areas is inherently multidisciplinary, and as a result, must overcome coordination challenges. For both research, and later development activities, defining interface standards is critical for enabling the decoupling of different parts of the software stack. The tradeoff between flexibility needed for innovation versus constraints needed for overall coordination and project efficiency need to be carefully managed.

Full-stack co-design for extreme heterogeneity and scalability

Operating-systems research in the context of full-stack co-design can address:

- Rapidly developing novel accelerator hardware and accompanying software support
- Common interfaces to heterogeneous hardware enabling effective OS management
- Autonomous resource management that reduces the burden on users
- System-wide security that enables breaking free from the “root is special” model in HPC

In this context, specific research questions include:

- How should we think about the OS managing extremely heterogeneous systems, and what new hardware and software mechanisms are needed to enable this?
- What are the right interfaces to communicate the needed information between programs, compilers, runtime/OS, monitoring subsystem architecture?
- Can autonomous resource management, assisted by co-designed hardware and OS software, reduce the user burden for managing extremely heterogeneous devices and memories?
- Can this be done portably, such that difficult porting efforts are not required when moving from one system to the next?
- How to design an intelligent memory management featuring automated, on-the-fly data optimization (compression, transformations, placement)?
- What role can the OS play in identifying inefficient resource usage and suggesting or automating potential improvements?

- What new hardware capabilities could help the OS do this?
- Through hardware and software co-design, can we enable system-wide security such that root is no longer a special thing on HPC systems, similar to the case on public clouds?
- Would this reduce the facility burden for managing security issues, as well as enable new types of cloud-native workloads to run on DOE supercomputers?

Adaptive management and partitioning of resources

Future research is needed to address a number of open questions regarding how applications, programming environments, and systems can exchange information and cooperatively react to changing workloads and resources. These questions include:

- How to extend programming languages and compilers so that users can express the user-relevant information?
- What is the communication architecture (i.e., what information is created, communicated, and consumed where and when)?
- How can current programming methods and compiler optimizations be extended to take advantage of the new information?
- How can dynamic optimization systems be constructed that adapt at runtime and in the field to changing environments?
- How to build performance models and decision support to guide users and compilers in these methods and optimizations?
- How to construct data-centric abstractions that are application-aware and that present a range of interfaces for a right fit to every application and runtime system, from fully transparent to fully explicit ones?
- How to enable integration across all of the on-node devices, including function offloading to the NIC?
- How to reintroduce resource partitioning capabilities in HPC environments, overcoming concerns regarding security and reproducibility?
- How to support containers within an HPC environment, including workflow integration and performance monitoring?
- What is the performance price of static execution and resource management as a function of scale and heterogeneity?
- What are “intrinsic” scales of resource partitioning as a function of scale and dynamism?
- In the presence of adaptive execution and dynamic resource behavior, what is the trade-off between global versus partitioned resource management at exascale?
- What are the challenges of providing the application complete control of the resources?
- How is information propagated across resources partitions?
- How can adaptive execution be implemented without incurring significant overhead?

Smart supercomputer systems and facilities

Specific research challenges to enable enhanced OI and intelligent facilities are:

- Autonomous resource management at different granularities: programming model runtime, node OS, global OS and facility
- Machine-in-the-loop feedback through operational intelligence: monitoring, operational data analytics, autonomous decision making and adaptive resource management
- Improving operational productivity and lowering operational costs through corrective actions while understanding performance, power consumption and resilience trade-offs
- Autonomous adaptation to system properties and application needs

There are a number of open research questions to make machine-in-the-loop operational intelligence for HPC systems and facilities a reality, including: (1) identification of relevant monitoring data for specific control problems, (2) offline vs. online operational data analytics, learning and decision making, (3) understanding and modeling the involved trade-offs, (4) leveraging community and industry software for reuse and maintainability, and (5) enabling real-time control.

Infrastructure Requirements

The fact that essentially all system stack components are involved and need to be experimented with puts high demands on the infrastructure. It needs to allow access and modifications across the system stack, include highly diverse platforms, and allow migration between classical HPC centers, Cloud platforms, and edge devices. It also needs to be as close to a production environment as possible, for adaption to be validated properly.

For full-stack co-design, enabling exploration of pre-silicon hardware designs with full software stacks requires access to FPGA accelerated simulation tools to deliver the required levels of performance. These may be hosted in the cloud, as with the Berkeley FireSim tool, or located on premise with additional porting work. A cluster of such FPGA resources would be required to simulate more than a handful of nodes.

Benefits of Success

Full-stack co-design for extreme heterogeneity and scalability

Success would demonstrate the ability to co-design novel OS functionality with supporting hardware designs in hours vs. years with current approaches. Proving out new OS ideas is difficult due to researchers having limited ability to affect hardware designs and limited access to large scale HPC platforms by which custom OS software can be booted. Performing experiments via FPGA-accelerated simulation tools attacks both of these challenges. Success would produce highly credible OS functionality, hardware designs, and actionable information which could be used to more effectively influence our vendor partners, ultimately leading to higher performing and more efficient DOE supercomputer platforms.

Adaptive management and partitioning of resources

A defining feature of success will be middleware capabilities that provide the abstractions and interfaces for distributed and high-performance resource management, that allow a broad range of multi-component applications and workflows that are agnostic to the specifics of underlying resources and platforms, while making it easier to scale up and out. Moreover, success will be characterized by significant performance improvements, improved resource utilization (less movement, better power consumption), and improved productivity for users/developers when porting to new platforms. Resource-partitioning improvements will result in better utilization of available node resources, improved performance for dynamic workloads (including ensemble calculations and AI), and opening HPC systems to a broader, more diverse set of workloads.

Smart supercomputer systems and facilities

Machine-in-the-loop operational intelligence improves productivity and lowers costs. It optimizes the efficiency and effectiveness of computing systems and facilities, enabling faster science breakthroughs at lower costs. For example, (1) adversarial workloads are scheduled better to avoid network congestion, (2) emerging reliability threats are recognized early and handled adequately to avoid unnecessary system downtime or degradation, (3) compiler and runtime parameters are tuned correctly to avoid wasting resources, and (4) workload requirements are understood better to inform the design and deployment process of next-generation systems.

Contributing Research Communities

Full-stack co-design for extreme heterogeneity and scalability

This work leverages a vibrant open-source hardware community in academia, which is producing increasingly high-quality hardware designs and high-productivity hardware development tools. It would additionally leverage a new class of hardware vendors that are emerging to help customers develop domain optimized hardware targeting their particular workloads. While HPC is not the direct focus of these communities, an effort within DOE could provide this linkage and help ensure the needs of DOE supercomputer facilities are addressed.

Adaptive management and partitioning of resources

Research communities across the entire system stack will need to collaborate, from low-level architecture to applications; from single platforms to wide-area distributed systems; from edge to Cloud, to traditional HPC systems. Designing middleware to support adaptive execution and dynamic resources, requires in addition to the DOE applications, facilities, and systems researchers, collaboration with the Software Engineering for Adaptive and Self-managing Systems (see: https://conf.researchr.org/home/seams-2021), and the community of middleware design principles, programming abstractions and paradigms for reconfigurable, adaptable, and reflective approaches (see: ACM Middleware).

For partitioning specifically, the cloud community faces and—to a greater or lesser extent—successfully solves many of these issues, though possibly at significant performance overhead costs. The execution model of cloud services is however more oriented towards duplication/shutdown-restart than long-running applications adapting at runtime. The edge community also addresses similar needs.

Smart supercomputer systems and facilities

This work involves the HPC node OS community, the parallel programming runtime community, the HPC global OS community, HPC operations personnel at computing facilities, the AI/ML (in control systems) community and the decision sciences (in control systems) community. It spans the DOE laboratories, academia and industry.

References

[1] DOE National Laboratories' Computational Facilities – Research Workshop Report. ANL/MCS-TM-388. February 2020. URL https://publications.anl.gov/anlpubs/2020/02/158604.pdf

[2] G. Ostrouchov, D. Maxwell, R. Ashraf, C. Engelmann, M. Shankar, and J. Rogers. GPU Lifetimes on Titan Supercomputer: Survival Analysis and Reliability. In Proceedings of the 33rd IEEE/ACM International Conference on High Performance Computing, Networking, Storage and Analysis (SC) 2020, pages 41:1-14, Atlanta, GA, USA, November 15-20, 2020. DOI 10.1109/SC41405.2020.00045