

OS Research: A Critical Requirement for HEC Revitalization

Ron Brightwell; 505-844-2099
Principal Member of Technical Staff
Scalable Systems Integration, Sandia National Laboratories, P.O. Box 5800
Albuquerque, NM 87185-1110
bright@cs.sandia.gov

Orran Krieger; 617-693-4374
Research Staff Member and Manager Advanced Operating Systems
IBM T.J. Watson Research Center
Yorktown Heights, NY, 10598
okrieg@us.ibm.com

Arthur B. Maccabe; 505-277-6504
Associate Professor of Computer Science and Associate Director HPC@UNM
Computer Science Department, University of New Mexico
Albuquerque, NM 87131-1386
maccabe@cs.unm.edu

May 16, 2003

In this white paper, we argue that operating systems are a fundamental technology critical to the future success of high-end computing (HEC) systems. We believe that this technology area has largely been ignored for the past decade and that this lack of attention has had a direct impact on the effectiveness of current HEC systems. To overcome this problem for future HEC systems, there must be a concerted and coordinated effort in operating systems research.

The current trend in large-scale HEC systems is to leverage operating systems developed for other areas of computing. An informal survey of the most powerful HEC systems currently deployed¹ reveals that all but a few of these machines run commodity operating systems that

were not specifically designed for large-scale, parallel computing platforms.

Since system software is only one component of a complex HEC system, it is non-trivial to gather empirical evidence that pinpoints specific operating system limitations. However, we note that the Sandia's ASCI/Red and the Cray T3E have demonstrated the highest level of scaling efficiency for several complex scientific applications and that both of these platforms employed specialized operating systems on their compute nodes.

We believe that the current performance and scalability limitations of HEC platforms are, in large part, due to a lack of fundamental OS research activities in the previous decade. The OS research community has become increasingly stagnant and has had little impact on current-

¹<http://www.top500.org>

generation HEC systems.

While we firmly believe in the value of specialized approaches, we also recognize that specialized operating systems cannot be sustained indefinitely. It is therefore important that research in the development of specialized operating systems be augmented by research that seeks to consolidate key features of specialized operating systems with main-stream, commodity operating systems.

The rest of this paper is organized as follows. We begin by considering the current state of OS research activities. We then identify key impediments to core OS research and identify OS research problems that are critical for HEC. We then describe a plan for addressing these problems. Finally, we describe two related activities.

Current State

Core OS research for HEC systems is essentially non-existent. This lack of activity in HEC OS research is largely a consequence of the lack of core research activity in the broader OS community. Current OS research activities are either focused on incremental changes to existing systems (e.g., adding transparent support for large memory pages) or are focused on non-core issues (e.g., file systems). There are only a handful of small research projects focused on addressing OS issues for large-scale HEC systems.

In the 90s, the OS research community became marginalized by the growing importance of commodity operating systems. The rise of open source operating systems like Linux held the promise that the community could be revitalized. Linux provides a real environment and could be used for core systems research. Unfortunately, instead of revitalizing the OS research community, Linux has done just the opposite.

We identify a variety of factors have become impediments to core OS research:

Services. It is no longer feasible to implement a special-purpose OS to solve a specialized set of problems. Users and application develop-

ers increasingly demand the rich set of services—even when they are working in a highly specialized environment like HEC.

Standards. “To be a viable computer system, one must honor a huge list of large, and often changing, standards: TCP/IP, HTTP, HTML, XML, CORBA, Unicode, POSIX, NFS, SMB, MIME, POP, IMAP, X, A huge amount of work, but if you don’t honor the standards, you’re marginalized.”². Such work is impossible in academia and prohibitive in all but the largest industrial environments.

Application portability. Introducing new interfaces is difficult. Since applications are required to be portable, modifying them to exploit a new interface that is not widely support is nearly impossible.

Benchmarks. Benchmarks are important driver for every discipline. While it is feasible to change open source applications for experimentation, complex applications are often difficult for OS researchers to customize in order to exploit new interfaces. As a result OS researchers have often focused on benchmarks that they can analyze rather than on benchmarks that are relevant to the HEC community.

Hardware access. In the past, changes in hardware have driven innovation in system software. This drive has faltered in the last decade. Most researchers only have access to sequential or very small parallel systems. As an example, the group responsible for developing OSF-1/AD for the Intel Paragon never had access to a system with more than 32 nodes. Given this lack of a reasonable development environment, it is not surprising that early versions of OSF-1/AD failed spectacularly on large systems.

Moore’s law. Performance always has been and will continue to be central to core OS research. However, Moore’s law has often led to designs that sacrifice performance for the sake of additional functionality, since processor

²Rob Pike, “Systems Software Research is Irrelevant,” <http://freshmeat.net/articles/view/175/>

performance has made OS performance a secondary concern. This has led to a feedback loop, where operating systems become more complex due to inefficient implementation of new features, making the task of the operating system researcher more and more difficult.

Using open source operating systems like Linux and Free BSD as research platforms might address these issues; however, these systems introduce several new problems:

Structure. The internal structure of these systems makes it difficult to study anything but incremental changes to the OS. As an example, there are many thousands of lines of code in the Linux kernel that understand the socket data structure. Any research that requires changes to this data structure is prohibitive. Where Linux presents a well-defined interface (e.g., Vnode, Streams, device drivers) there has been a great deal of innovation. Unfortunately, these interfaces are not common in current versions of the Linux kernel.

Acceptance. A new idea—no matter how beneficial it proves to be—is difficult to get accepted into standard Linux distributions. While the community operates as a meritocracy, the metric is performance on desktops and small servers. Changes that have any negative impact, even in the sense of complexity, are not accepted.

Culture. Finally, there is a serious cultural disconnect between the research community and the Linux community. The OS research community was incapable of producing the production quality code needed to successfully clone Unix. As a result, the Linux community tends to dismiss academic research. They are largely unaware of, and typically do not cite, the fundamental papers that explored the ideas they reimplement. Without much chance of getting innovative ideas accepted into Linux, and without papers or even attribution, the research community will continue to be detached from the Linux community.

Challenges

Adaptable to radical architecture changes. The structure of current operating systems stifles hardware innovation. As an example, the Linux hardware abstraction layer has a multi-level page table in it that maps directly to the Intel x86 processor architecture. Linux has been ported to other architectures by making those architectures look like the x86, hiding the advantages of hardware specific features. Over time, hardware architects have given up innovating in spaces that would require OS support.

Architectural innovation is critical to HEC. It has long been projected that Moore's law is nearing its end. If this is the case, new architectural innovation will be needed to find ways to improve performance. If, on the other hand, Moore's law continues for another decade, this will result in ever increasing memory and I/O latency. We have probably reached the limit on what hardware can do to hide this increasing latency, and major innovation in system software will be required (e.g., prefetching, cache injection, explicit management of the memory hierarchy, scheduling tasks without conflicting requirements, collecting all the complex hardware monitoring information required to drive above policies).

We can easily project several other areas of HEC hardware development that will require fundamental innovation in operating systems. Examples include multi-processor cores, processor integrated memory (PIM) architectures, and architectural enhancements to support power-aware computing.

Support multiple management policies. The structure of our current operating systems stifles innovation in application development. For each resource, the operating system implements a specific set of management policies. If the policy implemented does not match the needs of an application, that application suffers. In many cases, this means that performance sensitive applications incur unnecessary

complexity to implement their own version of resource management to match their needs. In these cases, performance is substantially worse than if the OS implemented the right policy in the first place.

Support for specialized HEC needs. The HEC community often has specialized needs, and those needs are not met by existing operating systems. For example, there are currently several efforts to develop specialized programming models for HEC systems. These new models may have specialized needs are unlikely to be met by traditional operating systems.

Support for sophisticated services. The HEC community requires much more sophisticated OS infrastructure than was needed in the past. As large systems become more and more complex, scientists with specialized skills are less and less capable of efficiently exploiting the hardware. Moreover, the need for sophisticated services like visualization requires higher-level services.

Consolidation with mainstream technology. As the need for more sophisticated OS services becomes critical, it is important that the operating systems for the HEC community converge with the more widely available commercial systems in order to: 1) exploit the larger community of programmers available in the commercial and academic sector, 2) exploit the programming model and other innovations that have raised the level of abstraction for programmers for general purpose systems, 3) avoid re-implementing the wheel, and 4) enable OS researchers to help the HEC community solve its problems. Such a convergence will itself require major innovation in the way in which we design operating systems.

Support for fault tolerance. We are rapidly approaching the point where our largest systems will encounter “interrupts” in standard operation (i.e., faults). This will require changes at all levels from applications, to runtime systems, to operating systems, to basic hardware structures.

Support for sophisticated security. As HEC

systems are connected to the broader infrastructure, new applications will demand new innovations, e.g., interactive HEC to run quick parameter studies. Effective security will become critical as highly secure applications are run. Management of large-scale systems is currently very complex. As system scale continues to increase, the simple admission control and other centralized servers in our existing HEC systems will be less and less acceptable.

Proposed Research Structure

With the right investment, the future will not be as bleak as the past. Linux provides us with a huge base of code and a critical set of applications. Linux has also demonstrated the effectiveness of the open-source model. Moreover, the interest in Linux has resulted in a large group of people with OS development skills. The Linux community has essentially finished the task of cloning Unix, and new grand challenges appeal to many leaders in that community. Finally, the research community is back to doing research now that the startup phenomena of the 90s is over.

In putting together an action plan, we focus on elements that will enable a flourishing research community, not on the government picking a winner and forcing the community to adapt it. Previous attempts at early unification (e.g., Mach), while well intentioned, have been counter-productive to the community as a whole. In broad overview, we propose that the following steps be taken:

Define interesting problems. It is critical to provide support to enable OS researchers to be able to experiment with workloads that are of interest to the HEC community. This might include providing benchmark suites and applications on the web that are simple enough to be modified but are illustrative of problems important to the HEC community. These should include mixed workloads of interactive and batch jobs, as well as jobs that stress OS services (e.g.,

I/O intensive, short running large jobs). Also, provide some key application experts that can help modify critical applications to exploit new OS interfaces and provide the information OS researchers need to port these applications to their platforms. Finally, publish challenges that will target OS researchers at the important problems.

Provide testbeds. It is difficult to study the effects of new and future hardware changes as well as issues of scale on the systems available to them. We need to invest in open-source simulators that provide a good environment for OS development and that can be used to study the effects of hardware tradeoffs and the value of performance monitoring functions on OS policy. We need to provide both small and large instances of “bootable” systems, available through the Internet, with the capabilities required to explore interesting problems. This must include instances of the largest machines, both SMP and distributed memory systems.

Fundamental innovations. It is critical to fund a small number of highly innovative OS projects that start from scratch. Projects that componentize OS services, especially in Linux, are critical. As we have noted, wherever well defined interfaces exist in Linux a tremendous amount of innovation has resulted. This might also include work on very lightweight operating systems that enable a small number of applications on experimental hardware.

Low-level mechanisms. Resource protection and mediation in the presence of competition for resources are also fundamental to OS research and need to be supported. With advent of gigabit networking technologies, the operating system was viewed as the critical bottleneck in the delivery of bandwidth to the application. Rather than fix the operating system, developers decided to bypass the OS. While understandable, this approach means that the OS can no longer serve its primary role. Examples from our research include the Quintessential kernel (the basis for Puma) and hypervisors.

Bridge research and development. Without the possibility of deployment, OS research quickly becomes irrelevant. We believe it is necessary to provide a bridge between the Linux and research communities. For example, have a team of developers integrated and accepted into the Linux community that can aid researchers in getting their innovations deployed. Support could also be provided for organizing workshops and internet forums to pull together Linux and academic researchers.

Reduce legal barriers. Finally, we believe it is necessary to reduce the legal barriers. In government procurements, insist on open source. There also needs to be a streamlined process, e.g., an appeal body with legal authority to deal with the intellectual property issues related to the GNU General Public License (GPL). The lawsuit by the SCO group against IBM for contributions to Linux is having a chilling effect on the ability of corporations to contribute innovations to open-source. The same large corporations critical to solve some of the major problems in OS research are the ones most threatened by litigation.

Related Activities

Two activities, FAST-OS (Forum to Address Scalable Technology for runtime and Operating Systems) and DARPA HPCS (High Productivity Computing Systems), that are relevant. Fred Johnson at the Department of Energy initiated FAST-OS in response to the issues we have identified. This is an important first step, but it needs to have multi-agency support to ensure continuing relevance and a sufficiently broad mission.

The DARPA HPCS initiative will impact many of the issues we have identified. Unfortunately, most of the vendors participating in this initiative have not been involved in FAST-OS (Cray and IBM are notable exceptions). It would be a great start if future initiatives explicitly called for OS research along with architectural research.