SAND2011-1400P

Systems Software

NNSA working group participants

- Ron Minnich, Bob Ballance, Ron Brightwell, Sue Kelly (SNL)
- Chris Dunlap, Jim Garlick, Maya Gokhale, Pam Hamilton (LLNL)
- Mike Lang, Scott Pakin (LANL)

Scope

Systems software is comprised of: kernel, communication libraries, job scheduler, systems health and systems monitoring. Out of scope are: compilers and math libraries.

General

The pragmatic software environment cobbled together in the early 1990s for the first MPPs has not evolved significantly since. The brittle “perfect machine” model, OS bypass in lieu of an OS abstraction for message passing, and user-initiated checkpoint/restart are illustrative of these early approaches still in use on the largest MPP systems. These approaches have endured because the last 1000X increase in capability has largely arisen from increased node performance without substantially increased node counts. The windfall provided by Moore’s Law has mitigated the need for dramatic improvements in the scalability of the software stack.

Unfortunately, the end of Dennard scaling means clock rates have plateaued, and future systems may need to reduce them in order to curb the energy demands of dramatically larger systems. Increased capability will now require massive increases in the number of CPU cores -- a trend that has already begun in petascale systems. Exascale systems as envisioned by DARPA[citation] and others will require distributed operating system software that is fault-tolerant and capable of managing power and data movement as schedulable resources in a heterogeneous environment. The operating system will need to provide applications with the abstraction of a perfect machine, while managing the increased complexity of adaptive management and automatic tuning to optimize resource utilization.

Plans for evolutionary change, or incremental modifications to existing software, should face much sharper scrutiny than plans based on revolutionary change. We plan to keep this in mind for SSW.

Assessment of current effort within ASC program (and/or community)

There are several Systems Software efforts of note funded within the NNSA labs:

- The Kitten LWK has been written at Sandia and runs on the Cray XT series machines. It is the latest in a series of LWKs.

Non-NNSA funding for systems software at the NNSA labs includes:

- The HARE project includes Sandia, IBM, and Bell Labs. It is exploring the use of a non-Linux, non-LWK, "lightweight general purpose" kernel and has been ported to the Blue Gene series of machines. HARE ends this year.
- The ZeptOS project at ANL explores the use of Linux on Blue Gene machines, and has shown

very good results with a modified Linux kernel.
- The FOX project, which includes SNL (lead), PNL, LLNL, IBM, Bell Labs, BU, and OSU, is exploring new models of application runtime, fault tolerance, and operating systems concurrently. One of the operating systems is the new Osprey operating system, which replaces the Unix open/close/read/write model with a process IPC model.

Specific technical challenges to get to exascale

- Much of DOE HPC work to date has focused on the adaptation of consumer-grade software and hardware to HPC needs. It is not clear that this trend will easily continue to the exascale. Our exascale programs envisage parallelism for single programs on a scale thousands of times larger than other uses, including large-scale commercial applications such as Google.
- System software that manages millions of nodes is fundamentally different from that which manages thousands. We question whether replicating a complex operating system like Linux a million times will be feasible.
- We need a tighter integration to the programming model needs rather than providing a generic interface that may not be appropriate to application needs. Put another way, SSW should provide abstractions that are useful to the applications.
- Power management is another area of concern. The power subsystems information is not available to the parallel runtime. We need power management interfaces that are efficient, far faster than the multi-second power management interfaces available today on COTS PC systems, and capable of providing both global (whole machine) and local (per-node, per-board, per-rack) power information and control.
- Hardware features may be offline for power reasons, and a runtime should take this into account in the scheduler.
- A unified framework for event collection and propagation needs to be standardized upon to allow data from disparate components to be accumulated and synthesized by the RAS subsystems.
- The RAS subsystems function in an out-of-sight manner: information is gathered and forwarded to a central facility, and disseminated from there to programs that need it. This loop is too slow, and does not provide for fast reaction to local events. We feel that it will be essential to extend RAS to allow local decisions to be made and managed by the runtime.
- How do you keep the system running and the target computation(s) making forward progress in the presence of constant failures? Reliability and resilience issues will play a huge role at exascale. A change in resilience may affect everything: programming models, languages, runtimes, file systems, and operating systems. The RAS subsystems may allow for the proactive detection of select failures and the reallocation of resources.
- We will need to work out OS interfaces to allow runtime and application to expect and react to failures. The current mechanisms are extremely heavy-weight, usually involving the use of signal(). The OS must support lower-overhead paths.
- Dynamic management of resources figures heavily into the future. We need algorithms, applications, runtimes, and systems software that can respond rapidly to changing environments. One unknown question is just how rapid that has to be -- measured in milliseconds, seconds, or minutes?
- Current systems are perfectly uniform. In the future we will see heterogeneity everywhere, in both processors and network. It is probably time to plan for runtimes that can deal with "lumpy", i.e. non-uniform systems.
- Better access to hardware info is essential, but it must also be more efficient, faster access. Hardware notification of some events either does not happen at present or is vectored through the high-overhead firmware-based mechanisms.

- File system APIs have been driven by the Unix model. This model has led to the development of parallel file systems that, in turn, drive new operating systems to supporting the Unix API to support the file systems that ... It is time to get off this treadmill. We need to work out new, scalable file system APIs that are not tied to a 40-year-old API.
- Memory management has been stuck on the Unix model for decades: unique address spaces per process. This model in turn has led to the need for pinned virtual memory, and a host of other problems that plague our OS and runtime. It is time to look at richer models, including Single Address Space systems, which can allow for more efficient, lower-overhead network I/O.
- 3D Memory may be in our future, as may other even more complex memory systems. What is a workable model for this memory? Do we treat it as another level of cache, and consider DRAM as a set of cache lines? Is it application managed? Is it coherent? We need some strawman APIs, as well as simulation efforts and early hardware access, to figure this out.
- Threading is viewed as useful, but the Posix/Pthreads API is not seen to be a good fit for exascale. Something better is needed, possibly inspired by new efforts such as Google's concurrency model for the new Go language.
- Increased complexity and decentralization will necessitate “defense-in-depth” concepts and security features being built into many components of the system in a manner that does not impinge on performance.

R&D Opportunities for the near term (next 3 years)

- It might be worth porting a research kernel such as Kitten to BG/P to allow explorations of its capabilities; a partnership with IBM to explore extensions to its Compute Node Kernel with some HARE and/or Kitten capabilities could yield valuable results.
- Set up simulators to allow booting OSes and testing ideas.
- Use virtualization on systems such as Jaguar to support booting SSW on a large scale.
- Open up HPC resources for research -- INCITE has been useful in the past but few if any systems software projects made it into INCITE this year.
- We should continue to encourage development of mini applications and synthetic workloads amenable to testing in simulation.
- Need to provide interfaces for tools and programming models.
- Need to prototype and test APIs for data movement, resilience, power, and control systems.
- Need well defined requirements for visualization and analysis. Will these require multiple process support per node? Will we need to partition on a node basis?
- In any event, the more risky research started with FAST-OS, which ends this year, should built open to support exploration of new OS models that will have a better chance of scaling to future needs.