



High Performance Linux: Dream or Reality?

Ron Brightwell

Sandia National Labs

Scalable Computing Systems Department

rbbrih@sandia.gov



What is High Performance?

- A big Linpack number?
- The lowest ping-pong latency?
- The highest ping-pong bandwidth?
- How many jobs get through the system?
- How quickly **my** job gets through the system?

- Performance is application-specific



What is Linux?

- What you get from kernel.org?
- What you get from SUSE, RedHat, etc.?
- What you get from (pick a Linux vendor) and
 - Apply patches for (pick your HPC network)
 - Apply patches for (pick your disk vendor)
 - Apply patches for (pick your parallel filesystem)
- Is Bproc still Linux?

- Or is it the just the development environment?
- Or the application binary interface?



Linux Hardware Support

- **Linux is great for my desktop/laptop**
- **If you wait long enough, you're hardware will eventually be supported**
- **Not a good model for HPC**
- **But how much hardware support is needed?**



Deterministic Performance

- **Avoid work unrelated to the computation**
 - Interrupts to keep time
 - Daemons: `init`, `inetd`, `ipciod`
 - Kernel threads: `kswapd`, `kflushd`, `kupdate`, `kpiod`
 - Inappropriate resource management strategies
 - “Rogue OS” effects



Linux Memory Management

- **How much usable memory is there?**
- **Still issues with page pinning**
 - Linux wants to manage all of your pages for you
 - RDMA is a great model for high-performance networking
 - As long as a process' address map stays consistent
 - Linux memory management strategies are based on optimizing system performance by re-mapping memory pages frequently
 - Linux kernel developers don't like losing control of resources

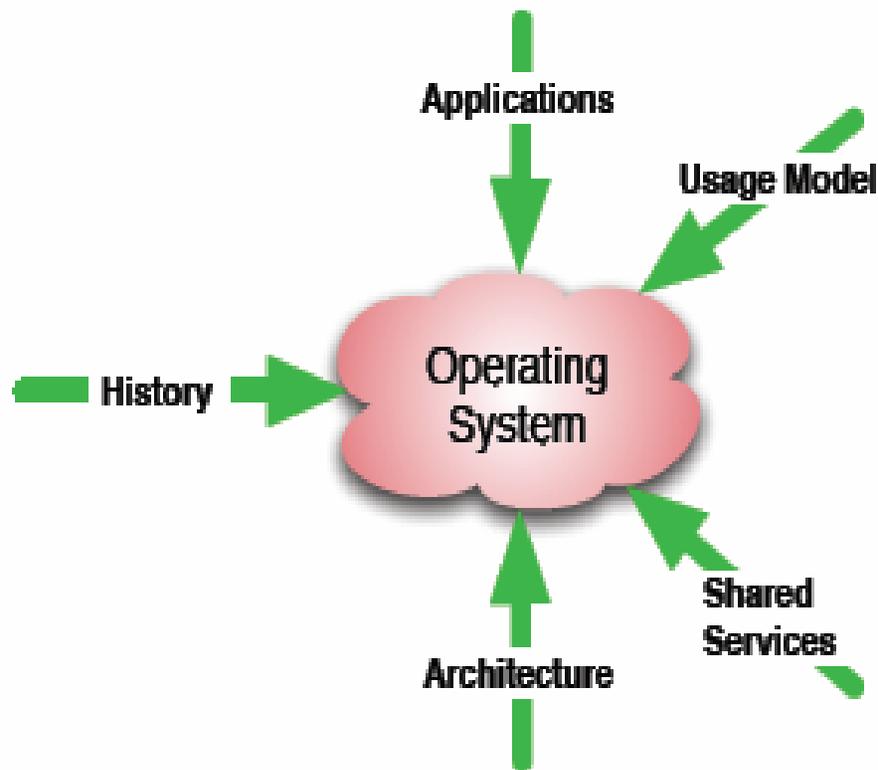


Linux Social Issues

- **Kernel development moves fast**
 - Significant resources needed to keep up
- **Distributions and development environments also change frequently**
 - Tool vendors have trouble keeping up
- **Server vs. multimedia desktop**
 - Not HPC



Lightweight Kernel Influences

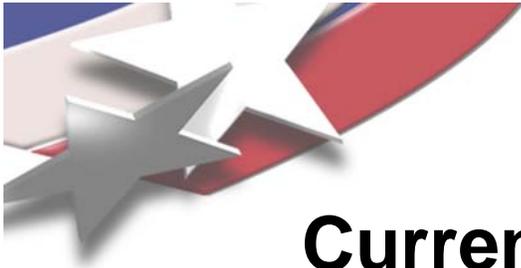


- **Lightweight OS**
 - Small collection of apps
 - Single programming model
 - Single architecture
 - Single usage model
 - Small set of shared services
 - No history
- **Puma/Cougar/Catamount**
 - MPI
 - Distributed memory
 - Space-shared
 - Parallel file system
 - Batch scheduler



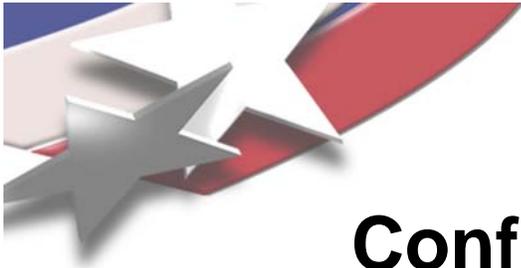
Other Issues

- **General-purpose operating systems**
 - **Generality comes at the cost of performance for all applications**
 - **POSIX**
 - **Assume a generic architectural model**
 - **Difficult to expose novel features**
- **Lightweight operating systems**
 - **Limited functionality**
 - **Difficult to add new features**
 - **Designed to be used in the context of a specific usage model**
- **OS is an impediment to new architectures and programming models**



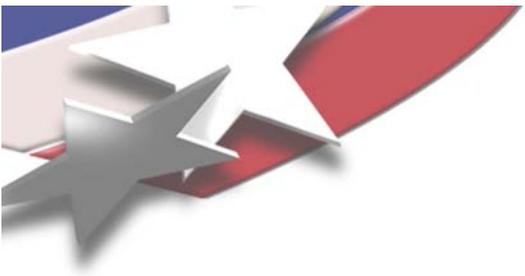
Current and Future System Demands

- **Architecture**
 - Modern ultrascale machines have widely varying system-level and node-level architectures
 - Systems will have further hardware advances (e.g., multi-core chips, PIMs)
- **Programming model**
 - MPI, Threads, OpenMP, PGAS, ...
- **External services**
 - Parallel file systems, dynamic libraries, checkpoint/restart, ...
- **Usage model**
 - Single, large, long-running simulation
 - Parameter studies with thousands of single-processor, short-running jobs



Configurable OS Research Project

- **Realize a new generation of scalable, efficient, reliable, easy to use operating systems for a broad range of future ultrascale high-end computing systems based on both conventional and advanced hardware architectures and in support of diverse, current and emerging parallel programming models.**
- **Devise and implement a prototype system that provides a framework for automatically configuring and building lightweight operating and runtime system based on the requirements presented by an application, system usage model, system architecture, and the combined needs for shared services.**



Approach

- **Define and build a collection of micro-services**
 - **Small components with well-defined interfaces**
 - **Implement an indivisible portion of service semantics**
 - **Fundamental elements of composition and re-use**
- **Combine micro-services specifically for an application and a target platform**
- **Develop tools to facilitate the synthesis of required micro-services**



Building Custom Operating/Runtime Systems

